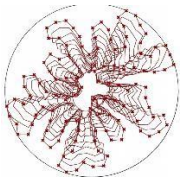
Universidad de Las Américas

Máster en Diseño Arquitectónico Avanzado -MADAA

DISEÑO ALGORITMICO PARA JUEGOS INFANTILES: SISTEMAS DE CRECIMIENTO DE CORALES

*Johanna Gabriela Garces Villarreal

2024



MADAA 2022-2024

MÁSTER EN DISEÑO ARQUITECTÓNICO AVANZADO

Diseño Algorítmico para Juegos Infantiles: Sistemas de Crecimiento de Corales

Johanna Gabriela Garcés Villarreal

Línea de especialización (Operaciones)
director: Sergio del Castillo
Codirectora: Ana Medina
Correo electrónico: johannagabriela29@gmail.com

ES

RESUMEN. Este proyecto se centra en modelar el crecimiento de corales, Porque estos presentan patrones de crecimiento altamente eficientes y adaptativos que pueden inspirar nuevas formas de interacción y aprendizaje en entornos de juego ya caducos. Mediante el desarrollo de algoritmos basados en técnicas de crecimiento generativo, diferencial y autómatas celulares. Como resultado los niños interactúan con un tablero inspirado en el "Juego de la Vida" de Conway, donde los puntos vivos se mojan, creando una experiencia educativa y lúdica que visualiza algoritmos y fomenta la participación.

PALABRAS CLAVE: ALGORITMO, DIFERENCIAL, GENERATIVO, AUTOMATA CELULAR, CONWAY, JUEGO INFANTIL.

EN

ABSTRACT. This project focuses on modelling the growth of corals, because they exhibit highly efficient and adaptive growth patterns that can inspire new forms of interaction and learning in outdated game environments. By developing algorithms based on generative, differential and cellular automata growth techniques. As a result, children interact with a board inspired by Conway's "Game of Life", where live dots get wet, creating an educational and playful experience that visualizes algorithms and encourages participation.

KEYWORDS: ALGORITHM, DIFFERENTIAL, GENERATIVE, CELLULAR AUTOMATA, CONWAY, CHILDREN'S PLAY.

MADAA 2022-2024

MÁSTER EN DISEÑO ARQUITECTÓNICO AVANZADO

Diseño Algorítmico para Juegos Infantiles: Sistemas de Crecimiento de Corales

Johanna Gabriela Garcés Villarreal

Sergio del Castillo

FECHA: 31 de mayo del 2024
FIRMA DEL DIRECTOR/A DE LA TESIS DE FIN DE MÁSTER

Agradecimiento

Quiero expresar mi más profundo agradecimiento a todas las personas que han sido fundamentales en este camino.

Agradezco a Dios por brindarme la fortaleza y la guía necesarias para seguir adelante. Sin Su apoyo, nada de esto habría sido posible.

A mis padres, por su amor incondicional, sus consejos sabios y por haberme enseñado el valor del esfuerzo y la dedicación. Ustedes han sido mi inspiración y mi motivación constante.

A mi esposo, Oscar, por su paciencia, comprensión y amor inagotable. Gracias por apoyarme en cada paso de este camino, por estar siempre a mi lado y por creer en mí incluso en los momentos más difíciles.

Y a mis hijos, Bianca y Pablito, quienes son la luz de mi vida. A ustedes les he quitado tiempo, pero todo este esfuerzo y dedicación es por y para ustedes. Mi mayor anhelo es que puedan ver en mí un ejemplo de perseverancia y dedicación, y que sepan que todo lo que hago es con el fin de asegurarles un futuro mejor.

Gracias a todos por su amor, su apoyo y por ser mi razón de seguir adelante. Este logro es de todos nosotros.

Con todo mi cariño y gratitud.

INDICE

Contenido

CAPITULO 1

ENFOQUE ANALITICO

1 ENFOQUE ANALITICO

1.1 Introducción

´´Los parques infantiles se convierten en señales de herramientas para la imaginación ¨

Aldo van Eyck

Este proyecto se centra en modelar el crecimiento de corales utilizando diversas técnicas algorítmicas, como el crecimiento generativo, el diferencial y los autómatas celulares, con el objetivo de replicar los procesos naturales de crecimiento y simbiosis. La razón principal radica en innovar en el diseño de estructuras educativas y lúdicas para niños, considerando el creciente problema de aprendizaje asociado al uso excesivo de dispositivos móviles en la infancia. Actualmente, los niños están cada vez más ocupados con dispositivos electrónicos, lo que puede afectar negativamente su desarrollo cognitivo y social. Es crucial reconocer la importancia del juego en la primera infancia, ya que no solo proporciona entretenimiento, sino que también facilita el aprendizaje, promueve la resolución de problemas y ayuda a superar traumas.

Se lleva a cabo la identificación y análisis de los detalles del sistema de crecimiento coralino, seguido del desarrollo de algoritmos precisos que replican estos patrones. Estos algoritmos se aplican en el diseño de una fuente interactiva con agua, específicamente diseñada para niños, proporcionando una experiencia educativa y lúdica que busca contrarrestar los efectos negativos del uso excesivo de tecnología. Ofreciendo una experiencia de aprendizaje inmersiva y entretenida. En resumen, este proyecto fusiona la ciencia, la tecnología y la educación para inspirar y educar a las futuras generaciones sobre la importancia del juego en la primera infancia, utilizando como herramienta los sistemas de crecimiento en la arquitectura infantil.

1.2 Justificación y Pertinencia

Los espacios de juego infantil ofrecen a los arquitectos una plataforma para contribuir de manera significativa a la sociedad. Según Unicef, en Ecuador el 33% de los niños menores de cinco años asisten a centros de cuidado infantil, donde los patios de recreo están limitados a áreas de césped, Al mismo tiempo, El Instituto Nacional de Estadísticas y Censos (INEC), revelo que el 51.3 % de la población mayor a cinco años y más, por lo menos tenia un celular activado, los dispositivos electrónicos afectan negativamente su desarrollo cognitivo y social. Es crucial reconocer la importancia del juego en la primera infancia ya que no solo proporciona entretenimiento, sino que también facilita el aprendizaje y promueve la resolución de problemas y ayuda a superar traumas.

Estudiar a Aldo van Eyck es crucial porque, tras la Segunda Guerra Mundial, él logró utilizar el diseño de parques infantiles como una herramienta para ayudar a los niños a superar los traumas de la guerra. Van Eyck, a través de su trabajo, demostró cómo la adversidad puede ser enfrentada mediante la creación de espacios que permiten a la infancia florecer en medio de la inocencia y el juego.

El arquitecto holandés Aldo van Eyck (1918-1999) planteó en su manifiesto "El niño, la ciudad y el artista" la crisis de la ciudad moderna, caracterizada por su impersonalidad y la falta de espacios adecuados para los niños (Van Eyck, 1962). Aunque no ofreció especificaciones formales sobre el diseño arquitectónico de estos espacios, sus numerosos playgrounds en Ámsterdam, diseñados en colaboración con Jacoba Mulder (1900-1988), pueden ser considerados como expresiones espaciales de su filosofía (Van Eyck, 1962). Van Eyck señaló que reflexionar sobre la infancia como metáfora de la capacidad humana de imaginación podría ayudar a la ciudad moderna a recuperar una identidad más humana (Van Eyck, 1962).

En el X Congreso de la Organización Mundial para la Educación Preescolar en 1964, auspiciado por la Unesco, el arquitecto y urbanista danés Steen Eiler Rasmussen (1898-1990) destacó la responsabilidad de los planificadores urbanos en satisfacer las necesidades de los niños y jóvenes, quienes carecen de representación democrática (Rasmussen, 1964). Rasmussen enfatizó que los niños en edad escolar necesitan más que simples patios de recreo o áreas deportivas; desean aventura, creatividad y la oportunidad de producir por sí mismos (Rasmussen, 1964).

La biomimética, que busca soluciones sostenibles inspiradas en la naturaleza, ofrece un enfoque prometedor para el diseño de juegos infantiles. Al observar cómo la naturaleza ha optimizado estructuras y funciones, los diseñadores pueden crear espacios que promuevan el bienestar y la armonía con el entorno (Benyus, 1997). Este enfoque permite a los niños sumergirse en la riqueza sensorial de la naturaleza, estimulando su curiosidad y fomentando el aprendizaje (Benyus, 1997).

Considerando estas perspectivas, esta investigación propone el diseño de un juego infantil basado en los principios de biomimética, específicamente inspirado en los sistemas de crecimiento de los corales. Este enfoque busca proporcionar a los niños un contacto más íntimo con las formas y procesos naturales, enriqueciendo su experiencia de juego y fomentando una conexión más profunda con la naturaleza.

1.3 Objetivo General

En este proyecto de investigación lo que busca es inspirar el desarrollo de algoritmos avanzados mediante la analogía del crecimiento de corales, replicando los procesos naturales de crecimiento y simbiosis, para innovar en el diseño de estructuras educativas y lúdicas para niños. que estimule la aventura, creatividad e imaginación.

Objetivo Especifico

1.- Crear y perfeccionar algoritmos inspirados en los procesos naturales de crecimiento y simbiosis de los corales, con el fin de replicar estos procesos en entornos computacionales

2.- Aplicar los algoritmos desarrollados para diseñar estructuras que sean educativas y lúdicas, enfocadas en el uso por parte de niños.

3.- Diseñar y construir estructuras y actividades que permitan a los niños interactuar directamente con formas y patrones inspirados en la naturaleza, para aumentar su conexión y comprensión del mundo natural.

1.4 Hipótesis

La representación mediante diseño algorítmico es valiosa porque describe un proceso en lugar de un único resultado, permitiendo identificar los pasos y variables involucrados. Esto posibilita realizar variaciones y obtener múltiples resultados. La construcción del algoritmo es en sí un proceso de diseño, donde se decide cómo navegar dentro de un cuerpo determinado. Esta investigación nos lleva a la pregunta.

¿Cómo pueden aplicarse los sistemas de crecimiento a la arquitectura para ofrecer soluciones de diseño en juegos infantiles? La respuesta se desarrollará a lo largo de la tesis, basada en la noción del diseño algorítmico**. Este concepto lleva a la hipótesis de que el diseño algorítmico será el método para visibilizar cómo funcionan las fuerzas en el sistema de crecimiento.**

1.5 Metodología

La metodología llevada a cabo en esta investigación se centra en la aplicación de sistemas de crecimiento para el diseño de juegos infantiles. Se seleccionaron tres tipos de sistemas de crecimiento: generativo, diferencial y autómatas celulares. Asimismo, se eligieron tres tipos de corales para estudiar sus patrones de crecimiento y aplicar estos principios en el desarrollo de algoritmos.

Para la implementación y testeo de los algoritmos, se decidió utilizar software especializado en diseño paramétrico. Aunque existen varias opciones en el mercado, como Cinema 4D y Blender, se optó por utilizar Rhino y Grasshopper debido a su robustez y flexibilidad para el diseño algorítmico. Esta selección permite una integración eficiente de los algoritmos y facilita la visualización y manipulación de los patrones de crecimiento en los modelos diseñados.

CAPITULO 2

ESTADO DEL ARTE

2 ESTADO DEL ARTE

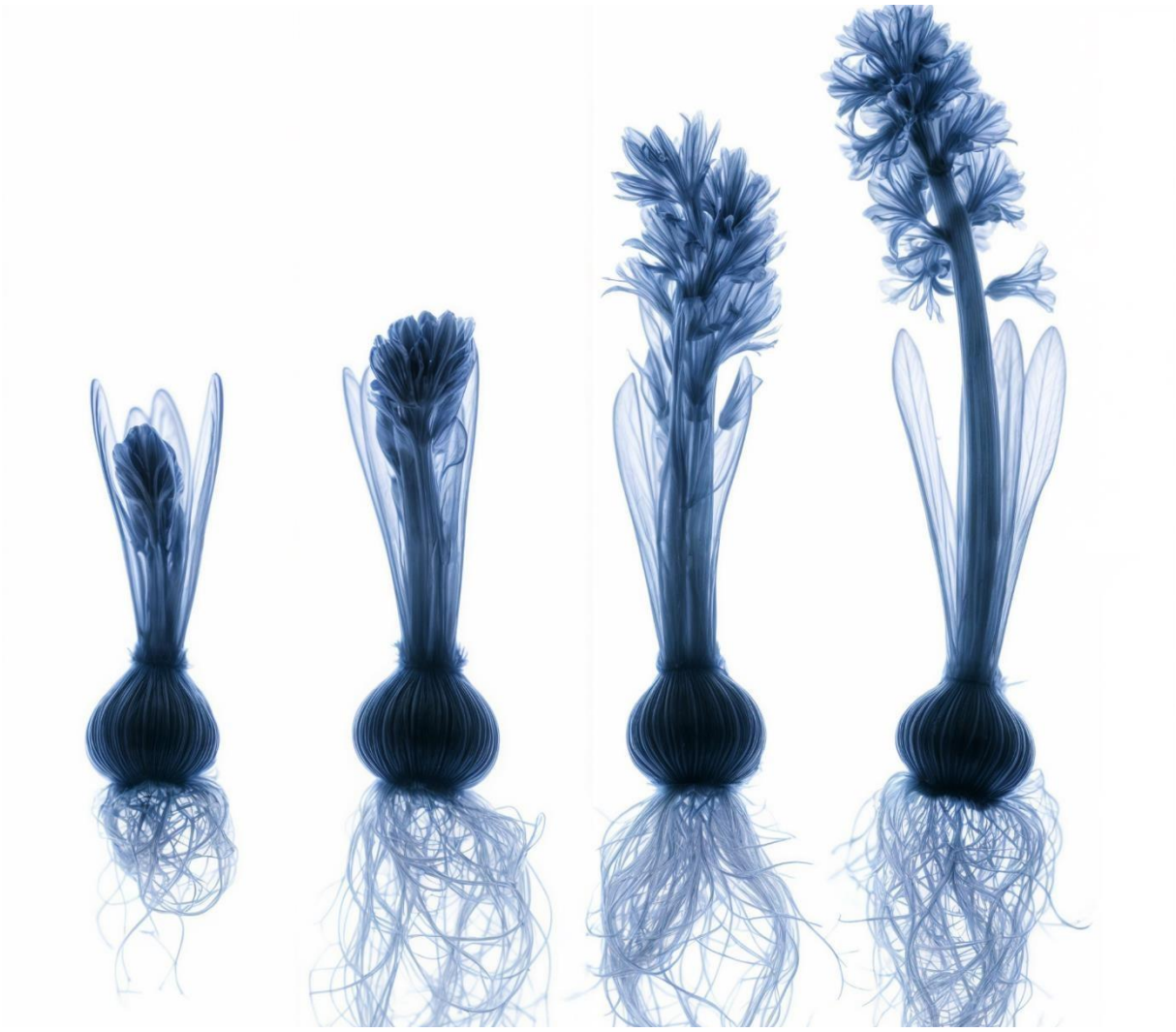


Fig.1 Radiografía coloreada de flores de jacinto en distintas fases de crecimiento. Fuente: Michael Hensel

2.1 Sistemas de Crecimiento

Se define como crecimiento al aumento irreversible de tamaño en un organismo, como consecuencia de la proliferación celular, misma que conduce al desarrollo de estructuras más especializadas del organismo, comenzando por las propias células y, pasando por tejidos, hasta llegar a órganos y sistemas.

D'Arcy Wentworth Thompson, en su obra On Growth and Form (Sobre el crecimiento y la forma), presenta una perspectiva única sobre el desarrollo y la evolución en la naturaleza. Thompson (1917) argumenta que los fenómenos biológicos, como el crecimiento de organismos y la formación de estructuras, están influenciados por principios matemáticos y físicos subyacentes. Thompson examina cómo las fuerzas físicas, como la tensión y la compresión, así como los patrones matemáticos, como las espirales y las formas geométricas, influyen en la forma y el crecimiento de los organismos vivos. Propone que muchas formas naturales, desde conchas de caracoles hasta la estructura de las alas de los insectos, pueden entenderse mejor mediante el estudio de los procesos de crecimiento y las fuerzas físicas involucradas en su desarrollo. En resumen, Thompson sugiere que los sistemas de crecimiento en la naturaleza son el resultado de una interacción compleja entre factores físicos, matemáticos y biológicos, y que comprender estos sistemas puede arrojar luz sobre la diversidad de formas y estructuras que observamos en el mundo natural. (Fig.1)

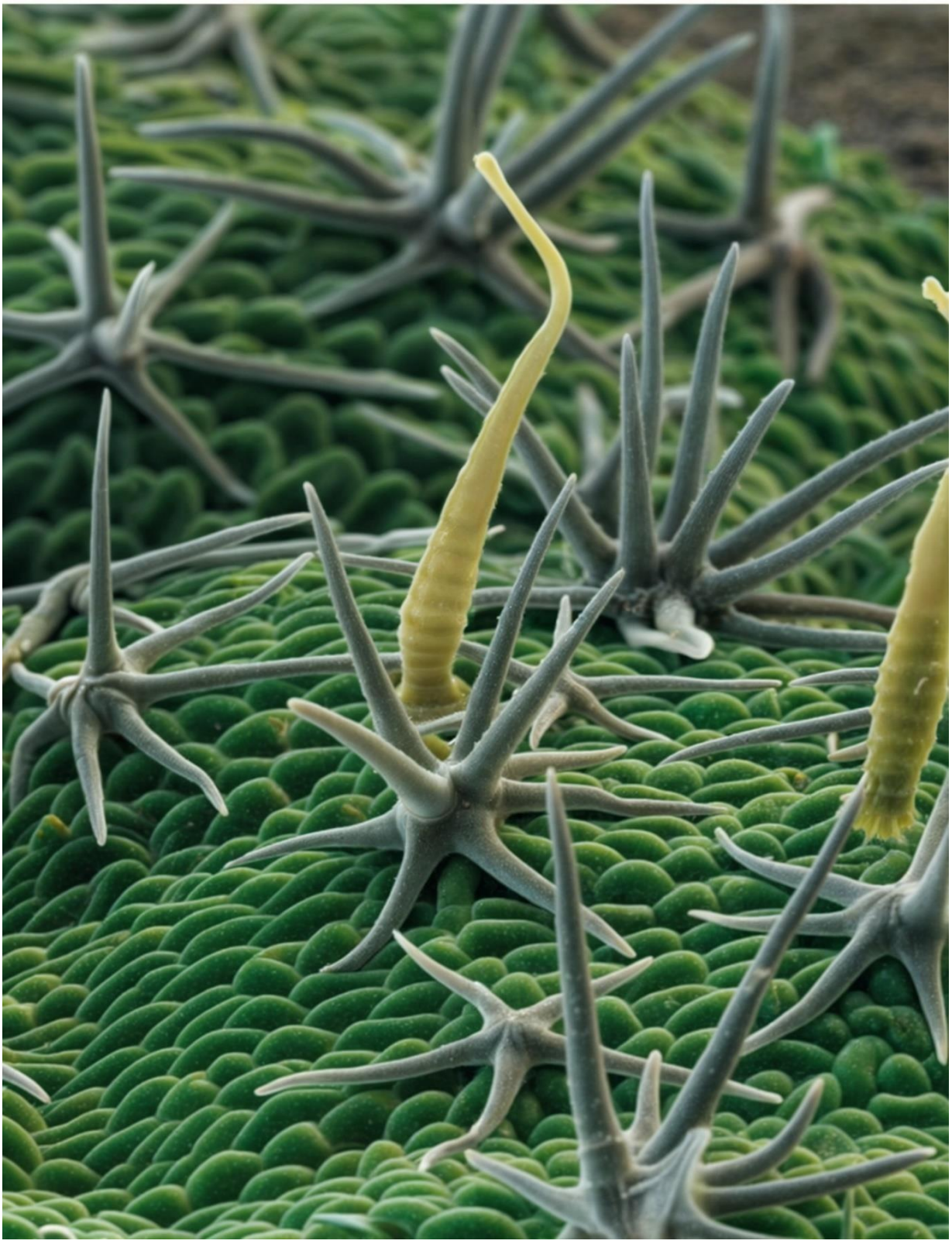


Fig. 2 Micrografía electrónica de barrido coloreado de la superficie foliar. Diferenciada y los pelos foliares de la jara (Citus Longifolius). Estos tricomas protegen la hoja contra el ataque de plagas. Fuente: Michael Hensel

2.2 Sistema de Crecimiento Generativo.

Los sistemas de crecimiento generativo son métodos utilizados para modelar y simular el crecimiento y desarrollo de estructuras complejas, inspirados en procesos naturales. Estos sistemas encuentran aplicaciones en campos como la biología, el diseño arquitectónico, el urbanismo y la generación de gráficos por computadora. Michael Hensel (2005)

2.2.1 Autoorganización

La autoorganización es un proceso en el que la organización interna de un sistema se adapta al entorno para promover una función específica sin ser guiada o gestionada desde el exterior. En biología esto incluye los procesos que conciernen la biología del desarrollo, que es el estudio del crecimiento y el desarrollo de los organismos y comprende el control genético del crecimiento. Michael Hensel (2005)

Biólogos y científicos computacionales han colaborado esta tarea con resultados muy interesantes. Es posible desarrollar digitalmente plantas que (crecen) en función de las condiciones ambientales. Cada cambio en la entrada produce un resultado de crecimiento diferente. En otras palabras, una articulación diferente de la especie modelada. Esto se domina modelización de crecimiento sensible al medio ambiente.

En 1968, el biólogo húngaro Aristid Lindenmayer investigo patrones de crecimiento de distintos organismos multicelulares múltiples. Ese mismo año empezó a desarrollar una descripción formal del desarrollo de estos organismos simples. Denominada sistema Lindenmayer o sistema L. Un sistema L en lo que informática se denomina una gramática formal, una estructura abstracta que describe un lenguaje formal mediante secuencia de varios objetos simples conocidos como cadenas.

Prusinkiewicz[1] destaca varias ventajas del uso de modelos computacionales en el contexto del desarrollo arquitectónico. En primer lugar, estos modelos pueden proporcionar una comprensión cuantitativa de los mecanismos de desarrollo. En segundo lugar, tienen el potencial de sintetizar la interacción entre los diversos aspectos del desarrollo. Esta capacidad puede enriquecer el diseño arquitectónico al ofrecer una nueva sensibilidad analítica y generativa, permitiendo una mejor comprensión de las sinergias entre sistemas y entornos.
Estos modelos arquitectónicos pueden utilizarse para la modelización de plantas enteras, o grupos de plantas, potencialmente integradas en una ecología determinada, o de diversas partes y subescalas de una sola planta.

[1] Przemyslaw Prusinkiewicz es un investigador y científico conocido por su trabajo en la biología matemática, gráficos por computadora y modelado de plantas

En cuanto a la modelización de plantas individuales, una de las características mas llamativas es la integración de la biomecánica en el desarrollo vegetal, que permite informar el crecimiento de la planta con aportaciones físicas, biológicas y ambientales extrínsecas. Los modelos avanzados incorporan el impacto combinado de la gravedad, el tropismo, el contacto entre los distintos elementos de una estructura vegetal y el contacto con obstáculos.

La configuración metodológica, el conjunto de herramientas y la elección de variables determinantes son igualmente interesantes para el diseño arquitectónico. De este modo, sistemas envolventes de edificios entero podría basarse en datos multivariables y optimizarse para satisfacer objetivos múltiples prestaciones.

La entrada de gravedad puede informar el comportamiento estructural que luego se negocia con la exposición a la entrada ambiental, por ejemplo, para recoger la energía solar, el agua, la lluvia etc. En lugar de una optimización paso a paso, objetivo a objetivo, al final del proceso de diseño. Llevada a cabo por especialistas que non fundamentales en el proceso de diseño, la respuesta a estímulos extrínsecos puede formar parte ahora del proceso generativo de la arquitectura.

El software desarrollado por el equipo Calgary[2] permite modelizar diversas características de plantas. Por ejemplo, la filotaxis en espiral, que sirve para optimizar el empaquetamiento de semillas escamas, o la orientación de exposición de las hojas a los factores ambientales. Como la luz. (Fig.3)

Es sorprendente como unas de las características tan diminutas de las plantas consiguen modular fuerzas muy fuertes de tal manera, que la planta puede sobrevivir condiciones bastantes extremas. La lección para los arquitectos es que estos elementos y sus funciones no son escalables.
Hay 2 maneras para aprovechar la lección aprendida por la naturaleza, en primer lugar, produciendo elementos del mismo tamaño para lograr el rendimiento observado en la naturaleza. En segundo lugar, determinando los tamaños apropiados de otros elementos para modular las condiciones elegidas en los rangos requeridos.

El equipo Calgary lo consigue generando esqueletos de planta e interpretando gráficamente como cilindros generalizados. A continuación, se especifican las propiedades del pelo, se mapea la superficie, las propiedades del pelo, se mapea sobre la superficie y se ajusta según la información posicional.

[2] departamento de informática en la Universidad de Calgary (Alberta, Canadá)

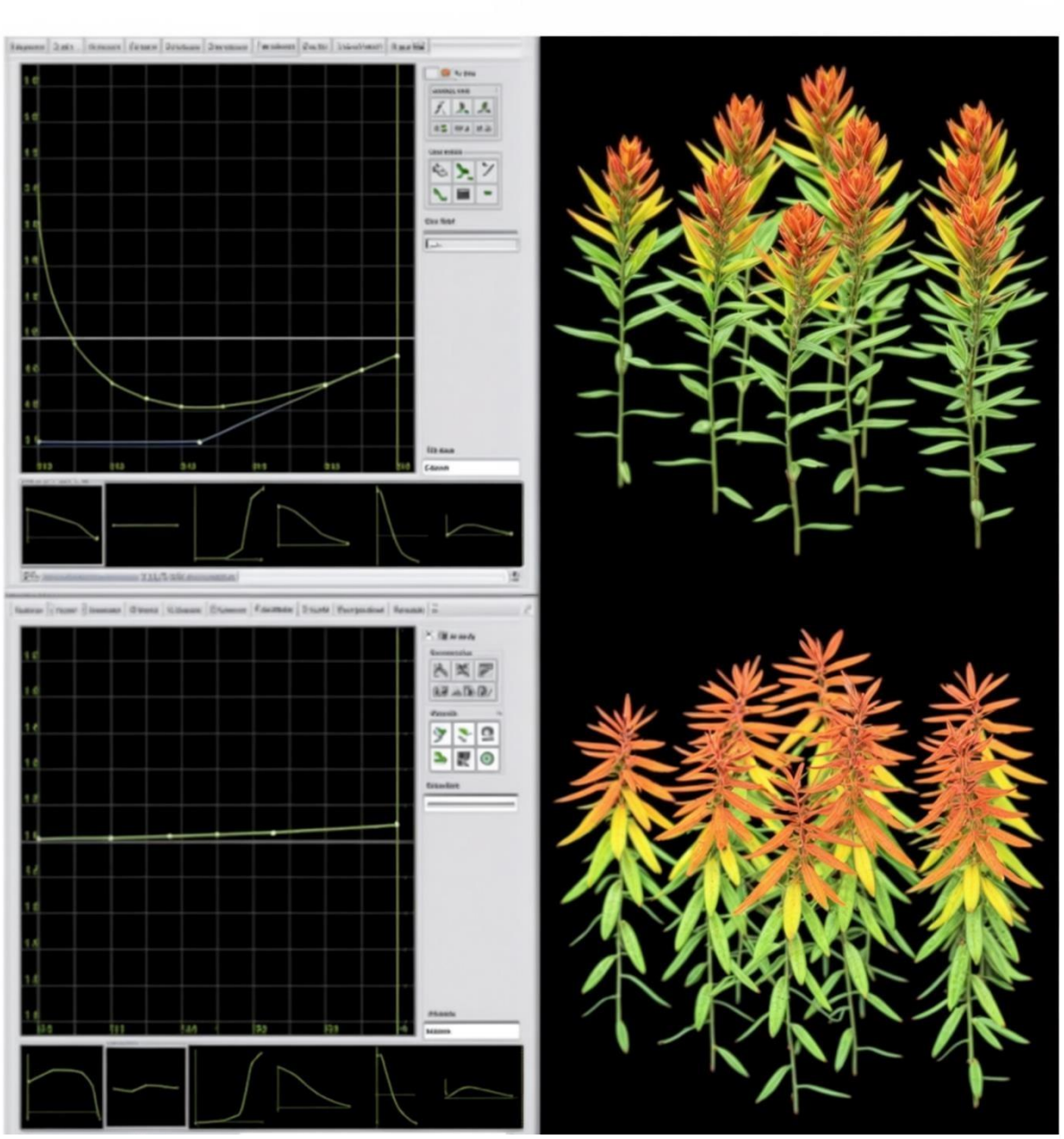


Fig. 3 L-studio es un software para Windows que permite crear modelos de simulación y realizar experimentos virtuales utilizando L-systems.Fuente; Michael Hensel

Resulta interesantes que programas como 3D studio o Cinema 4D hayan incorporado recientemente la simulación de pelo en relación con el flujo de aire, de forma que se puede determinar propiedades específicas de pelo y de flujo de aire, estas herramientas pueden ser de gran valor en el diseño arquitectónico, De este modo el análisis y generación de diseño pueden sintetizar modelos de procesos crecimiento sensibles al sistema.
La ecología es el estudio de la relación de los organismos con su entorno se puede estudiar en varios niveles, desde el organismo individual hasta las poblaciones de comunidades de especies, los ecosistemas por última biosfera.

A nivel de organismo individual. – el comportamiento es una acción o respuesta observada en un organismo, especie o factores ambientales. Esto implica en primer lugar un estímulo, es decir un agente interno o externo que produce una reacción o cambio de organismo en segundo lugar la sensibilidad que es la capacidad de percibir un estímulo en tercer lugar la irritabilidad es la capacidad que tiene un organismo de responder a estímulos.

A nivel de poblaciones. - en otras palabras, son organismos que constituyen un grupo específico y se dan en un hábitat concreto, la ecología de las poblaciones se ocupa en la dinámica de las poblaciones dentro de las especies e interacción de las poblaciones con factores ambientales.
La ecología de comunidades son especies que interactúan entre sí, en una región específica en condiciones ambientales relativamente similares.

2.2.2 Mushtari Exploración del Crecimiento Generativo

La ecología es el estudio de la relación de los organismos con su entorno se puede estudiar en varios niveles, desde el organismo individual hasta las poblaciones de comunidades de especies, los ecosistemas por última biosfera. Mushtari, es una creación de Neri Oxman, es una estructura transparente impresa en 3D que trasciende los límites del arte, la ciencia y el diseño.

El Objetivo de Mushtari era de crear una prenda de vestir a escala real con canales de fluidos. Para el diseño implico un proceso computacional que emula el crecimiento natural, pero de forma controlada. Esta condujo al desarrollo de estructuras naturales y orgánicas, dando lugar a una ¨simbiosos ¨ de diseño entre el crecimiento artificial (computacional) del entorno y el crecimiento natural (biológico) de los microorganismos contenidos.

Mushtari se diseñó utilizando un algoritmo de crecimiento generativo que emula el crecimiento biológico. Mediante el desarrollo de geometrías complejas en múltiples interacciones.

Procesos similares son, por ejemplo, los de Lindemayer Systems. (L-Systems) las reglas se aplican en paralelo para generar estructuras ramificadas, otro ejemplo, son las ecuaciones de reacción – difusión, en las que se aplica ecuaciones diferenciales parciales para modelar la distribución de componentes dentro de una reacción, imitando fenómenos de formación de patrones que se caracteriza por la variación de propiedades, también, Mushtari propone el uso del sistema en masa-resorte sobre una malla con una estrategia de refinamiento, selectivo, para la generación de formas celulares.

Aplicando un sistema similar, el planteamiento permite informar sobre la geometría general, la geometría local de la malla y las variaciones en la distribución de propiedades de los materiales a lo largo de la escala de longitud, alterando fuerza relativa de relajación, atracción y repulsión a lo largo de varias interacciones. Este enfoque permitió diseñar un único canal que ´´crece¨ a lo largo de numeraciones iteraciones para generar un wearable con 58 metros de estructura interna hueca. El canal presenta variaciones de diámetro (de 1 a 25mm) . (F

En la (Fig. 4, 5,). Se muestra el marco general de la generación de Mushtari. Una representación geométrica de entrada se transforma primero en una representación intermedia. A partir de esta representación intermedia, se genera una representación implícita gruesa. A continuación, se utiliza la información de tres representaciones para deformar la representación geométrica inicial. Por último, la representación dada se deforma y se refina continuamente. A medida que el proceso se repite, las deformaciones se agregan en el crecimiento de una forma coherente.

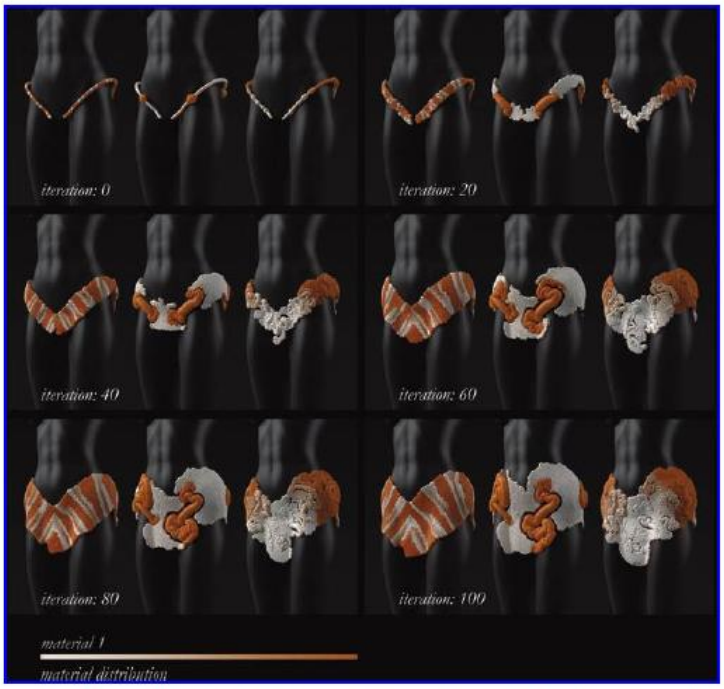


FIG.11 Visualización de 100 pasos de iteración de tres variaciones de crecimiento diferentes utilizadas para generar el wearable fluídico *Mushtari* (orientaciones *izquierda*, *central* y *derecha* en cada panel). Se muestran los controles a escala global y local sobre la generación de los canales fluídicos, así como la capacidad de ajustar las propiedades de los materiales durante el crecimiento computacional. Los materiales en *rojo representan zonas* transparentes, mientras que los materiales en *blanco* representan zonas opacas. El enfoque *situado más a la izquierda* (en cada panel) varía la transparencia como una función periódica de la longitud del filamento. El enfoque del centro varía la transparencia en función del diámetro del tubo. En el enfoque del extremo *derecho*, las propiedades del material se asignan por regiones. Imágenes en color disponibles en línea en www.liebertpub.com/3dp

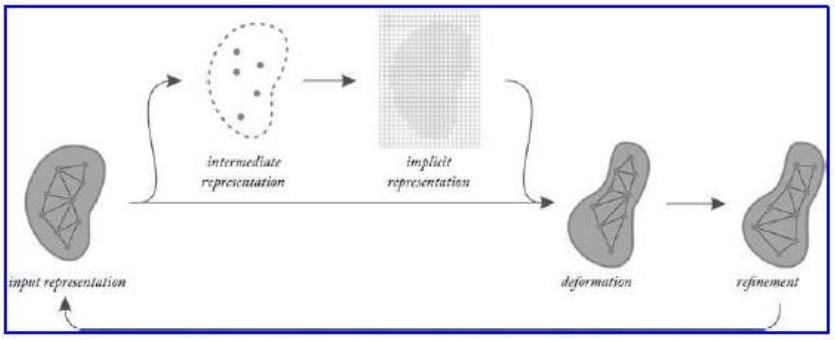


FIG.12 Esquema del marco general utilizado para la generación de estructuras cultivadas computacionalmente.

Fig. 4 Visualización de crecimientos diferentes
wearable fluido

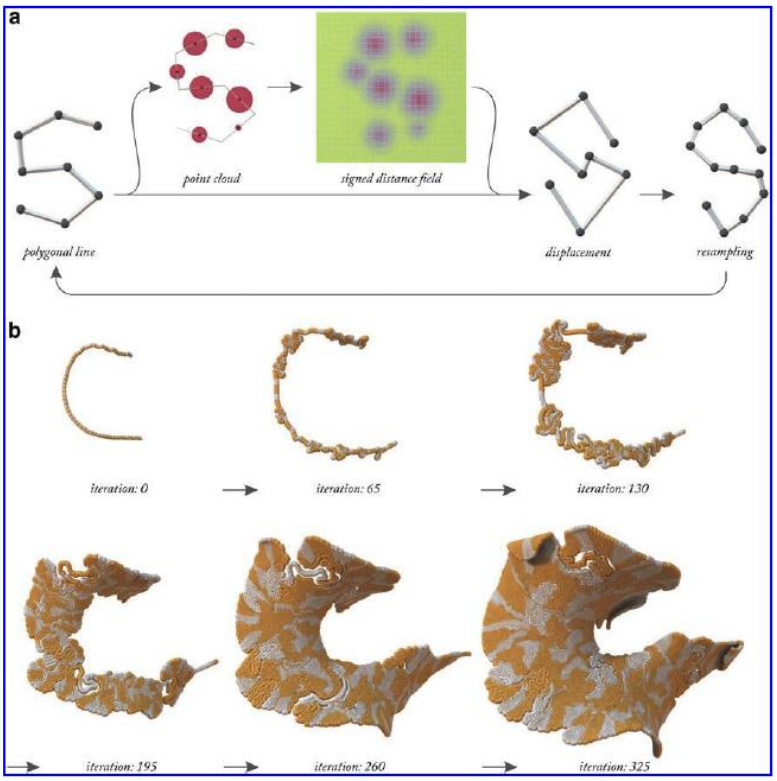


FIG.13. (a) Ejemplo del marco general aplicado a una *línea poligonal*. (b) Varias iteraciones de este proceso dan como resultado final una arquitectura en espiral compacta. En esta figura, el proceso se limita al espacio tangente de una malla triangular guía subyacente (no mostrada). Imágenes en color disponibles en línea en www.liebertpub.com/3dp

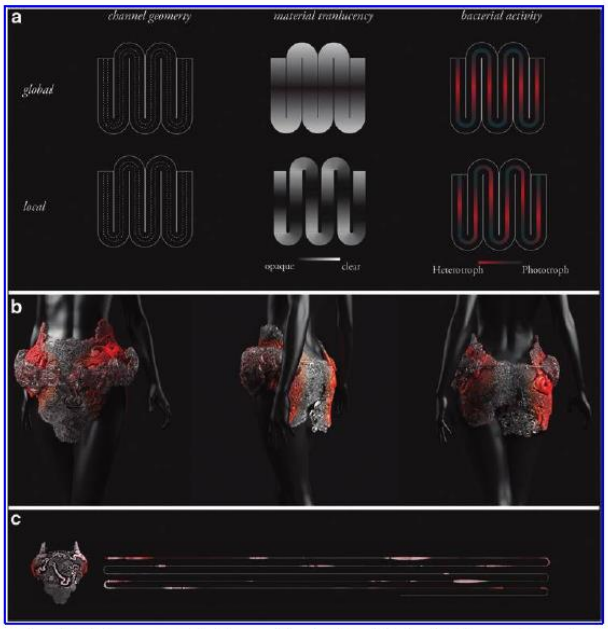


FIG.14 (a) Efecto deseado de la transparencia del material sobre la actividad microbiana. En las regiones translúcidas, el cocultivo recibe luz, aumentando la actividad fototrófica (*columna derecha*). En las regiones opacas, el cocultivo está a oscuras y los heterótrofos consumirían los fotosintatos, aumentando así su actividad metabólica. La transparencia puede controlarse globalmente, como se muestra en la *columna superior central*, independientemente del filamento, o localmente, como se muestra en la *columna inferior central*, utilizando la parametrización de la longitud del camino como guía para las distribuciones de material. (b) Implementación del enfoque de modelado heterogéneo global en el wearable. (c) Visualización de la hebra desplegada que muestra cómo el enfoque global basado en fuentes influye en los cambios locales de opacidad a lo largo de la hebra. Imágenes en color disponibles en línea en www.liebertpub.com/3dp

Fig. 5 Ejemplo del marco general de una línea poligonal

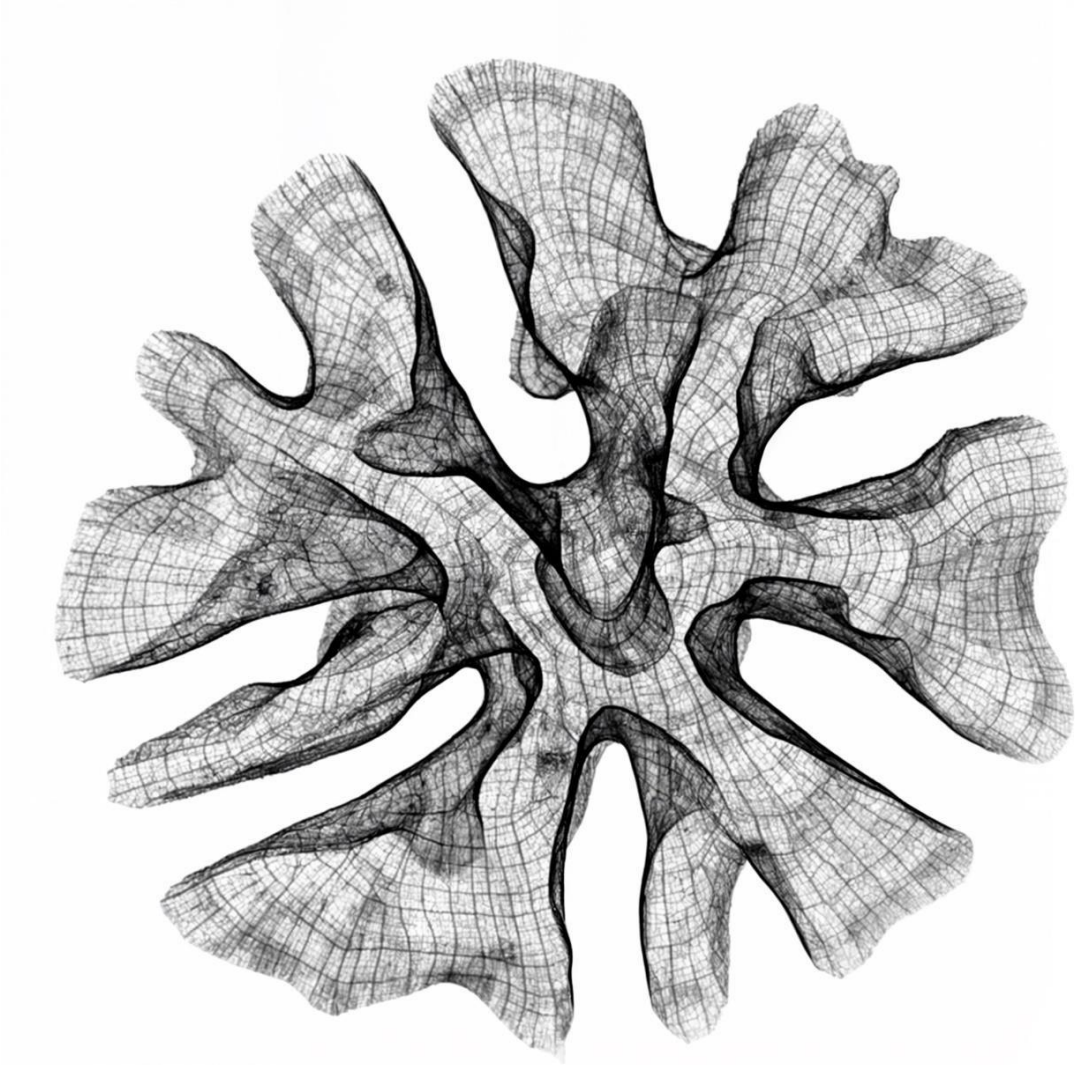


Fig. 7 Crecimiento Diferencial. Fuente: Anders Hoff

2.3 Sistema Crecimiento Diferencial

En la naturaleza, una amplia gama de formas y estructuras surge de la interacción entre diversos procesos químicos y mecánicos que actúan sobre los materiales. Estos procesos, a menudo gobernados por reglas simples, dan lugar a una diversidad sorprendente de formas. Un ejemplo de esto es el fenómeno del crecimiento diferencial, el cual genera formas sinuosas y ondulantes que evocan imágenes de ríos serpenteantes, la textura ondulada de las superficies de frutas y semillas, los patrones foliosos de los líquenes, así como los movimientos de llenado de espacio de organismos como gusanos y serpientes, e incluso, se especula, la organización del cerebro (Hoff, 2022) (Fig.5).

2.3.1 Crecimiento

El crecimiento diferencial es un mecanismo mediante el cual los sistemas biológicos adquieren su forma. Al regular la velocidad de crecimiento de distintas partes de una superficie, se puede determinar su forma final[3]. (Anders Hoff, 2022) (Fig. 4) (Fig.6)

En este caso, nos enfocaremos en líneas en lugar de superficies, aunque el concepto básico es el mismo. Ejemplos inspiradores de este tipo de crecimiento se pueden observar en nueces, pétalos de flores y las capas de repollo.

Aunque el crecimiento diferencial en la naturaleza es un proceso complejo y multifacético, es posible simularlo con solo algunos factores clave:

Nodos y bordes: los nodos están conectados a un determinado Número de nodos vecinos a través de las aristas.

Atracción: los nodos conectados tratarán de mantener un razonablemente cerca unos de otros. En la siguiente figura La atracción se produce entre los nodos conectados en el bucle.

Rechazo: los nodos intentarán evitar estar demasiado cerca a los nodos circundantes (dentro de una cierta distancia). Los fores de rechazo son indicado por líneas cian en la figura abajo.

[3] Hoff, A. (2024, Abril). *inconvergente*. https://incorvergent.net/2016/shepherding-random-growth/

Divisiones: Si una arista se alarga demasiado, un nuevo nodo inyectarse en el centro del borde.

Fig. 8 Línea del rio. Fuente: Anders Hoff, https://incorvergent.net/2016/shepherding-random-growth/

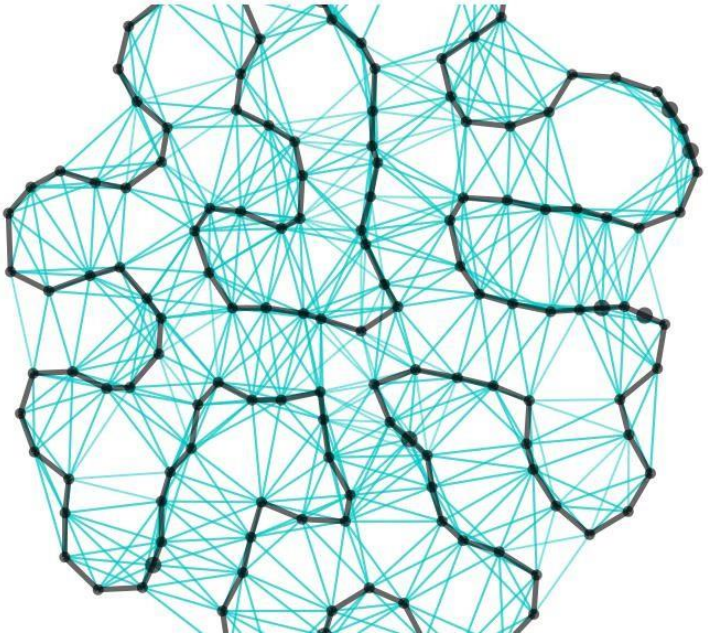


Figura 9. Crecimiento. Fuente: Anders Hoff

2.3.2 Uniforme

El tipo más simple de esquema de crecimiento es seleccionar aleatoriamente un borde, e Inserte un nuevo nodo en el medio. Para reducir el desorden, debe evitar Inserción de nodos si el borde es demasiado corto. Esto tenderá a conducir a Colisiones. (Fig. 7)
La intensidad con la que se insertan los nuevos nodos tendrá un gran impacto en la tasa de crecimiento, así como la forma, por lo que hay una serie de cosas que experimenta. (Anders Hoff, 2022)

2.3.3 Curvatura

Propone priorizar las partes de la estructura donde la curvatura de la superficie es mayor, indicada por círculos cian. Se utiliza una fórmula para estimar la curvatura en cada punto de la línea, comparando ángulos entre aristas vecinas. Luego, se propone muestrear las aristas para dividirlas, priorizando aquellas con mayor curvatura. Se genera un número aleatorio entre 0 y 1, y se divide el borde solo si es lo suficientemente largo y si el número aleatorio es menor que el valor de la curvatura, de esta forma, las mayores curvaturas se seleccionan con mayor frecuencia. Este proceso se repite en cada borde en cada iteración. (Fig. 8)
(Anders Hoff, 2022)

2.3.4 Nodos

Este proceso es bastante intensivo desde el punto de vista computacional. La más costosa operación es encontrar nodos que están en las proximidades de un nodo dado, por lo que se pueden calcular las fuerzas de rechazo. (Anders Hoff, 2022) (Fig. 9)

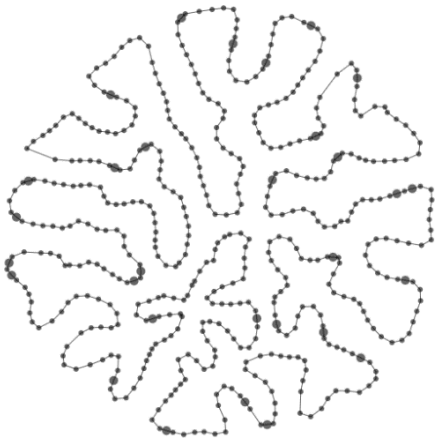


Fig. 10 Crecimiento Uniforme.
Fuente: Anders Hoff. 2024

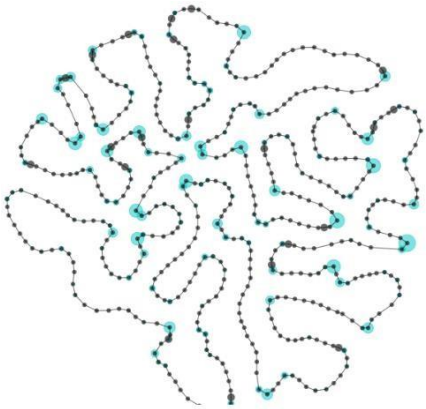


Fig. 11 Crecimiento Curvatura.
Fuente: Anders Hoff 2024

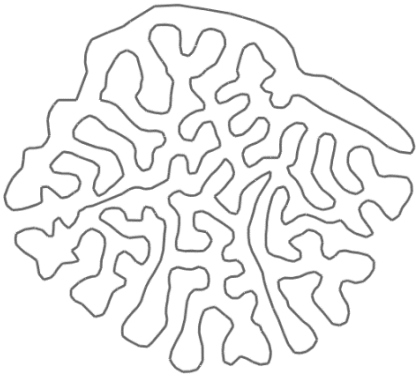


Fig. 12 Crecimiento por Nodos.
Fuente: Anders Hoff 2024

2.3.5 Floraform .- Exploración del crecimiento diferencial

Fig. 13 Floraform (joyería por crecimiento diferencial)

Floraform se inspira en la biomimetica del crecimiento de las hojas y las flores en flor y explora el desarrollo de las superficies a través del crecimiento diferencial. Utiliza este sistema para crear computacionalmente una nueva colección de joyas impresas en 3D

Floraform es una simulación de una superficie elástica de crecimiento diferencial para explorar cómo los sistemas biológicos crean forma variando las tasas de crecimiento a través del espacio y el tiempo. Comenzó con una flor inusual, Celosía cristata, a través diferenciación celular, geometría diferencial discreta, marinas cleptoplásticas, movimientos násticos y zoótropos del siglo XIX. El resultado es una serie de esculturas y joyas impresas en 3D generadas a través de un proceso computacional de crecimiento diferencial.

El crecimiento diferencial pasa que la célula se dividiera crecería uniformemente, daría lugar a una mancha arrugada. Sin embargo, a través de una subdivisión y diferenciación cuidadosamente coordinadas, los sistemas biológicos producen estructuras con formas y funciones específicas y reproducibles. El crecimiento no es uniforme sino diferencial.
En pocas palabras, algunas áreas crecen más que otras, y conduce a formas macroscópicas. Estas formas son el resultado de la interacción de los procesos de crecimiento celular subyacentes y la mecánica de los propios materiales
La flor llamada Creta de Gallo, esta presenta una forma ondulada se la conoce como flor cerebral.
El crecimiento de la cresta de gallo fue simulado con el crecimiento diferencial hacia el borde.

Mapeo del espacio de simulación

Basado en puntos

El área de crecimiento activo es un solo punto. Esto es similar a la punta de crecimiento del tallo en las plantas, llamada meristemo apical. Podemos permitir que este punto se divida en varios puntos a lo largo del tiempo. Cuando la zona de crecimiento se bifurca, los nuevos puntos se separan de forma natural, formando ramas. (Fig. 13).

Basado en líneas

El área de conocimiento activo comienza en un punto, pero en lugar de dividirse en dos puntos separados, estos puntos permanecen conectados formando una línea de expansión. Esto está inspirado en las plantas fascinadas, y los resultados se asemeja al cactus. (Fig. 14).

Basado en el borde

El área de crecimiento activo es el borde, forma de geometrías sinuosas e hiperbólicas (Fig. 15).

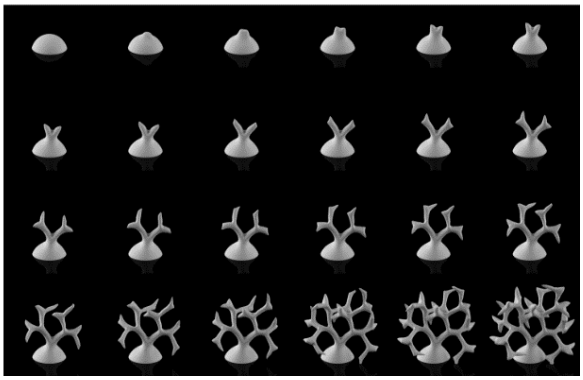


Fig. 13 Crecimiento Basado en Líneas

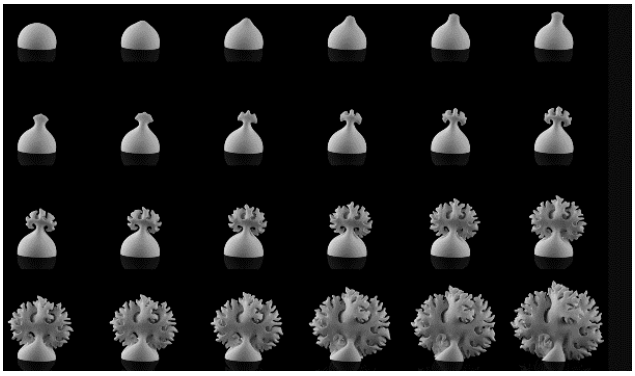


Fig. 14 Crecimiento Basado en Puntos

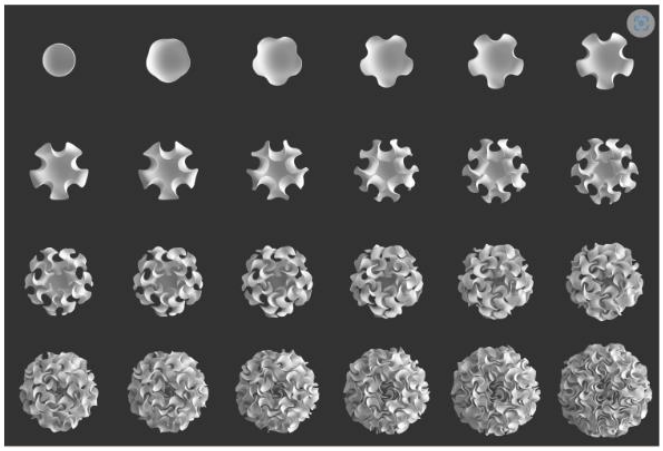


Fig. 15 Crecimiento Basado en el Borde

Física

Cada borde de la malla se modela como un resorte que resiste al estiramiento de la superficie. El borde entre dos caras también tiene energía y de flexión proporcional al ángulo entre las caras que resiste la flexión.

Detección de colisiones

Para evitar que nuestra superficie se interseque, necesita una forma de detectar colisiones entre partes de la superficie a medida que crece e introducir fuerzas para evitar que se superponga. Utilizamos un sistema de colisión basado en partículas. Cuando dos cuerpos de colisión cruzan, introducimos una fuerza rígida similar a un resorte que empuja uno del otro.

Distancia Geodésica

La tasa de crecimiento de la superficie está controlada por la distancia a lo largo de la superficie desde el borde o alguna otra zona de crecimiento especificada. A continuación, expandimos la longitud de las aristas en función de esta distancia. La distancia a lo largo de una superficie se conoce como distancia geodésica

Subdivisión Adaptiva

A medida que nuestra superficie crece, necesitamos mantener el nivel de detalle de nuestra malla para tener una superficie lisa y una estimulación estable. Cualquier arista que crezca por encima de una longitud de umbral se subdivide de tal manera que cada triángulo vecino se divide en dos. Para mantener una buena forma de triángulo, una arista solo se divide si es la arista más larga de un triángulo. Además, introducimos el volteo de bordes para equilibrar la topología de la malla a medida que se subdivide.

Fig. 16 Crecimiento Autómata Celular. Fuente: Articulo Discrete vs. Discretized Growth

2.4 Sistemas de Crecimiento Autómata Celular

Autómatas Celulares son modelos matemáticos para sistemas dinámicos que evolucionan en el espacio y el tiempo. Consisten en una regularización espacial de células, cada una de las cuales puede estar en uno de un número finito de estados. El estado de cada célula cambia en función de un conjunto de reglas que dependen del estado de las células vecinas. Este concepto fue introducido por John von Neumann[4] y Stanislaw Ulam en la década de 1940.

Características de los Autómatas Celulares:
Rejilla Regular: El espacio se divide en una rejilla regular de células.
Estados Finitos: Cada célula puede tomar un número finito de estados.
Reglas de Transición: Las reglas determinan cómo cambia el estado de una célula basado en los estados de sus vecinas.
Evolución Discreta: El tiempo avanza en pasos discretos, y en cada paso, todas las células se actualizan simultáneamente según las reglas.

Para simular procesos de crecimiento se ha utilizado diferentes lógicas matemáticas basadas en cuadriculas. Hoy en día, suele calcularse de forma libre en el espacio tridimensional, pero en un principio se concibió para trabajar cuadriculas bidimensionales (Written y Sander 1981)

Sarkar (2000) explica que el Autómata Celular como herramienta de diseño generativo está bien documentado desde que von Neuman lo introdujo en 1963. Especialmente el Juego de la Vida de Conway, un tipo de AC, se ha utilizado para simular procesos de crecimiento (Wolfram 1983, Gardner 1970) Se ha hecho diferentes intentos de utilizarlos para la arquitectura y el diseño urbano (Al-Qattan, Yan y Galanter 2017, Shiffman 2012, Adilenidou 2015, Kuo y Kausinger 2010).

Aunque este proceso puede lugar a estructuras geométricas complejas derivadas de una geometría simple, Herry Ford (2007) afirma que la transición de un Algoritmo de AC genérico a una herramienta de diseño especifica no se conoce muy bien. Sostiene que los sistemas AC Tienen que someterse en la mayoría de los casos múltiples adaptaciones para producir resultados útiles en el campo del diseño arquitectónico.

[4] John von Neumann[1] hizo contribuciones significativas en varias áreas de las matemáticas, la física, la informática y la economía. Es conocido por su trabajo en la teoría de juegos, la teoría cuántica, la informática y el desarrollo de la arquitectura de von Neumann para las computadoras.

2.4.1 El juego de la Vida

El Juego de la Vida es un autómata celular creado por el matemático británico John Horton Conway en 1970. Es un juego sin jugadores, donde la evolución depende únicamente del estado inicial y no requiere intervención posterior.
Desde su creación, ha generado gran interés por la variabilidad en la evolución de sus patrones. El Juego de la Vida es un excelente ejemplo de emergencia y autoorganización, y resulta fascinante para científicos, matemáticos, economistas y otros estudiosos observar cómo reglas muy simples pueden dar lugar a patrones complejos.

2.4.2 El juego

El Juego de la Vida es un juego sin jugadores, lo que implica que su progresión está determinada por el estado inicial y no requiere ninguna entrada adicional. El "tablero de juego" consiste en una cuadrícula plana compuesta por cuadrados (llamados "células") que se extiende infinitamente en todas las direcciones. Cada célula tiene ocho células "vecinas", que son aquellas adyacentes a ella, incluyendo las diagonales. Las células pueden estar en uno de dos estados: "vivas" o "muertas" (también conocidas como "encendidas" y "apagadas"). La evolución del estado de las células ocurre en intervalos discretos de tiempo, o turnos. En cada turno, el estado de todas las células se evalúa para determinar su estado en el siguiente turno. Todas las células se actualizan simultáneamente siguiendo estas reglas:
Nacimiento: Una célula muerta se convierte en viva en el siguiente turno si exactamente tres de sus células vecinas están vivas.
Muerte: Una célula viva puede morir por uno de dos motivos:
Sobrepoblación: Si tiene más de tres vecinos vivos.
Aislamiento: Si tiene menos de dos vecinos vivos.
Supervivencia: Una célula viva permanece viva en el siguiente turno si tiene dos o tres vecinos vivos.

2.4.3 Estado Finales

Normalmente, después de un cierto número de ciclos, el sistema puede alcanzar uno de los siguientes estados finales:
Extinción: Después de un número finito de generaciones, todas las células vivas desaparecen.
Estabilización: La población se estabiliza después de un número finito de generaciones, ya sea en una forma fija o en un patrón oscilante entre dos o más formas.

Crecimiento constante: La población continúa creciendo indefinidamente con cada turno. Inicialmente, esta evolución se consideraba solo teórica, pero más tarde se descubrieron patrones que crecían sin límites durante un número infinito de turnos.

Osciladores

Los osciladores son patrones que se repiten a sí mismos tras un número finito de generaciones, es decir, regresan a su estado inicial después de ciertos ciclos. El número de generaciones necesarias para que esto ocurra define el período del oscilador. Existen osciladores para todos los períodos, ya que hay reglas que permiten generar osciladores con cualquier período deseado.

Un oscilador consta de un rotor y un estátor. El rotor incluye las células que cambian de estado durante la evolución del oscilador, mientras que el estátor consiste en las células que permanecen vivas en todas las fases de esta evolución. (Fig.11)

Vidas estáticas

Las vidas estáticas son patrones que permanecen invariables de una generación a la siguiente. Se pueden considerar como osciladores de período 1. Generalmente, se asume que las vidas estáticas son finitas y no vacías. Se pueden clasificar en vidas estáticas estrictas y pseudo vidas estáticas. Las vidas estáticas estrictas son aquellas cuyas partes no son estáticas por sí mismas.

Naves espaciales o planeadores

Las naves espaciales, o planeadores, son patrones que reaparecen en una posición diferente después de completar su ciclo. Es decir, tras un número finito de generaciones, regresan a su estado original, pero en una ubicación distinta. La velocidad de una nave espacial se define como el número de celdas que se desplaza dividido por la duración de su período. La velocidad máxima posible es una celda por generación, conocida metafóricamente como c (la velocidad de la luz). (Fig. 12)

Matusalenes

Los matusalenes son patrones que pueden evolucionar a lo largo de muchos turnos, o generaciones, antes de estabilizarse. El patrón "*Diehard* desaparece después de 130 turnos, mientras que "*Acorn"* tarda 5206 turnos en estabilizarse en forma de muchos osciladores, y en ese tiempo genera 13 planeadores.

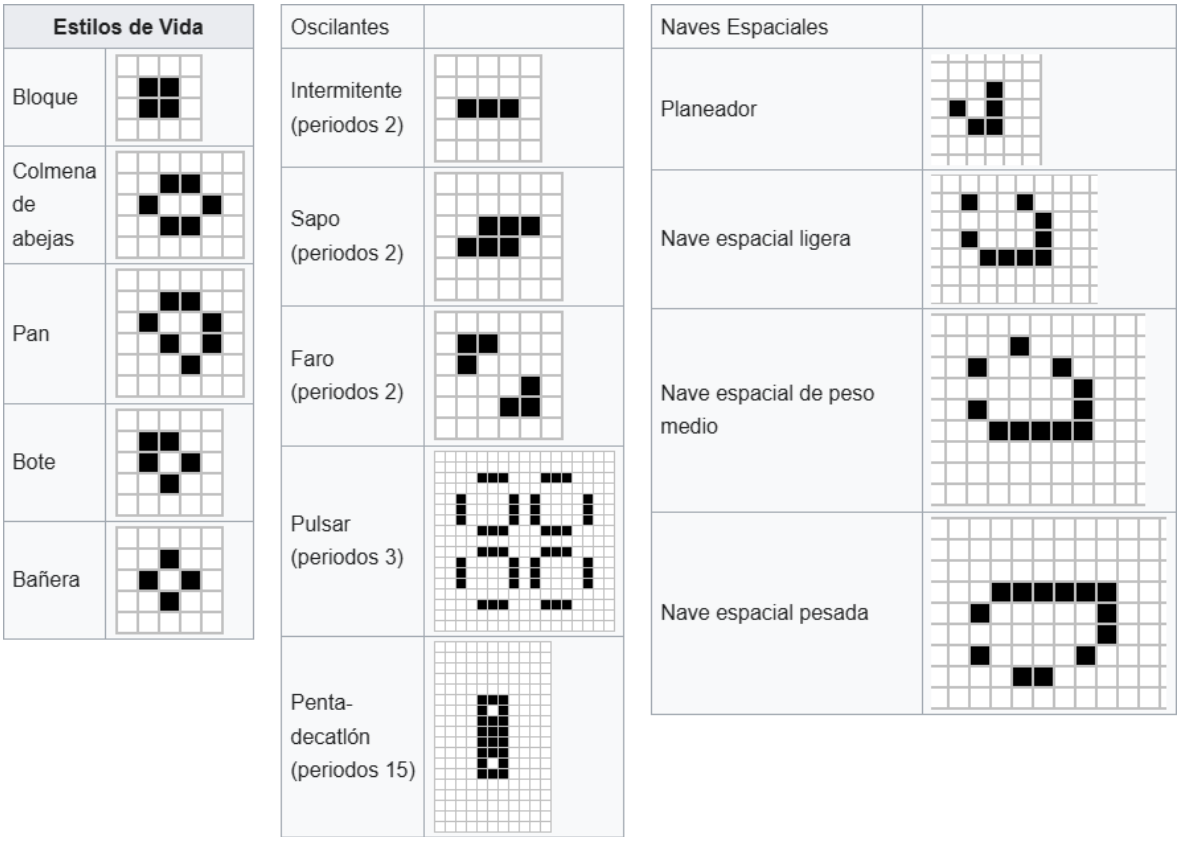


Fig. 17 Tipos de Patrones de Celdas que pueden tener en el juego de la vida.
Fuente: Wikipedia

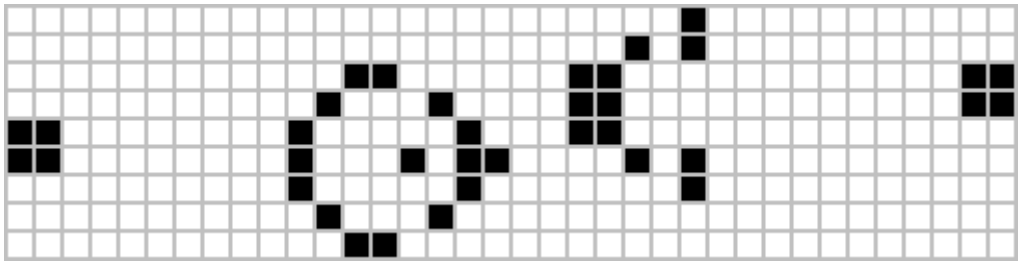


Fig. 18 Cañón de planeadores de Gasper (Gasper Glider Gun). Fuente: Wikipedia

2.4.4 Geometrías basadas en autómatas celulares

Este proyecto continúa la exploración de un enfoque procedimental de la forma. En lugar de trabajar con superficies como en los experimentos de subdivisión, este proyecto utiliza celdas volumétricas (vóxeles) como su geometría básica.
Estos vóxeles contienen datos que pueden interactuar con datos de vóxeles próximos de acuerdo con conjuntos de reglas preestablecidos. Al realizar de forma iterativa estas interacciones, los datos se pueden propagar a través del espacio de vóxeles. Eventualmente, estos datos se pueden visualizar, ya sea como elementos individuales o como un casco que rodea elementos con valores específicos.

Se exploran dos grandes algoritmos para controlar la interacción de vóxeles entre ellos: los autómatas celulares similares al Juego de la Vida y los procesos de difusión de reacciones. En el primer proceso, las celdas suelen tener solo uno de los dos estados (encendido/apagado), cuya elección depende de la combinación de los estados de los vóxeles circundantes.
Este último proceso, la difusión por reacción, simula las interacciones químicas entre las sustancias contenidas en los vóxeles. Este proceso se ha asociado con la formación de patrones no sólo en una serie de organismos, sino también en los campos de la geología y la ecología. (Fig.19)

Fig. 19 Forma de reacción-difusión. Fuente: Michael Hansmeyer

Columnas Subdivididas

Este proyecto de Michael Hansmeyer consiste en la concepción y diseño de un nuevo orden de columnas basado en procesos de subdivisión. Explora cómo la subdivisión puede definir y embellecer este orden de columnas con un elaborado sistema de ornamentos.

Una columna dórica abstracta se utiliza como formulario de entrada para los procesos de subdivisión. A diferencia de la entrada mínima del proyecto Platonic Solids, la columna abstracta transmite información topográfica y topológica significativa sobre la forma que se va a generar. El formulario de entrada contiene datos sobre las proporciones del eje, el capitel y la base de la columna. También contiene información sobre su estriado y éntasis.

El formulario de entrada se etiqueta para permitir que el proceso de subdivisión distinga entre componentes individuales. Esto permite una aplicación heterogénea del proceso, con distintos ajustes de parámetros locales.

La columna subdividida inicial se fabricó con pizarra gris en ETH Zurich en 2010. Posteriormente, se produjeron cuatro columnas de ABS fresadas por CNC para la Bienal de Diseño de Gwangju 2011. El proyecto se revisó para la Expo 2017 en Astaná, haciendo uso de una nueva cortadora láser en serie de alta velocidad. (Fig. 20)

Fig. 20 Columnas Subdivididas Fuente: Michael Hansmeyer

2.5 Aldo van Eyck en el parque infantil

Ámsterdam en los años finales de la década de los 40 del siglo XX. Se trata de una ciudad sumida en la posguerra. Gran parte de la ciudad precisaba una intervención urgente: algunos edificios afectados por los bombardeos quedaron en un estado ruinoso, y se decidió la demolición con el fin de evitar accidentes. Esto provoco la aparición de numerosos solares en el tejido urbano consolidado. La ciudad poco a poco fue apropiándose de estos vacíos, dotándoles de usos concretos, como estacionamientos para el creciente número de vehículos que circulaban en la ciudad. Pero la mayoría de ellos se convirtieron en simples vertederos donde los escombros y basura crecían por momentos. Muchos niños, dada la falta de lugares apropiados para ellos, utilizaban para jugar. Esta situación provoco que muchos ciudadanos escribieran al ayuntamiento de la ciudad para solicitar un lugar apropiado para sus hijos.

De aquí surgió la oportunidad que llevo a Aldo van Eyck[5] a diseñar su primer parque en el año 1947. Su trabajo comenzó con el estudio de la tradición holandesa en espacios lúdicos. A raíz de este análisis, reinterpreto los elementos principales, aislándolos de su carácter ornamental y tomando la esencia de la razón que van a ser utilizados por niños.

La situación que vivía Holanda, y más concretamente Ámsterdam y Roterdam y otros. Dado el grado de devastación de la ciudad y los recursos económicos eran escasos o destinados a otras tareas. CarlTheodor Sorensen ideo una serie de equipamientos vinculados a viviendas sociales de urgencia en que los niños podrían pasar tiempo jugando.

Estos equipamientos se llamaron JunK Playgrounds (Parques de Basura) la idea era muy sencilla. Se trataba de organizar lo que los niños ya hacían por cuenta: facilitar el acceso a un recinto cerrado en el que podrían encontrar para su uso y disfrute diferentes materiales de construcción, escombros y otros. Este juego controlando, eliminando los materiales con potencial peligro, se desarrolla en zonas abiertas, solares sin construcción o espacios libres de la ciudad. (Fig. 21).

¨El Objetivo de la educación, primero, ante todo, es claramente es a estimular que el niño sea niño, no convertirlo en adulto¨. Esta situación nos lleva a concebir la ciudad como un elemento agresivo al niño. Aldo van Eyck pone al niño como principal agente y usuario de la ciudad. El niño como motor de decisiones como polo de atracción y generador de estímulos urbanos.

El niño es el máximo exponente de usuario en la ciudad, ya que todo aquello que hacemos de niños nos forja como adultos del futuro. Basta con dejar hacer el niño, dejar su imaginación y su manera de vivir la ciudad se libera para ver la necesidad de la ciudad.

[5] Aldo van Eyck (1918-1999) fue un arquitecto y urbanista holandés, destacado miembro del grupo Team 10. Es famoso por su trabajo en el Orfanato Municipal de Ámsterdam y su enfoque en la creación de espacios que fomenten la interacción humana y la comunidad. Para más información, ver: Strauven, F. (1998). *Aldo van Eyck: The Shape of Relativity*. Amsterdam: Architectura & Natura Press.

Imaginación, apropiación, y pertenencia son tres cuestiones claves en la relación del niño con los parques. Son también herramientas de diseño, intangibles, que Aldo van Eyck incluyo en su desarrollo compositivo en el proceso de creación de los parques infantiles. Gracias a estas herramientas el niño es capaz de dotar vida aquello ajeno al juego y diseñado con otro fin. Precisamente ahí radica el logro de los elementos configuradores de los parques infantiles. Aldo van Eyck fue capaz de diseñar parques y elementos de juego con una configuración abierta, que el niño exploraba libremente. Simplemente ideo un soporte, un lienzo, para explorar la creatividad infantil. De este modo, los parques adquieren vida gracias al niño, donde el niño encuentra elementos geométricos abstractos inertes, pero potencialmente puede ser cualquier cosa que el niño imagine. (Fig.21).

Fig. 21 Imagen de los ¨jumping Stone¨ del parque Transvaaplein año 1950.
Fuente: Tesis Jaime Álvarez Santana

CAPITULO 3

TESTEO DE ALGORITMOS

3 CASOS DE ESTUDIO LOGICO-GRAFICO

¨Un nuevo lenguaje requiere una nueva técnica ¨

Philip Glass

En este capítulo, exploraremos tres sistemas de crecimiento: crecimiento generativo, crecimiento diferencial y crecimiento mediante autómatas celulares, utilizando un algoritmo de programación gráfica como Grasshopper. Tomando el coral como inspiración y analogía, analizaremos cómo estos sistemas pueden replicar y generar patrones de crecimiento orgánico. El crecimiento generativo se centrará en la creación de formas mediante algoritmos y parámetros controlados. El crecimiento diferencial simulará la variación de crecimiento en distintas partes de la estructura. Mientras tanto, el crecimiento mediante autómatas celulares empleará reglas simples para determinar la evolución de la forma. Este enfoque nos permitirá comprender los principios del crecimiento biológico y aplicarlos creativamente en el diseño y la simulación de estructuras complejas, aprovechando la belleza y la eficiencia estructural del coral como referencia.

SISTEMA DE CRECIMIENTO **GENERATIVO**

3.1 Sistema de crecimiento generativo del Coral Acropora

Para investigar y crear un modelo relacionado con el crecimiento generativo, es esencial comprender el proceso de generación y reproducción del coral, además de considerar datos cuantitativos, ya que el diseño incluirá parámetros para simular este organismo. Para desarrollar un algoritmo gráfico, será necesario simular al sol como un agente modificador que, según su movimiento en relación con el coral, pueda influir en su generación y en su resultado geométrico físico. Utilizando software como Rhinoceros 7.0 y Grasshopper, junto con complementos como Anemone, Kangaroo y Cocoon, estableceremos un conjunto de parámetros para controlar el sistema biocomputacional.

3.1.1 Coral Acropora

El Acropora cervicornis, también llamado coral "cuerno de ciervo" o "estrella", es un coral duro que pertenece al género Acropora. Habita en aguas tropicales poco profundas del Atlántico occidental, desde Florida hasta el Golfo de México y el Caribe. Destacado por su papel fundamental en la formación de arrecifes, esta especie contribuye notablemente a la estructura y biodiversidad de los arrecifes de coral en estas regiones. (Fig.1)

Hábitat y distribución

Suelen vivir en zonas pocas profundas de 1 a 40 m, ocasionalmente en 60 m y usualmente entre 1 y 25 m, bien iluminadas y cercanas a las costas. Aunque también se encuentran el lagunas y zonas protegidas, mayoritariamente se dan en zonas de fuertes corrientes.

Morfología

La especie Acropora exhibe una diversidad en su desarrollo influenciada por diversos factores como la ubicación, condiciones ambientales y tamaño. Esta variabilidad complica la identificación de la colonia Acropora, generalmente realizada mediante observación microscópica del esqueleto, caracterizado por ser poroso y ligero.

La colonia de Acropora adopta una estructura arborescente, con ramas más dispersas que otras especies similares. Las ramas secundarias crecen perpendicularmente a la principal, pudiendo alcanzar longitudes de 1 a 2 metros y diámetros de 4 a 7 centímetros. Estas colonias pueden extenderse varios metros.

Su crecimiento es notable, entre 10 y 20 centímetros por año, aunque este proceso se ve contrarrestado por la rotura de ramas durante las tormentas. La especie suele presentar colores dominantes de amarillo y marrón.

Alimentación

Los pólipos contienen algas simbióticas llamadas zooxantelas. Las algas realizan la síntesis produciendo oxígeno y azucares, que son aprovechados por los pólipos, y se alimentan de los

catabolitos del coral (especialmente fosforo, nitrógeno) Esto les proporciona entre el 75% y 95% de sus necesidades alimenticias. El resto se obtiene atrapando por plancton microscópico y materia orgánica disuelta en el agua.

Reproducción – Generación

En general alcanza la madurez sexual entre los 3 y 5 años, con un diámetro de sus ramas 4cm a 7 cm. Se reproducen asexualmente mediante la gemación y por fragmentación, siendo este último el modo dominante de reproducción, cuando las ramas de las colonias se rompen a debido a los temporales, y sus fragmentos originas nuevas colonias.

Sexualmente son hermafroditas simultáneos, lo que quiere decir que las colonias generan gametos masculinos y femeninos, lanzando simultáneamente al exterior sus células sexuales, siendo por tanto la fecundación externa, permanecen a la deriva arrastrados por las fuertes corrientes, más tarde se forma una plánula, que tras, de deambular cae al fondo, secreta carbonato de calcio para conformar un esqueleto, o coralito. Posteriormente forman la colonia mediante la división de pólipos por gemación.

Fig. 1 Fotografía de una familia de arrecife corales de genero Acropora Cervicorni, donde el sistema reacciona a cada cambio en los flujos del agua y tiende a elevarse con los rayos del sol. Fuente: Bonaire, 2007

3.1.2 Enfoques aplicados al proceso de diseño

Es preciso establecer una analogía con las fases del proceso de diseño desde la perspectiva de la generación geométrica computacional, enfocándose específicamente en el software utilizado para dicha generación, en este caso Rhinoceros 7.0 y Grasshopper. Por consiguiente, esta analogía se presenta mediante una serie de pasos realizados con el mencionado software. (Fig. 2)

1. Geometria de soporte a partir de una superficie **RHINO +GRASSHOPPER**
2. Generacion de ciclos de ramificación GR**ASSHOPPER+ANEMONE**
3. Presion del agua y corrientes marinas **GRASSHOPPER+KANGAROO**
4. Modificador del crecimiento del sol **GRASSHOPPER**
5.Representacion de la geometría afectada **GRASSHOPPER**
6.Variacion de la geometría en crecimiento **RHINO+GRASSHOPER**
7. Representacion geométrica final **GRASSHOPPER+COCOON**

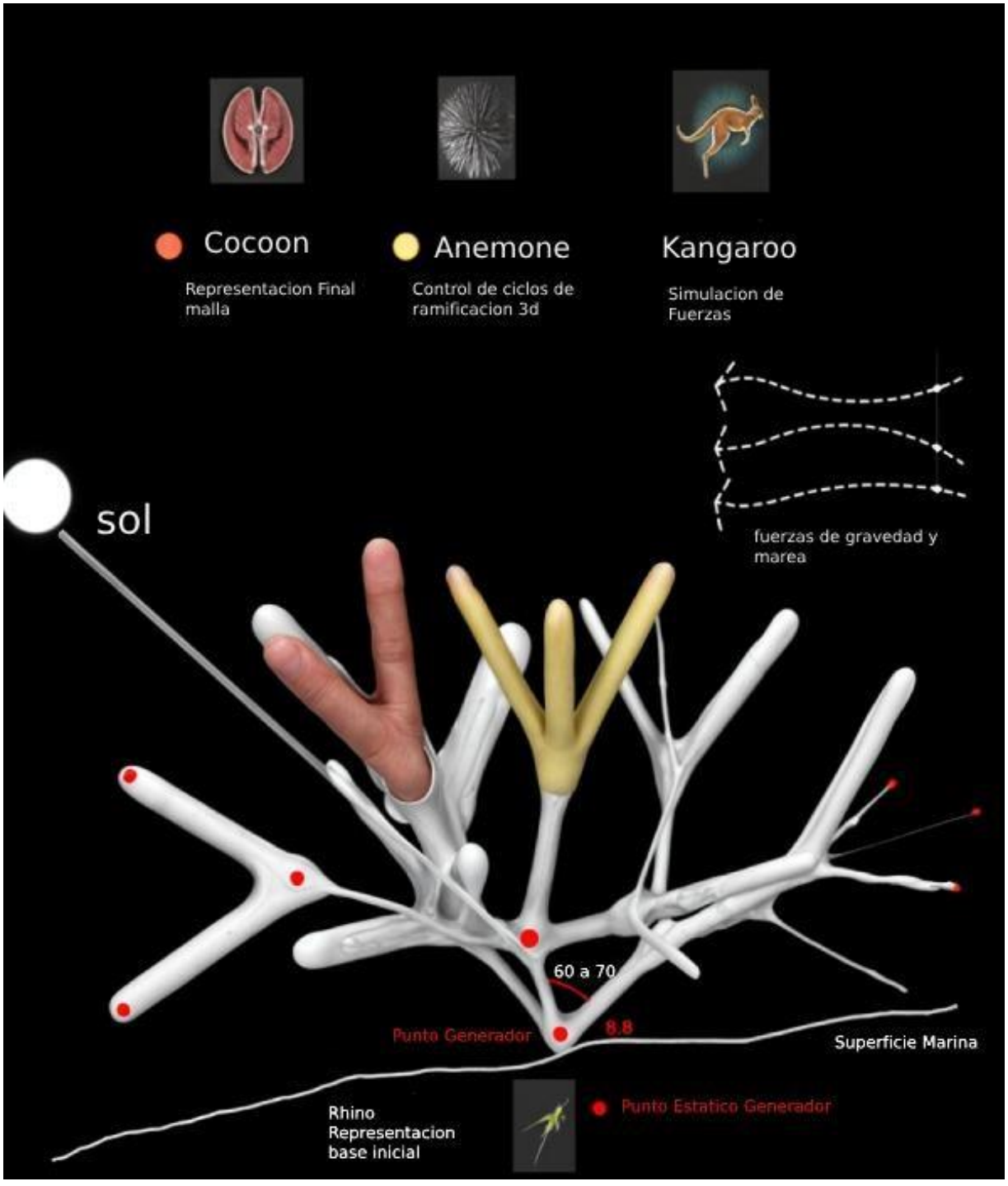


Fig. 2 Esquema de diseño, el cual contempla las etapas computacionales. Fuente: Elaboración Propia

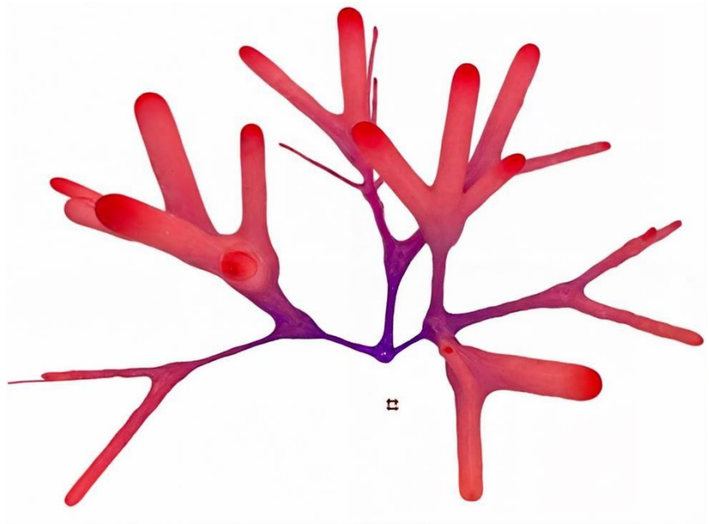


Fig. 4 Coral Acropora . Fuente: Elaboración Propia

1 ETAPA Geometría de soporte a partir de una superficie **RHINO +GRASSHOPPER**

1 ETAPA generación de ciclos de ramificación GR**ASSHOPPER+ANEMONE**

2 ETAPA presión del agua y corrientes marinas **GRASSHOPPER+KANGAROO**

3 ETAPA Modificador del crecimiento del sol **GRASSHOPPER**

3 ETAPA. Representación de la geometría afectada **GRASSHOPPER**

3 ETAPA. Variación de la geometría en crecimiento **RHINO+GRASSHOPER**

4 ETAPA. Representación geométrica final **GRASSHOPPER+COCOON** (Fig. **3**) (Fig.4)

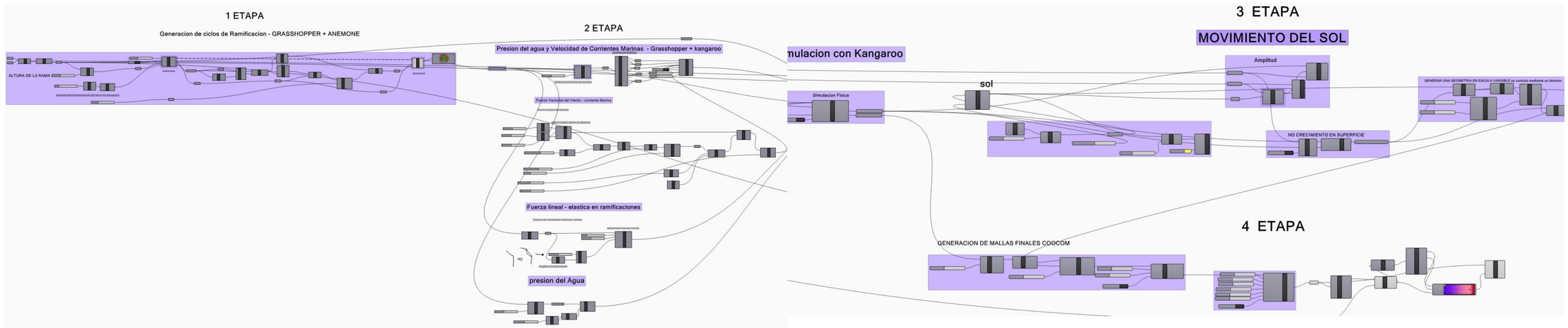


Fig. 3 Algoritmo en sus diferentes etapas de diseño de crecimiento generativo. Fuente: Elaboración Propia

1 ETAPA Geometría de soporte a partir de una superficie **RHINO +GRASSHOPPER**

1 ETAPA generación de ciclos de ramificación GR**ASSHOPPER+ANEMONE**

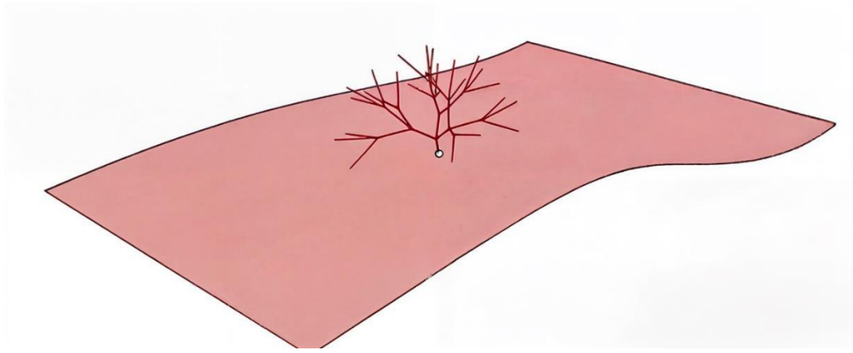


Fig. 6 Ciclos de Ramificación. Fuente: Elaboración Propia

1 ETAPA DE DISEÑO: Generación de Ciclos de Ramificación (GRASSHOPPER+ANEMONE)

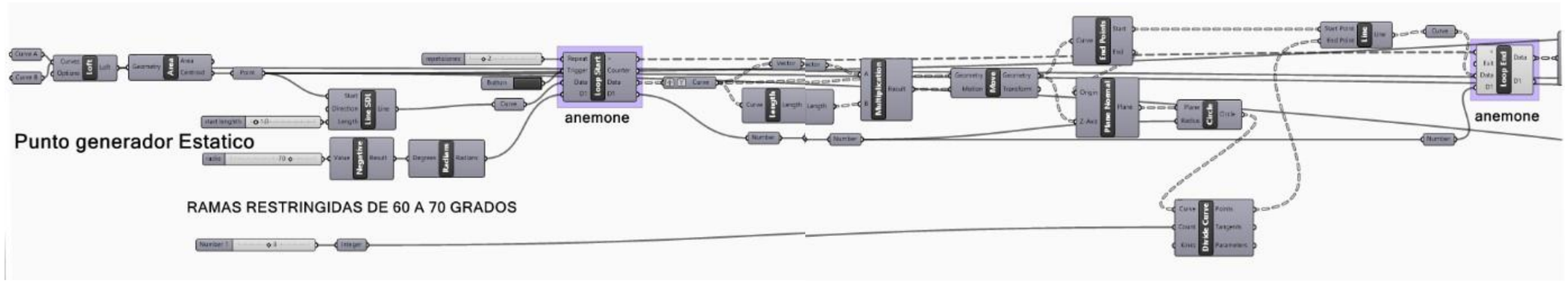


Fig. 5 Algoritmo 1 Etapa de Diseño, ciclos de Ramificación. Fuente: Elaboración Propia

3.1.3 Primera Etapa del Diseño

Como principio del análisis, el organismo posee un soporte inicial, que representa la superficie marina, así como el inicio de la ramificación y de soporte y generación principal de todo el sistema. A partir de estas ramas surgen las subyacentes, que poseen un número máximo de divisiones que se restringen en tres ramas. Por otra parte, cada una de las ramificaciones que conforman todo el sistema, se encuentras expuestas al entorno, y por consiguiente a las fuerzas que modifican su crecimiento y propagación a lo largo de la vida del organismo; las ramas además están restringidas por ángulos de 60 a 70 grados que son modificados por fuerzas. (Fig. 7)

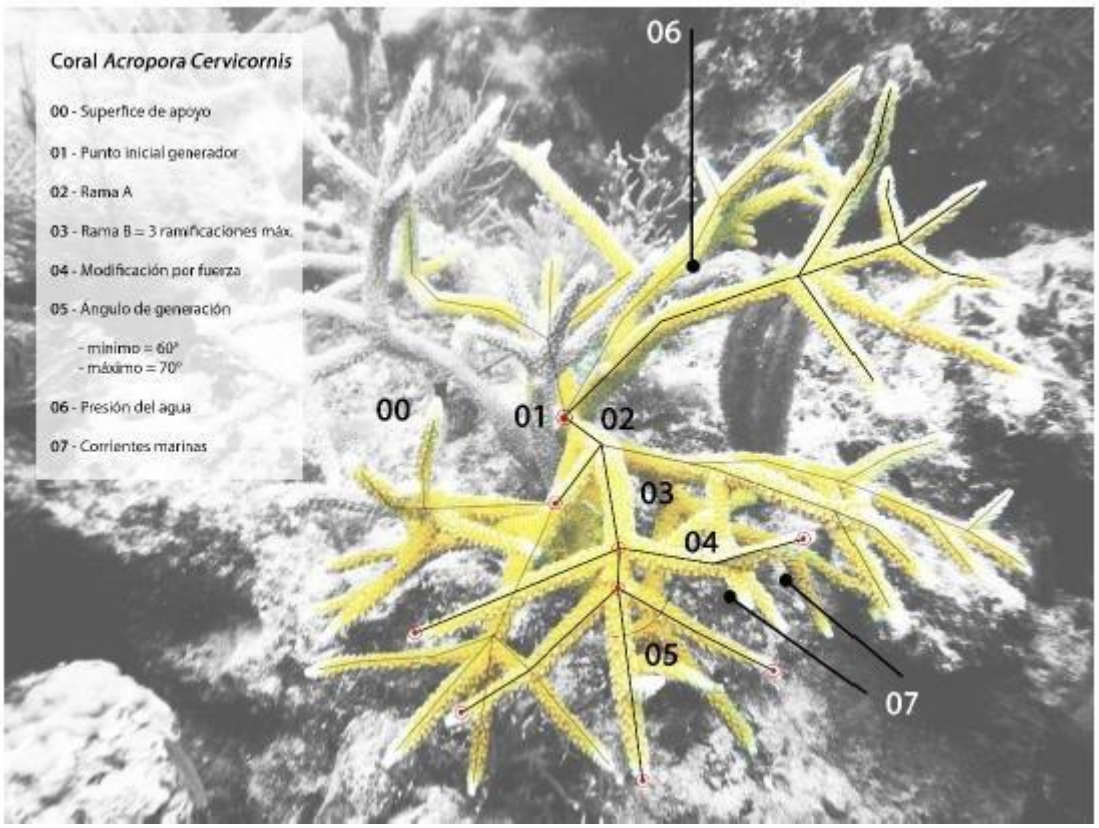


Fig. 7 Configuración de generación de la forma del género Acropora. Fuente: Fotografía del biólogo y ecólogo Dr. Diego Liman. Miami.20212. EUA

Superficie Marina + Punto inicial del Generador = Soporte del Sistema

Para poder distinguir y comprender cada una de las fases de generación, se colocará el grupo final de la fase, y se explicara los componentes y parámetros que se aplicó.

La fase inicial contempla la imitación del soporte del coral Acropora, es decir, el medio de sustento físico a partir del cual se genera. Este soporte se da en la superficie o rocas marinas donde el organismo se desarrolla, pero como se desarrolla en un entorno modelado computacional 3d no existe organismos digitales. La primera premisa es generar dos curvas que se llama Curve A y Curve B, estas nos permitirán continuar con la superficie de soporte inicial a través del componente loft de Grasshopper nos permite generar una superficie controlada. (Fig. 8)

El componente mesh área, localizara y generara un punto central 3D en la superficie generada, si la superficie es modificada, el punto cambiara de posición, adaptándose al medio geométrico generador inicial. Este punto será desplazado hacia la parte inferior de la superficie para generar una rama inicial con el componente SDL. **El resultado es una curva**. (Fig. 9)

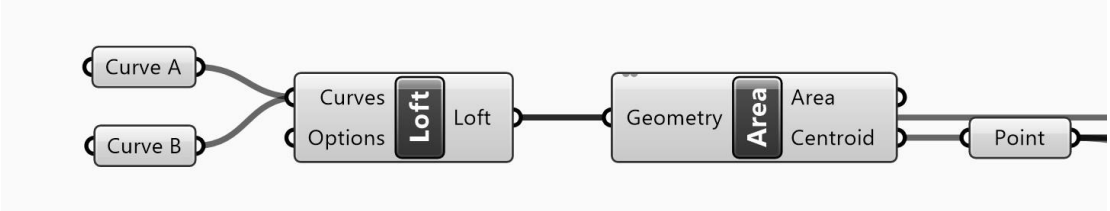


Fig. 8: Esquema especifico que muestra la fase inicial de generación a través de una superficie. Fuente Elaboración Propia

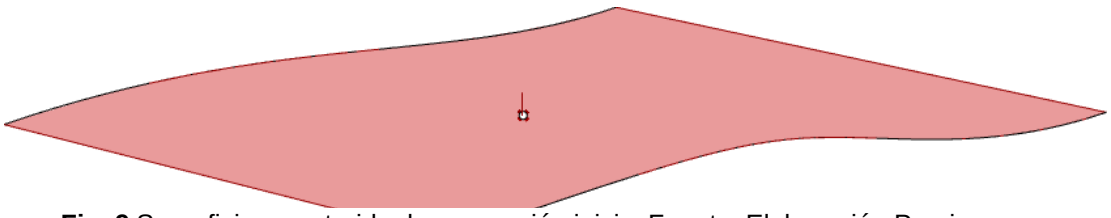


Fig. 9 Superficie y centroide de generación inicia. Fuente: Elaboración Propia

Fase de Soporte del Sistema

Esta fase implica el desarrollo del soporte del sistema, que se puede comparar con el tallo de un árbol, el cual servirá de base para la ramificación.

El componente Line SDL se conectará con Curve y simultáneamente se enlazará con dos componentes: Vector y Curve Length. Estos, a su vez, se conectarán con las entradas del componente Multiplicación. Para obtener una copia, utilizaremos el componente Move, el cual nos ayudará a duplicar la curva resultante. Estos pasos detallados (Fig.10)

Conexión de Line SDL con Curve:
Conecta el componente Line SDL con el componente Curve. Esto establecerá la relación inicial necesaria entre estos componentes.
Enlace de Line SDL con Vector y Curve Length:
Conecta Line SDL con el componente Vector. Esto permitirá que las propiedades del vector influyan en la línea SDL.
Conecta también Line SDL con el componente Curve Length. Esto proporcionará la longitud de la curva a los componentes subsiguientes.
Conexión de Vector y Curve Length con Multiplicación:
Conecta las salidas del componente Vector y del componente Curve Length con las entradas correspondientes del componente Multiplicación. Esto permitirá realizar las operaciones de multiplicación necesarias utilizando los valores derivados de ambos componentes.
Uso del componente Move para hacer una copia:
Conecta la curva obtenida a través del proceso anterior con el componente Move.
Configura el componente Move para realizar la duplicación de la curva según los parámetros deseados.

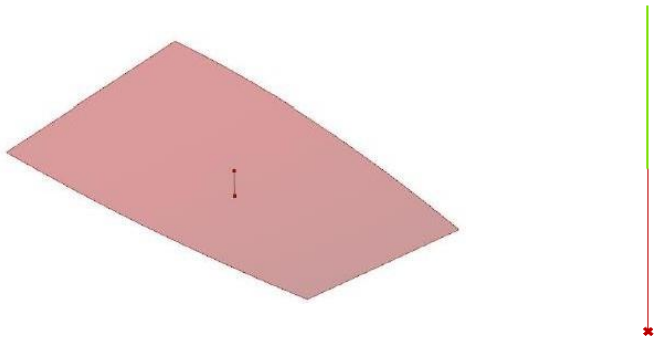


Fig. 10 Línea vectorial de generación inicial. Fuente; Elaboración Propia

A partir del soporte surgen las primeras ramas, en las cuales se desarrollarán las estructuras adicionales del sistema. Estas ramas iniciales proporcionarán la base para el crecimiento y la expansión del modelo, permitiendo la generación de nuevas ramificaciones y elementos secundarios que contribuirán a la complejidad y realismo del diseño final.

Para conectar el componente "move" a dos componentes simultáneamente, se conectará a "end points" y "plane Normal". El componente "end points" coloca puntos al inicio y al final de la curva, eligiendo el punto final de la curva y conectándolo al origen de "Plane Normal". Luego, arrastramos el componente "círculo" y conectamos el "plane" al componente "Plane Normal", creando así un círculo con un radio especificado. A través de un "slider", se define el radio del círculo en 70 grados. Se utilizará el componente "radianes", que transforma ángulos a grados, y se conectará al componente que define el radio del círculo.

Añadir el componente Divide Curve:
Arrastra el componente "Divide Curve" al lienzo de Grasshopper.
Este componente se utiliza para dividir la curva del círculo en segmentos iguales.
Configurar el número de segmentos:
Conecta un "slider" al componente "Divide Curve" para definir el número de segmentos. En este caso, configura el "slider" en 3, ya que el coral Acropora tiene un máximo de 3 ramificaciones.
Añadir el componente Line:
Arrastra el componente "Line" al lienzo.
Este componente requiere dos puntos: el punto de inicio y el punto final de cada línea.
Conectar puntos de inicio y final:
Conecta el punto de inicio del componente "Line" al componente "End Points".
Conecta el punto final del componente "Line" al componente "Divide Curve".
Resultado:
El resultado será la creación de 3 ramas a partir del círculo, simulando las ramificaciones del coral. (Fig.12)

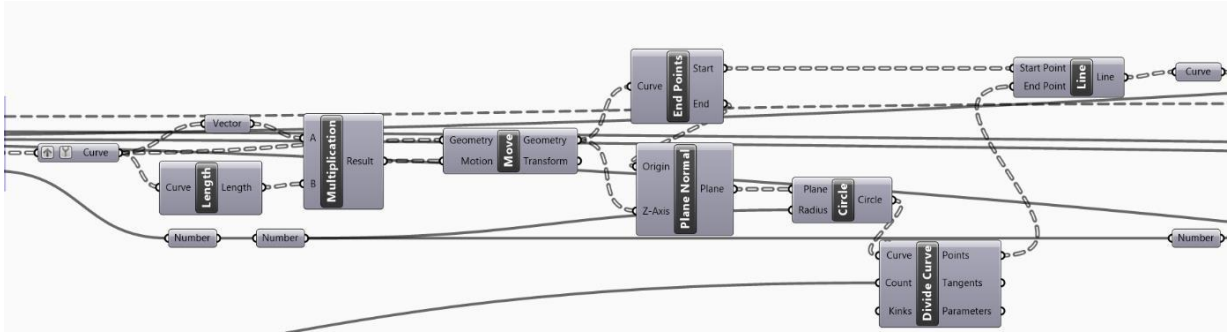


Fig. 11 Esquema específico del soporte del sistema y ramificación. Fuente: Elaboración Propia

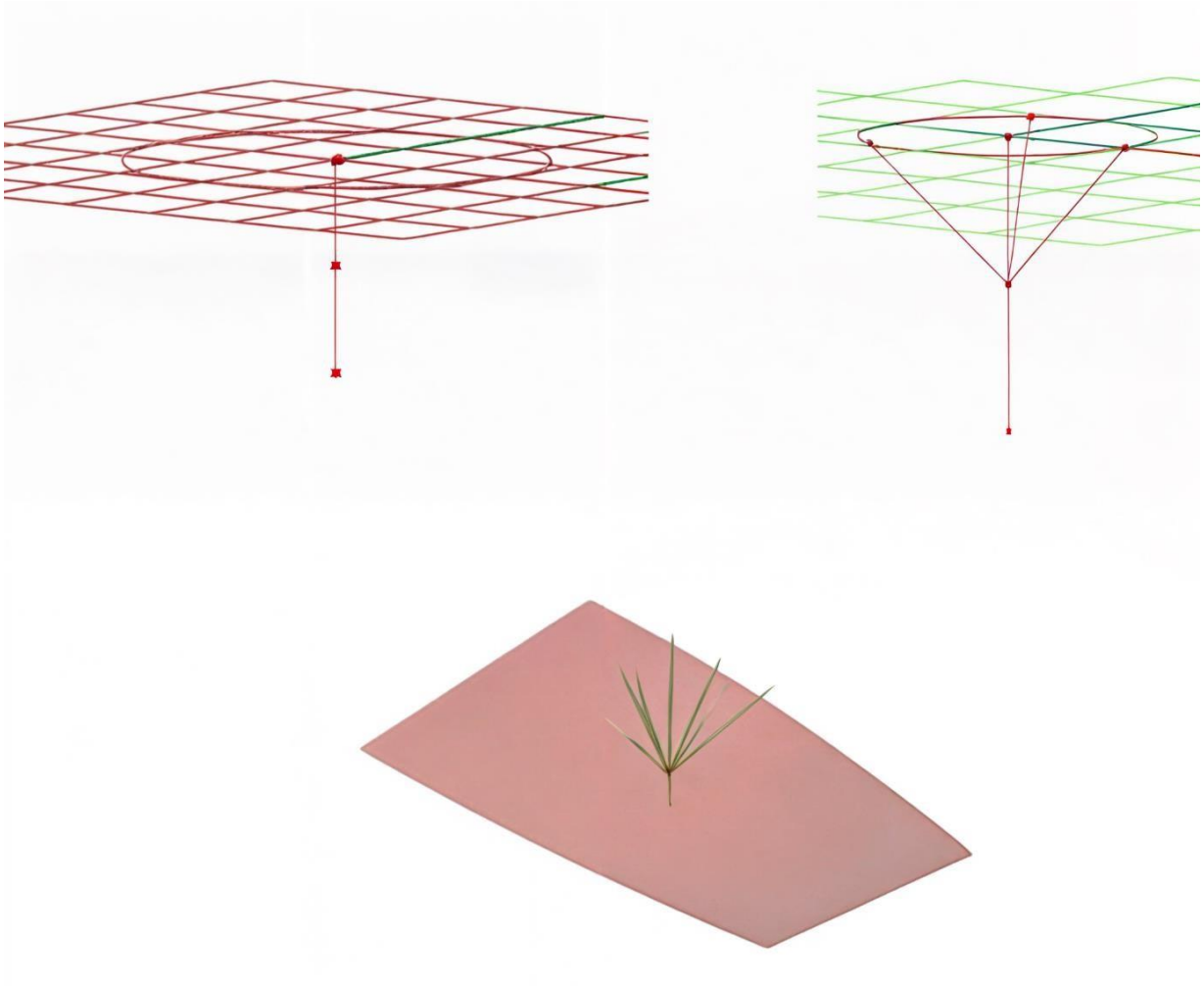


Fig. 12 Punto generador de las ramificaciones. Y ramificaciones. Fuente: Elaboración Propia

Ciclo Exponencial Inicial

El componente externo Anemone (plugin) nos ayudará a generar ciclos de ramificación controlados, basados en la geometría y el control vectorial. Este plugin permitirá definir un ciclo inicial y final de generación, proporcionando un mayor control sobre el proceso de crecimiento y ramificación del modelo.

Con Anemone, se crea bucles utilizando componentes como "Loop Start", "Loop End" y "Iterations", lo que te permite controlar el número de iteraciones y realizar acciones repetitivas de manera eficiente en tu modelo de Grasshopper. Este plugin es especialmente útil para la creación de algoritmos complejos y la exploración de diseños generativos. (Fig. 13)

Implementación del Bucle de Ramificación en Grasshopper usando Anemone

Configurar el Punto de Partida del Bucle (Loop Start):

Arrastra el componente "Loop Start" al lienzo de Grasshopper.

Este componente marcará el inicio del bucle y debe colocarse después del componente "línea SDL" y antes de la curva.

Establecer Parámetros Numéricos:

Conecta los sliders que definen los parámetros numéricos, como el número de repeticiones del sistema, al componente "Loop Start".

Usa un "boolean toggle" para controlar las diversas operaciones, permitiendo activar o desactivar el bucle.

Activar Repeticiones:

Si el número de repeticiones aumenta, presiona el botón ("button") para ejecutar la orden.

Esto iniciará el primer ciclo de generación.

Conectar el Ciclo (Loop Start y Loop End):

Conecta la salida del parámetro del primer ciclo (Loop Start) con el parámetro de entrada del "Loop End".

Esto establece una relación cíclica directa con las variables y números iniciales de control del sistema de generación ramificado. (Fig.14)

Definir la Geometría Resultante:

El componente "curve" será denominado "Ramificaciones Finales" y representará la geometría resultante del ciclo de generación. (Fig.15)

Esta curva será utilizada para realizar simulaciones físicas que afecten al sistema biológico-computacional en conjunto.

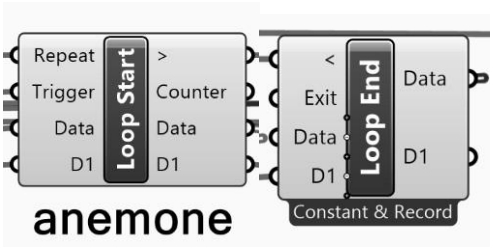


Fig. 13 Ciclo Generador inicial y final Anemone

ANEMONE ANEMONE

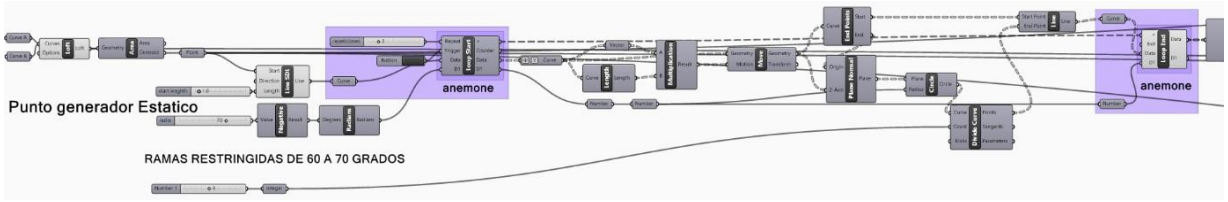


Fig. 14 generación de ciclos de ramificación completa

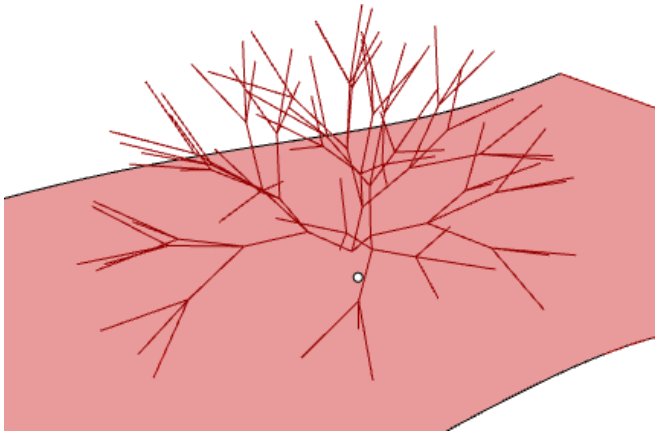


Fig. 15 Sistema Fractal Ramificado

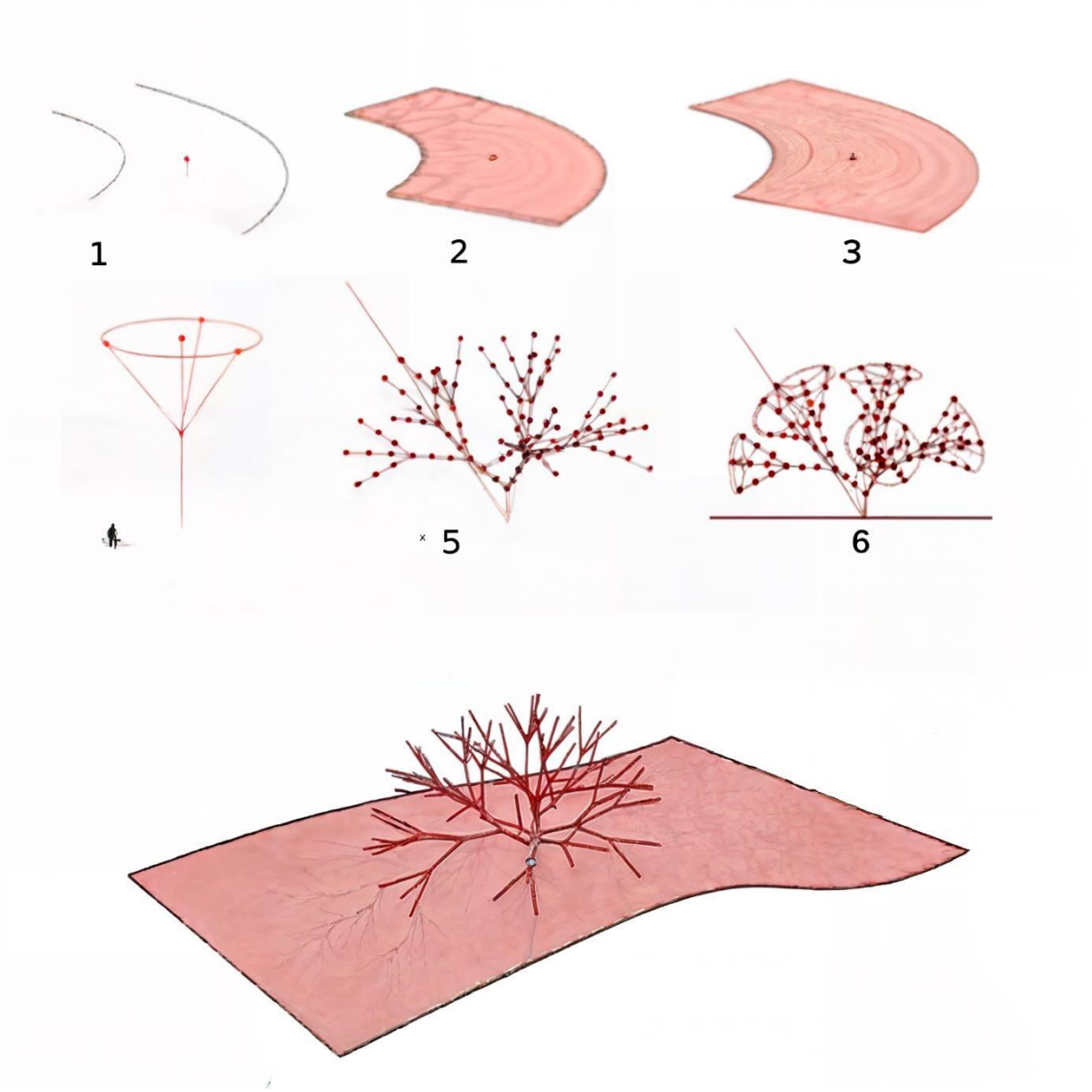


Fig. 16 conformación geométrica lineal final del ciclo fractal de ramificación, basado en un algoritmo de programación grafica de grasshopper . Fuente: Elaboración Propia

3.1.4 Segunda Etapa de Diseño

Entonces, en esta etapa del proceso de diseño, estás trabajando en el desarrollo de conceptos que se basan en los enfoques descritos al principio de la metodología. Estos enfoques integran la física, las matemáticas y la representación de los resultados, junto con modificaciones a través de la geometría, en conjunción con la simulación de fuerzas como la presión del agua y la velocidad de las corrientes marinas, que impactarán en el modelo mediante el uso del plugin Kangaroo[6] para Grasshopper.

[6] Kangaroo es un plugin para Grasshopper, desarrollado por Daniel Piker, que permite realizar simulaciones físicas interactivas en el entorno de modelado paramétrico de Rhinoceros. Para más información, ver: Piker, D. (2013). Kangaroo: Form Finding with Computational Physics. Architectural Design, 83(2), 136-137. https://doi.org/10.1002/ad.1568

1. **GRUPO A** : Aplicación fuerzas
2. **GRUPO B**: Generación de fuerza vectorial del viento – corrientes marinas
3. **GRUPO C**: generación de fuerza lineal- estática en las ramificaciones
4. **GRUPO D**: Presión del mar

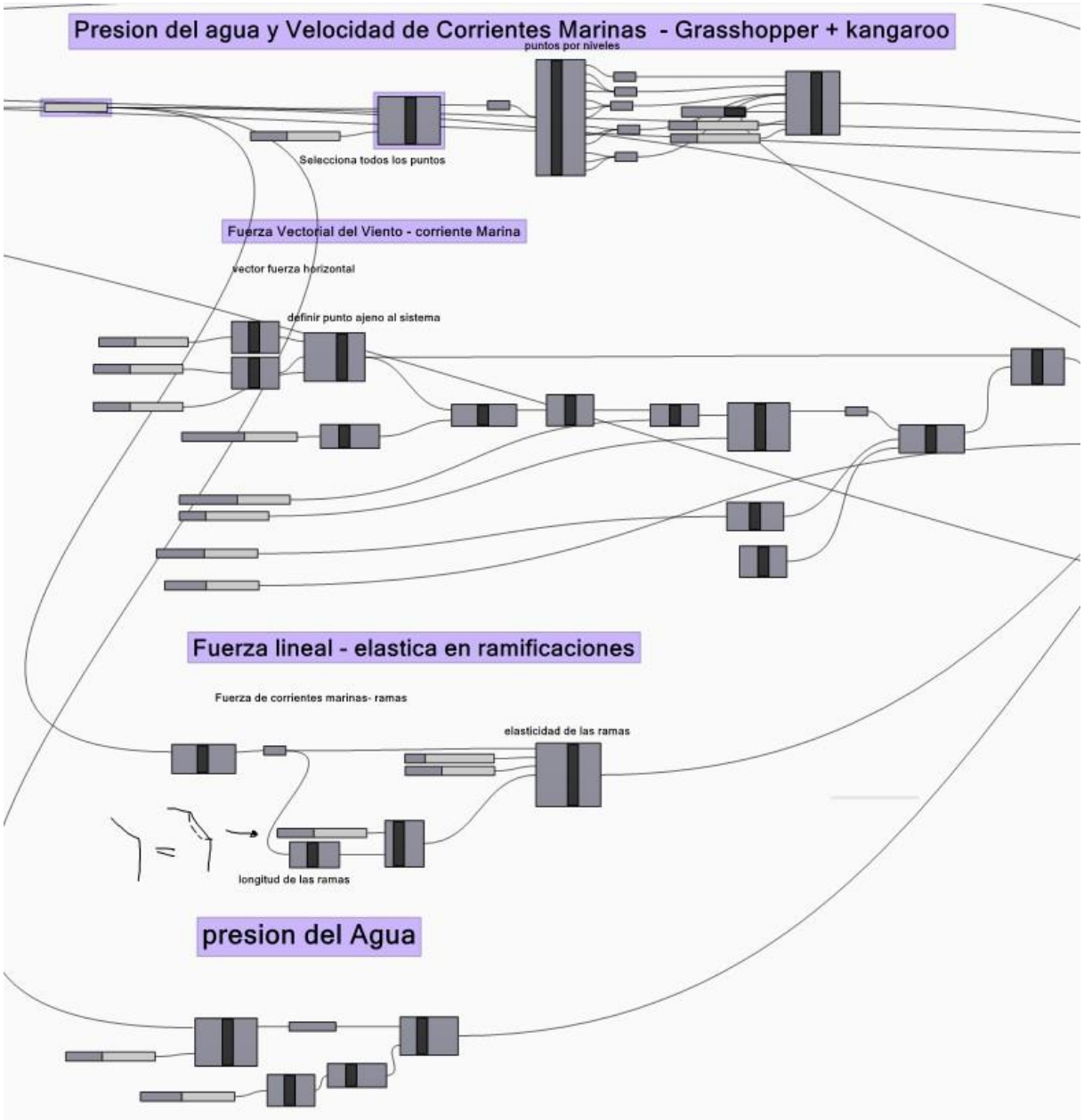


Fig. 17 Segunda Etapa del Diseño. Algoritmo en Grasshopper . Fuente: Elaboración Propia

En el enfoque previo de diseño, aplicado a través de la programación logarítmica gráfica, se desarrolló el ciclo de ramificaciones fractales, desde su inicio hasta su finalización. El resultado de este proceso nos proporcionó la representación de estos ciclos y, por ende, de cada grupo de ramificaciones. Esto se representa a través de un árbol de datos, donde cada iteración de ramas representa un ciclo de generación basado en la analogía de un organismo natural.

Con lo anterior, hemos logrado crear un conjunto unificado que se divide en 13 partes o ramificaciones. Para llevar a cabo una simulación física, es necesario acceder a cada una de estas ramificaciones y descomponerlas en curvas y puntos. Esto nos permite aplicar fuerzas a cada elemento, recopilar su comportamiento y luego volver a reunir el conjunto para representar el resultado de la simulación en un objeto definido por su geometría. (Fig.18)

Es por ello por lo que hemos empleado el componente curve (ramificaciones finales), con el propósito de transferir todos los datos del "Loop End" (ciclo final) a un componente que represente estos datos en forma de geometría. Esto nos facilitará desarrollar la siguiente etapa del proceso. En este caso específico, retiraremos el componente "Ramificaciones Finales" del grupo asociado al ciclo final.

Es importante destacar que, de la misma manera en que hemos utilizado grupos para explicar cada fase del proceso de diseño y su correspondiente representación, en esta sección también emplearemos grupos para abordar el proceso de diseño relacionado con la simulación física y matemática.

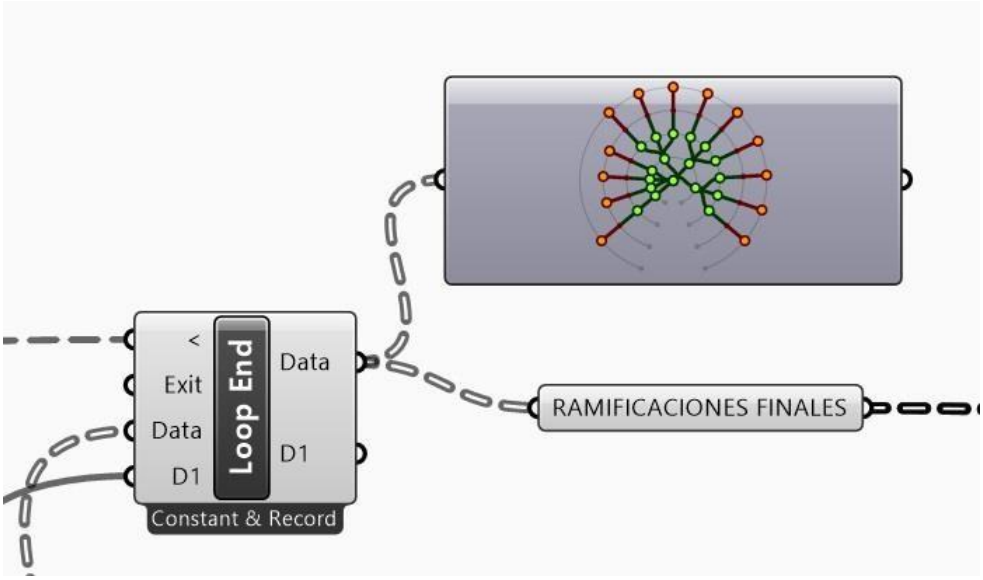


Fig. 18 Grupo de ramificaciones y árbol de datos. Fuente: Elaboración Propia

(Grupo A) Aplicación de Fuerzas

Como se mencionó anteriormente, el primer paso para poder aplicar fuerzas a nuestro ciclo de ramificaciones es desglosar esta geometría en sus componentes fundamentales. (Fig.19) **Es decir, necesitamos identificar cada punto de inicio y final de las ramas, así como la línea que las conecta. Esto nos habilitará para aplicar fuerzas relacionadas con la presión del agua y las corrientes marinas que afectan al organismo. Al finalizar la simulación, podremos nuevamente reunir estos elementos para representar la geometría unitaria en su totalidad.**

Fuerzas - Corrientes Marinas

El paso inicial consiste en segmentar los elementos en puntos, los cuales serán utilizados para aplicar las fuerzas de las corrientes marinas mediante el componente wind de Kangaroo. Este paso es fundamental ya que actúa como el iniciador del primer ciclo de modificación o alteración, ejerciendo su influencia sobre cada uno de los puntos que generan las líneas de ramificación secundaria.

En este caso el componente Evaluate Curve, nos permitirá hacer una selección unitaria de todos los puntos que se encuentra conformando a la totalidad del sistema de ramificación, el parámetro 1 en unión con t determina la selección es de todos los puntos. Con esto obtenemos todos los puntos del sistema. (Fig. 20)

Al tratarse de un sistema unitario, es necesario acceder a cada uno de los ciclos de generación R1+R2+R3+R4 etc representados a través de puntos esto nos ayudara a definir puntos cuales son los más afectados y menos afectados por las fuerzas a aplicar en cada ciclo. Como ya lo hemos mencionado en los corales sucede que las ramas superiores (jóvenes) son afectadas con mayor proporción debido a la relación directa con el entorno (agua).
El componente explode tree (BANG), El cual nos permitirá acceder al sistema de puntos. (Fig.21)

Es necesario tener un punto de apoyo el componente wind lo requiere. En nuestro caso vamos a nombrar un punto de anclaje, lo cual es necesario para poder aplicar una fuerza al sistema, de lo contrario, todo el sistema se desplazaría a causa de la magnitud aplicada. Por lo anterior necesitamos incorporar el punto centroide generador inicial como punto de anclaje. (Fig.21)

Los puntos necesarios a los cuales aplicar fuerza wind están determinados por las interacciones (0,0,0), (1,0,0) asi sucesivamente. La simulación de la fuerza de corrientes marinas se llevará a cabo

con el componente Wind de Grasshoper, a partir de una relación con los de iteraciones obtenidos, que representa los elementos a afectar. (Fig.22)

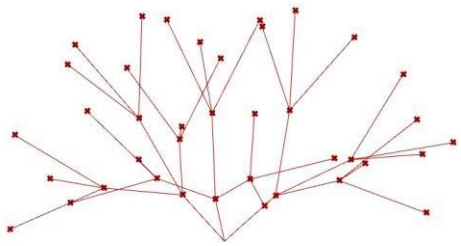


Fig. 19 Totalidad del sistema de ramificación. Fuente: Elaboración Propia

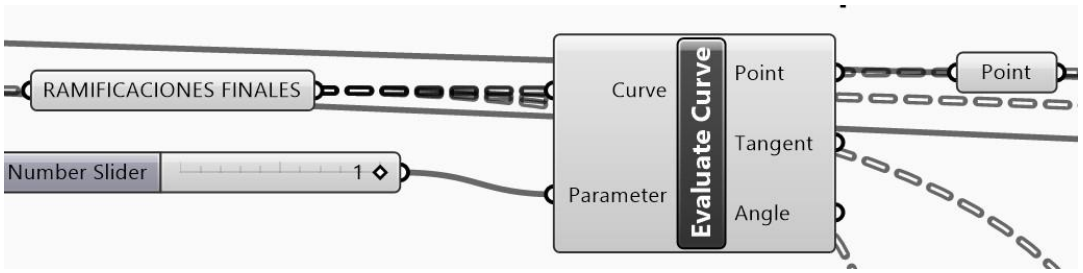


Fig. 20 Evaluate Curva, selección unitaria del sistema de ramificación. Fuente: Elaboración Propia

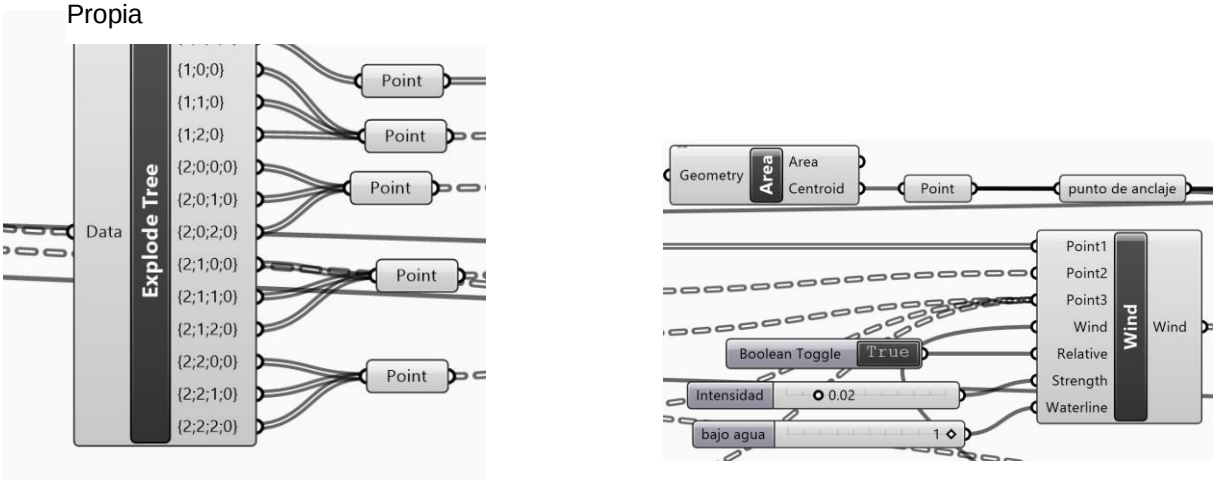


Fig. 21 Componente Bang, accede al sistema de puntos, punto de anclaje- para llevar a cabo la simulación. Fuente: Elaboración Propia

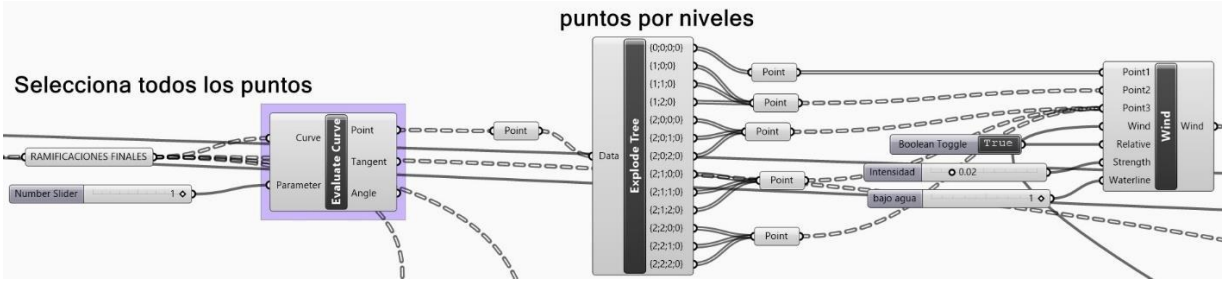


Fig. 22 Grupo A , Aplicación de Fuerzas – Corrientes Marinas completo

Entrada de Point 1 se relaciona con la iteración (0,0,0), (1.0.0), (1,1,0), lo que indica que los miembros de este conjunto experimentarán una menor magnitud de fuerza. Por otro lado, Point 2 será afectado por una fuerza de mayor magnitud en comparación con la otra iteración. En cuanto a Point 3, serán los más afectados por las fuerzas aplicadas, dado que son los miembros más jóvenes del sistema y, por lo tanto, capaces de resistir más que los más ancianos, como ocurre en la naturaleza. (Fig.23)

Por otro lado, el componente "wind" requiere un parámetro llamado "relative", en este caso el valor booleano es verdadero (true), ya que la fuerza aplicada varía en cada iteración. El parámetro de entrada "Strength" (intensidad) es de 0.02, lo que representa una magnitud adecuada para un sistema orgánico ubicado bajo el mar, según el parámetro de entrada "Waterline". En este parámetro, un valor de 1 indica que el sistema está bajo la influencia del agua (mar), mientras que un valor de 0 indica que no hay influencia de este elemento.(Fig.24)

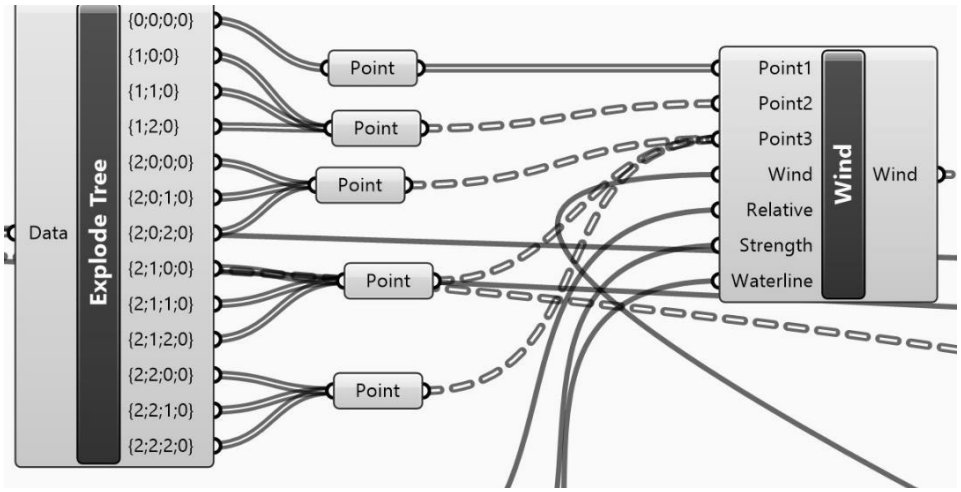


Fig. 23 Relación Puntos de interacción con entradas de puntos a afectar. Fuente: Elaboración Propia

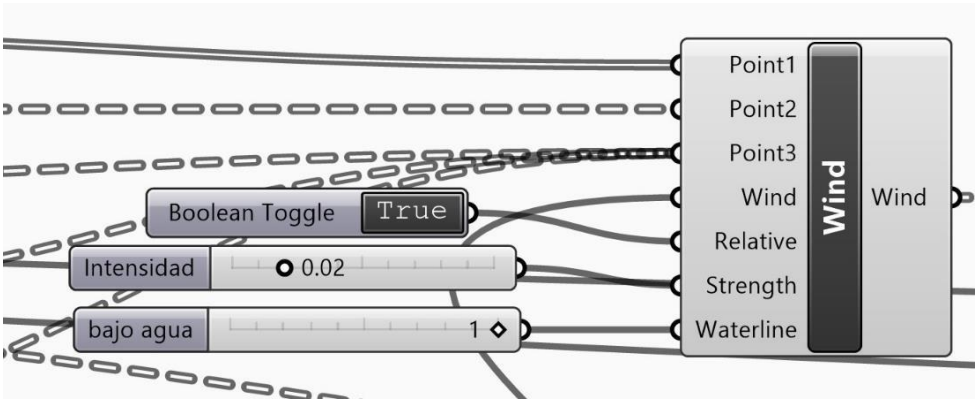


Fig. 24 Componente wind aplicado diferentes fuerzas. Fuente; Elaboración Propia

Grupo B- Generación de Fuerza vectorial del viento – corrientes marinas

En cuanto al componente de fuerza del viento (corrientes marinas), es crucial tener en cuenta un tensor vectorial que incluya tanto una magnitud como una dirección variable**. Es esencial abordarlo como un elemento externo al sistema a modificar, pero que actúe como el parámetro que influirá y cambiará la posición y configuración del sistema final mediante la fuerza del viento, es decir, su controlador**

Vector Fuerza Horizontal

Determinaremos un punto ajeno al exterior del sistema. El punto generador deberá estar ajeno al sistema de ramificación fractal inicial, es por ello por lo que se encuentra en un parámetro numérico a (-10) unidades con el fin de aplicar una fuerza de manera lineal y a 1 unidad del eje z. Este punto será inicio del vector generador de fuerza.

Una vez tengamos el punto inicial para introducir un vector de fuerza, la referencia del eje z nos indicará la altura desde la cual la fuerza afectará al sistema. En resumen, necesitamos que el punto inicial esté al nivel del sistema. Para lograr esto, el componente "move" (copia) con el punto de generación vectorial inicial nos permitirá restringir la amplitud del vector utilizando el eje z y un control deslizante de valor 5. (Fig.25)

Para establecer un control sobre la dirección y magnitud del vector de fuerza a incorporar, podemos utilizar el componente "circle". Esto nos permite, basándonos en un radio de amplitud y un punto referente en una circunferencia, relacionar el punto de inicio y el punto final con una magnitud y dirección determinadas mediante el control de rotación. Este proceso nos permite que las fuerzas de las corrientes marinas afecten al sistema en cualquier dirección deseada.

El componente "circle" se utiliza para establecer la dirección del vector de fuerza, basándose en un comportamiento radial. Por otro lado, el componente "Evaluate Curve" determinará el punto final de dirección del vector. Este punto final se define en referencia a la división de un punto (utilizando el componente "point") que interactúa con toda la circunferencia, utilizando el valor 1 como referencia.
El componente rotate (rotación) es el indicado para controlar la dirección del punto final y el punto a partir del cual va a rotar es el inicial. (Fig.26)
Al incorporar rotate necesitamos referirnos a la geometría (la entrada G) que vamos a rotar, en este caso, el punto generado por evaluate curve y representado por point. El uso de este componente

permite el uso de grados, en este caso 120 radianes que equivale a 360 grados. Al tener un comportamiento de rotación, representado por la circunferencia, es de considerar su plano origen, es por eso por lo que definimos el componente xy plane como plano de rotación, ya que es el mismo plano a partir del cual generamos la circunferencia de control. (Fig.27)

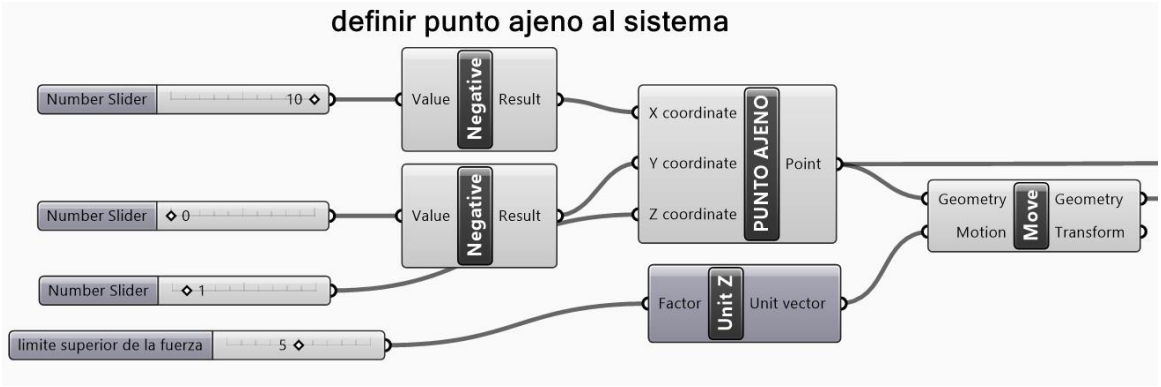


Fig.25 Definir punto ajeno al sistema. Fuente: Elaboración Propia

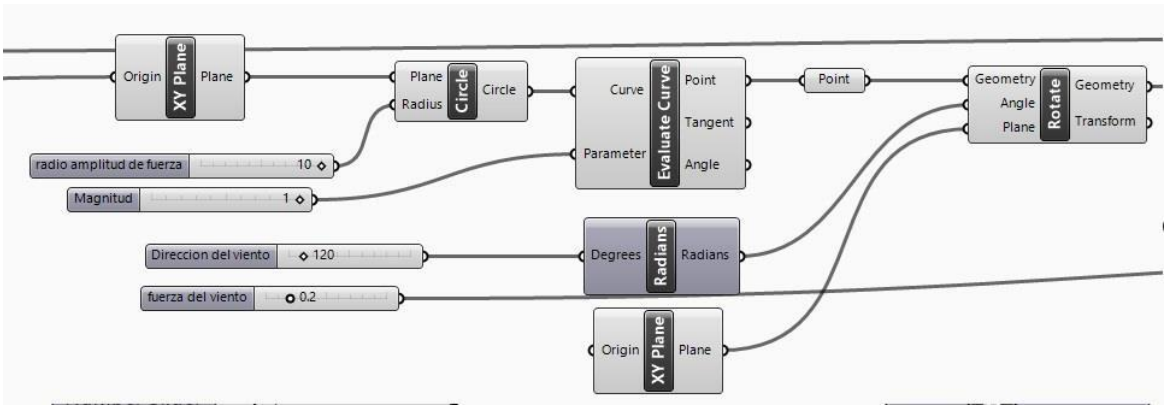
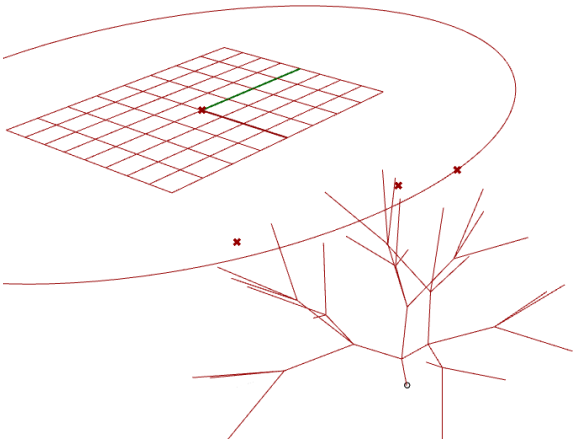


Fig. 26 Establece una dirección al vector fuerza: Fuente: Elaboración Propia



Necesitamos representar al valor resultante esto se obtiene a través de vector from two points: el punto inicial representa el inicio de la fuerza y el punto final, la dirección y en conjunto. A unir ambos puntos con las entradas de datos, hemos representado un vector que proporciona la fuerza de las mareas del agua (mar) afectando al sistema en conjunto.

Al obtener una representación vectorial de la fuerza y direcciones de las corrientes marinas afectaran y modificaran al sistema de ramificación, existe un problema referido a los datos que interpretara kangaroo – Wind . por lo que el componente amplitude nos ayudara. (Fig. 28)

Al relacionar los datos de salida vectoriales del componente "vector2Pt" con el parámetro de entrada

Fig. 27 Vector resultante que representa la fuerza de las corrientes marinas al aplicar al sistema. Fuente: Elaboración Propia

"V" del componente "amplitude", realizamos una interpretación de parámetros vectoriales numéricos que representan una magnitud de fuerza física. Estos parámetros necesitan tener un valor de amplitud. Con esto, y basándonos en los parámetros geométricos, numéricos y booleanos, hemos definido las corrientes marinas que afectarán al sistema.

El componente "wind" es una fuerza que debe ser simulada mediante un componente que traduzca los parámetros de control de una simulación computacional física, la cual impactará en el sistema geométrico de ramificaciones. (Fig.30)

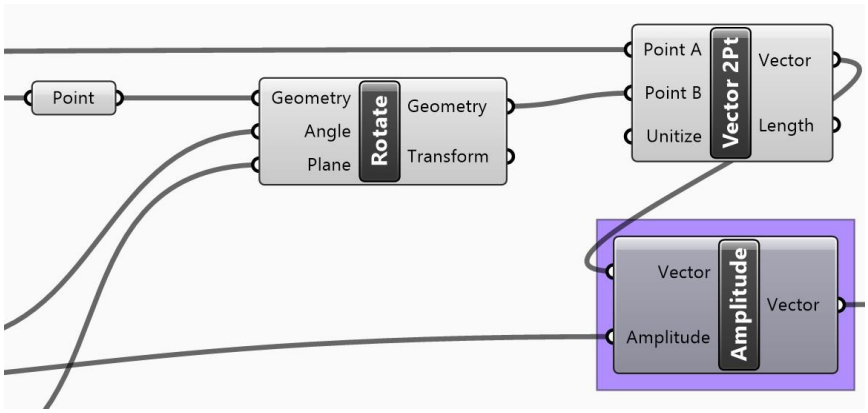


Fig. 28 Vector que proporciona la fuerza: Fuente: Elaboración Propia

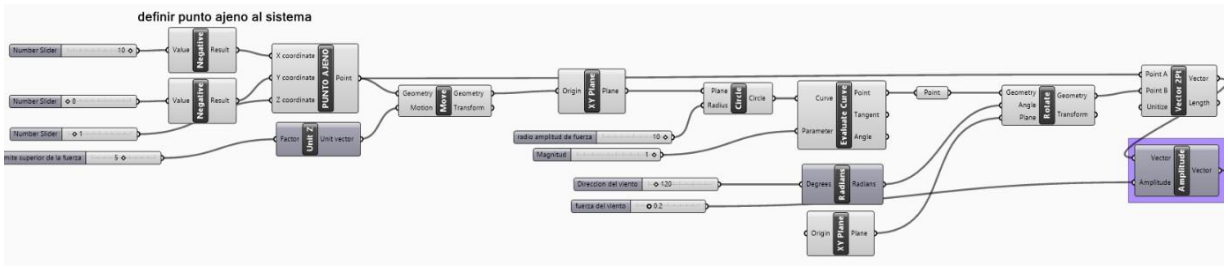


Fig. 29 Grupo B- Fuerza vectorial del viento, Ubicación del componente Amplitud. Fuente: Elaboración Propia

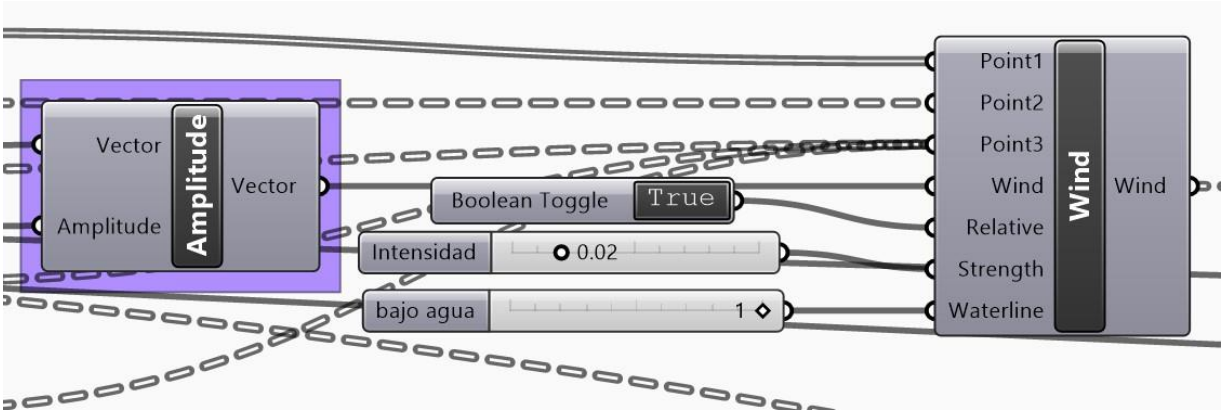


Fig. 30 Componente wind, simulación computacional. Fuente: Elaboración Propia

Grupo C Generación de Fuerza lineal – elástica en las ramificaciones

En el grupo anterior, caracterizamos la influencia de las corrientes marinas utilizando el modelo Kangaroo Wind en los puntos de cada iteración de la generación de ramificaciones. Ahora, necesitamos extender esta caracterización a las ramas del sistema, es decir, a las líneas que componen el mismo. Por lo tanto, nuestro enfoque se centrará exclusivamente en las líneas, dejando de lado los puntos.

Es importante considerar que, al trabajar en líneas, este sistema está conformado por puntos que dan lugar a las líneas. En este caso necesitaremos utilizar un componente que nos ayude a excluir líneas y puntos denominados Explode Curve (explotar la curva).

Al aplicar el componente Explode Curve a partir de las ramificaciones del ciclo de generación final (Ramificaciones Finales), podremos separar el sistema geométrico en vértices (puntos) y segmentos (líneas) del organismo digital. Luego, utilizaremos el componente Curve para conectar exclusivamente las líneas de este sistema, sobre las cuales aplicaremos la fuerza de las corrientes marinas mediante el componente Kangaroo - Springs (elasticidad). (Fig.30)

Ahora necesitamos conocer la Dimensión de cada uno de los ciclos de ramificaciones con el fin de establecer un límite máximo de fuerza de elasticidad a aplicar mediante un parámetro de multiplicación matemático y un número que permita asignar la magnitud limite correspondiente.

El componente Length curve nos ayudara a conocer la dimensión de cada iteración de las líneas de generación. incorporando a su vez un panel para ver datos. El panel calcula cada iteración de líneas.es de considera 1R es interpretado como el punto de generación inicial. Se representa en el siguiente esquema.

Va a ser multiplicado por multiplication y al añadir un parámetro numérico 1.25, obtendremos la longitud máxima (en rojo). (Fig.31)

El último paso, la salida del componente curve ramas y la entrada del componente Springs – Connection , con esto tenemos la fuerza aplicar a al objeto al que se le aplicara tal fuerza. Además, hemos definido el rest length (la longitud máxima de modificación en cada linea según la iteración generadora, Tenemos ahora que definir la fuerza de rigidez que tendrá las líneas para ello utilizaremos un parámetro numérico 0.1, con el fin de que las ramas no sean tan rígidas, La entrada Damping (amortiguación) será incorporado un parámetro numérico de 10, valor que nos permite disminuir la fuerza al aplicar a las ramas. (Fig.32) (Fig.33)

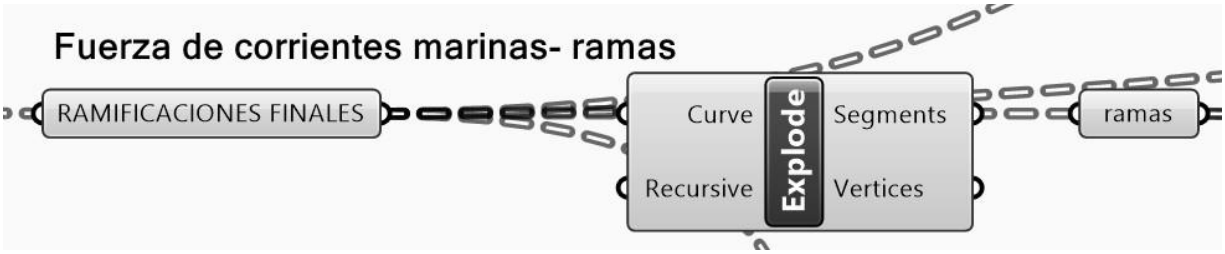


Fig. 30. Separación de vértices y segmentos del organismo digital. Fuente: Elaboración Propia

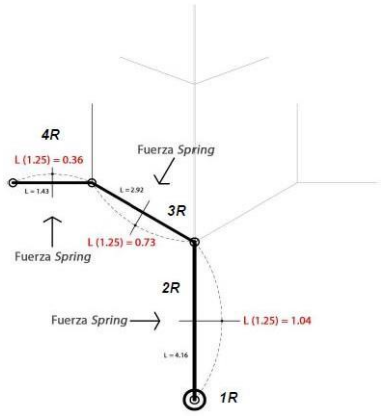


Fig. 31 Longitud máxima de transformación a cada interacción.
Fuente: Alan Rodríguez

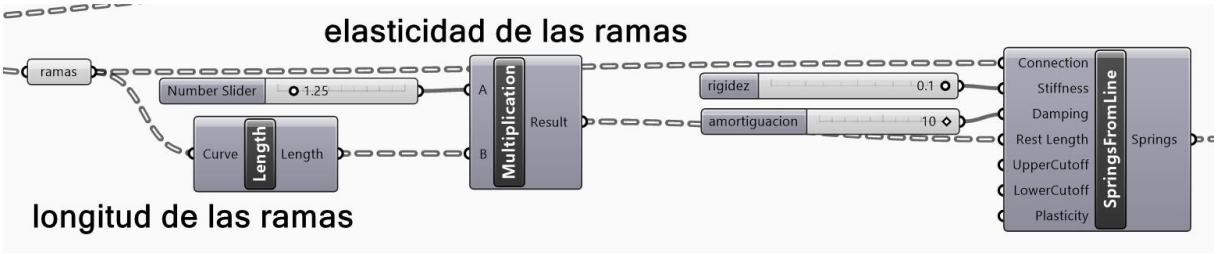


Fig. 32. Componente Springs, Fuerza simulada en Kangaroo. Fuente: Elaboración Propia

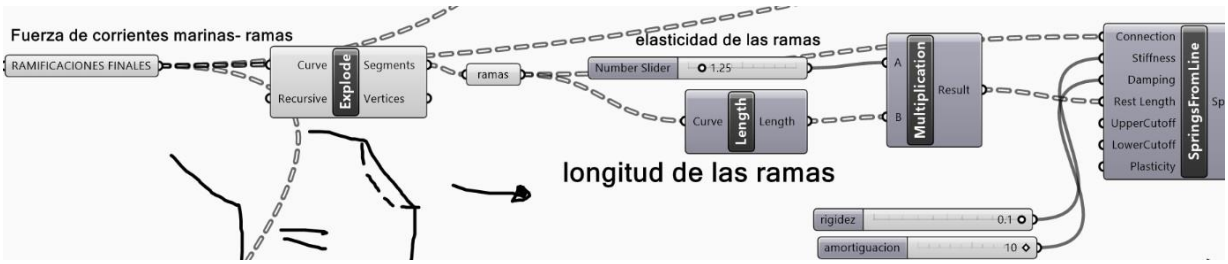


Fig. 33 Grupo C, generación de la fuerza lineal. Fuente: Propia

Grupo D Presión del mar

Es imprescindible comprender cómo la gravedad afecta la superficie terrestre. En el caso del agua, esta influencia varía dependiendo de la profundidad; no es igual estar cerca de la superficie que sumergido a 1 metro o a 300 metros de profundidad. En este escenario, optaremos por un valor numérico bajo, ya que el coral Acropora se encuentra generalmente a profundidades entre 5 y 25 metros.

A partir de Ramificaciones finales y el componente Evaluate curve, con la entrada t tendrá un valor numérico de 1. Luego obtendremos un segundo ciclo generador de a través de puntos, al cual será aplicada fuerza mediante el componente Kangaroo UnaryForce que simular a la presión ejercida por el organismo digital- natural. (Fig.34)

El componente Point (Puntos de Gravedad) en conjunto con Kangaroo UnaryForce generará una fuerza capaz de alterar la posición del objeto. La entrada del componente Force-UForce requiere la especificación de una magnitud, la cual se define mediante un parámetro numérico de 0.02, correspondiente al valor de la fuerza a aplicar. Este valor se ajusta para representar una presión baja, adecuada para la poca profundidad en el mar en este caso de estudio.
Es crucial destacar que la simulación de la presión del agua en el organismo solo considera un eje de aplicación de la fuerza, ya que de lo contrario estaría sujeto a múltiples fuerzas. (Fig.35)

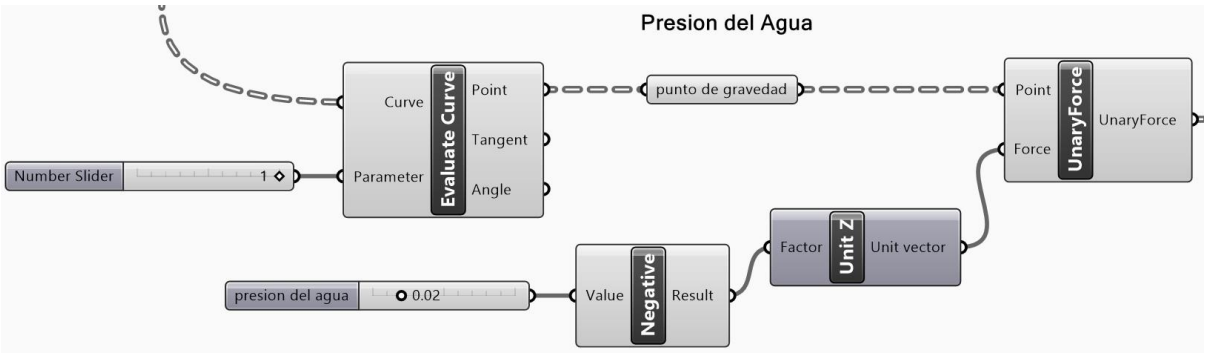


Fig. 34 UnaryForce y los vectores de aplicación de la fuerza de presión marina. Fuente; Elaboracion Propia

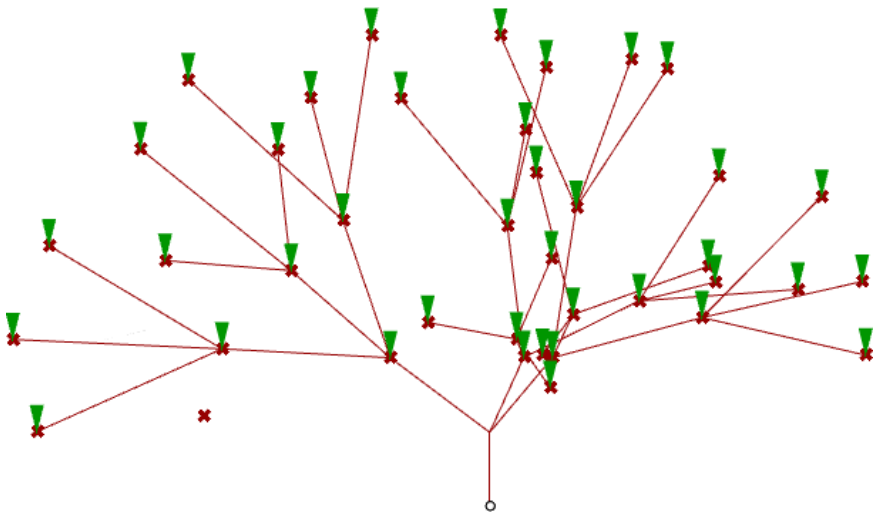


Fig. 35 La aplicación de la fuerza de presión del agua. Fuente. Elaboración Propia

Simulación Física – Kangaroo Physics

Esta variación formal y matemática se implementará mediante el plugin Kangaroo Physics en Grasshopper. Al abordar cada etapa de generación de fuerzas y elementos afectados, hemos logrado distribuir estas fuerzas a cada componente geométrico de la generación inicial. Esto se verá reflejado en una simulación física que producirá resultados variables en la forma. En resumen, el esquema inicial de la simulación se puede describir de la siguiente manera.

En resumen se obtiene (Punto de Anclaje + Wind + Springs + Unary Force) estos deberán conectarse a la entrada de Force Objects (Objetos de Fuerza) en este parámetro de entrada debería calcular los elementos en forma separada, por eso se coloca la condición Flatten. Por otra parte, el componente curve llamado Ramificaciones Finales está conectado a la entrada de Geometry. La relación del botón de reinicio (Toogle) nos permitirá realizar el cálculo una y otra vez del inicio; cuando la condición booleana sea verdadera (True) la simulación estará funcionando. (Fig.36) (Fig.37)

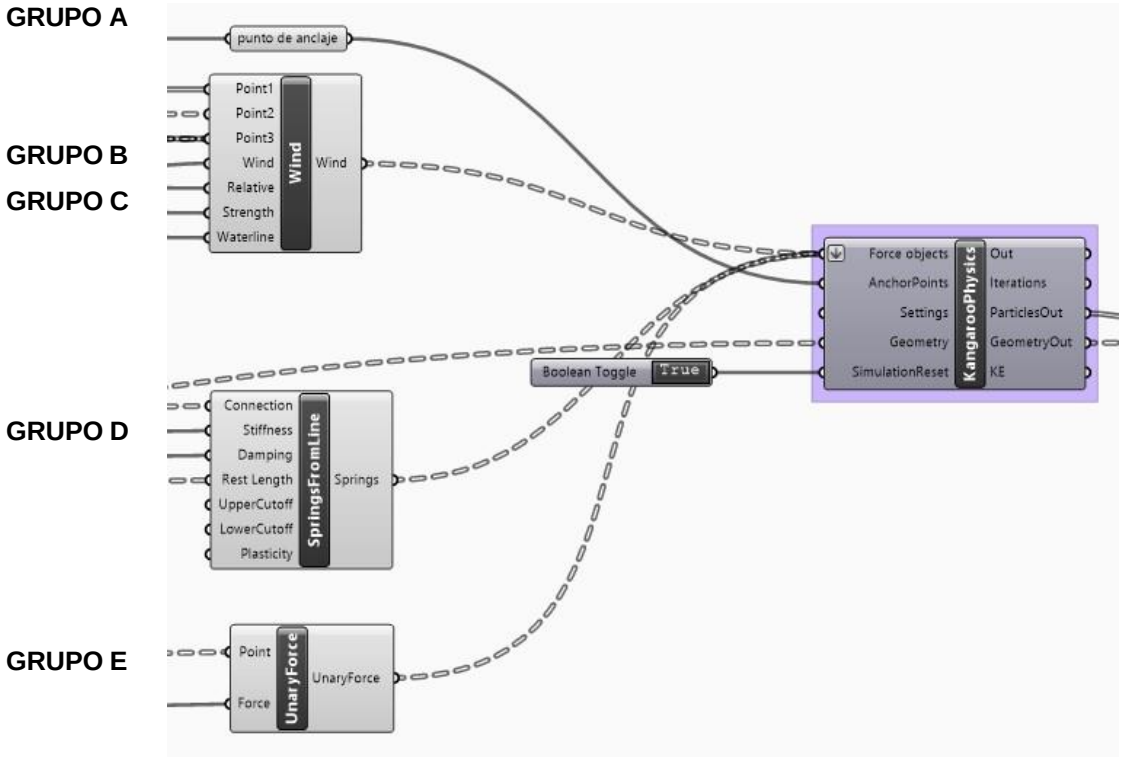
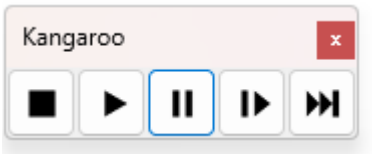
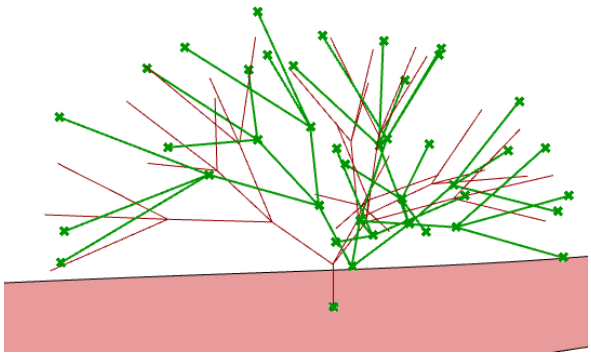
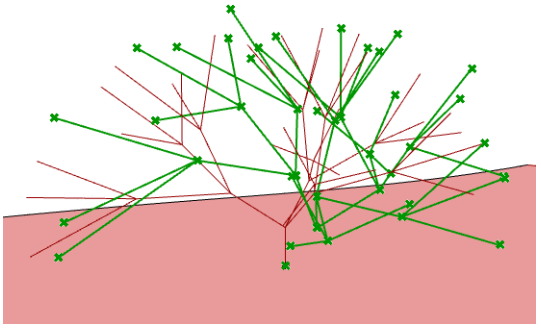


Fig. 36 Fuerza para aplicar e inicio de simulación del sistema primario de ramificación. Fuente: Elaboración Propia

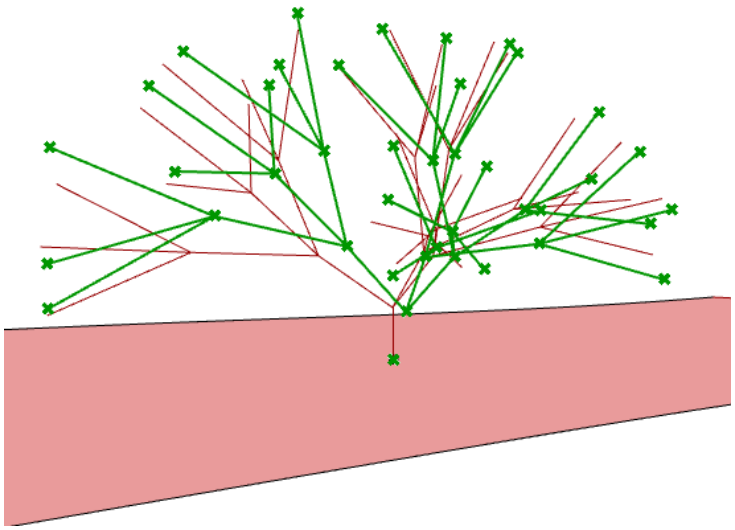






A = + 14S

A = +36 S



A = + 6 S

Fig. 37 Variabilidad geométrica a partir de la simulación física, basada en un algoritmo de programación grafica grasshopper. Fuente: Elaboración Propia

3.1.5 Tercera Etapa de Diseño

En la fase previa, hemos establecido un procedimiento de simulación mediante la aplicación de fuerzas físicas a un objeto específico, cuyos efectos se reflejan en una geometría variable de puntos y líneas. Los resultados obtenidos en esta simulación servirán como punto de partida para generar una geometría que responda al comportamiento solar, con base en el principio de maximizar o minimizar la cantidad de radiación absorbida por el organismo. (Fig. 38)

En el contexto de la formación de la estructura a través del material del organismo (Carbonato de Calcio), hay diversas variables que influyen en la generación de la materia y que determinan su forma final. Una de estas variables es el movimiento y la cantidad de energía que emite el sol y que es absorbida por el organismo. Este concepto es crucial, ya que delimita y establece la cantidad de materia que el organismo acumula, siendo mayor la cantidad de materia absorbida con mayor energía solar.
En este primer conjunto de acciones, nos enfocaremos en establecer un punto de referencia que simula el movimiento de un tractor a lo largo de una circunferencia, representando así el sol y su trayectoria. También estableceremos una relación de la geometría final resultante de la simulación física.

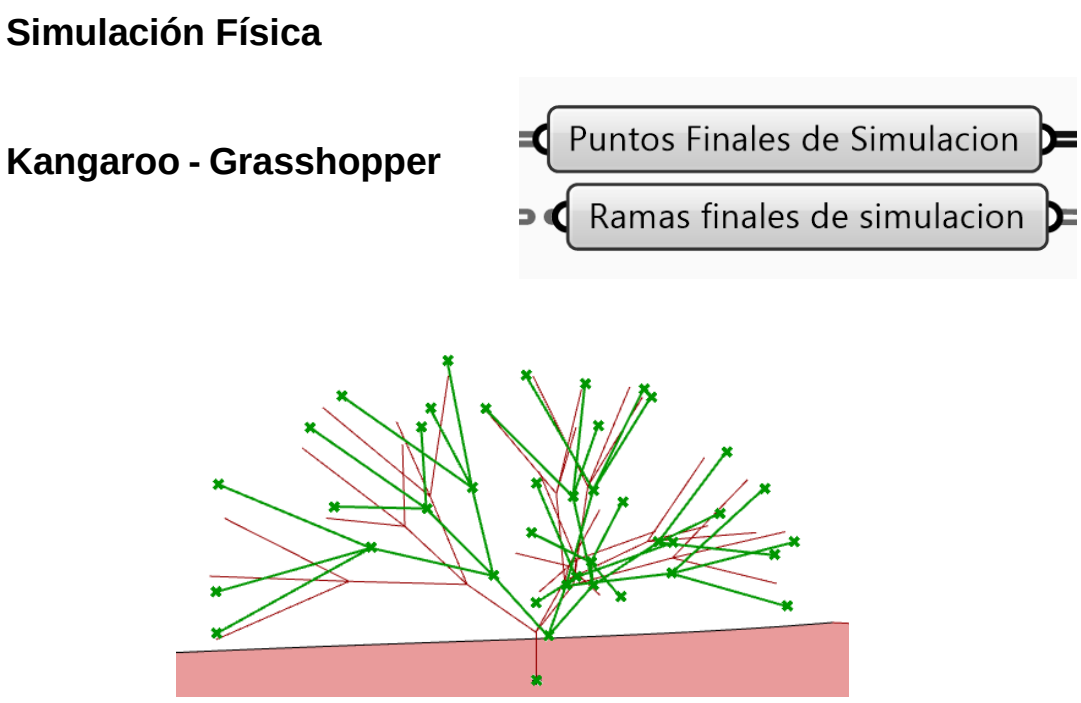


Fig. 38 Geometría representativa del esquema anterior en el segundo 6. Punto de inicio para la generación de dato, geometría variable, según el modificador del sol

Grupo A – Camino y Movimiento del Sol

Para generar geometría utilizando un objeto atrayente, es fundamental definir inicialmente un límite y una ruta de movimiento. Este comportamiento se visualiza mediante una curva de 180 grados (medio círculo), que se crea utilizando la herramienta "Arc" en Rhinoceros. Una vez que se ha creado el arco (representando la trayectoria del sol), este debe envolver completamente el organismo digital para garantizar una relación adecuada entre el componente modificador (sol) y los elementos afectados (puntos y líneas).

Ubicaremos un punto en la circunferencia que servirá como posición del sol. Para encontrar un punto en la curva, emplearemos el componente "Evaluate Curve", el cual nos permitirá establecer un punto en relación con su trayectoria (en este caso, la trayectoria del sol). Mediante la entrada "t", que tendrá un parámetro numérico de 60, equivalente a 180 grados de rotación a lo largo de la trayectoria del sol, determinaremos la posición del punto. Para representar al sol, que se genera a partir del punto en movimiento de la curva, utilizaremos el componente "Sphere" de Grasshopper. (Fig.39)

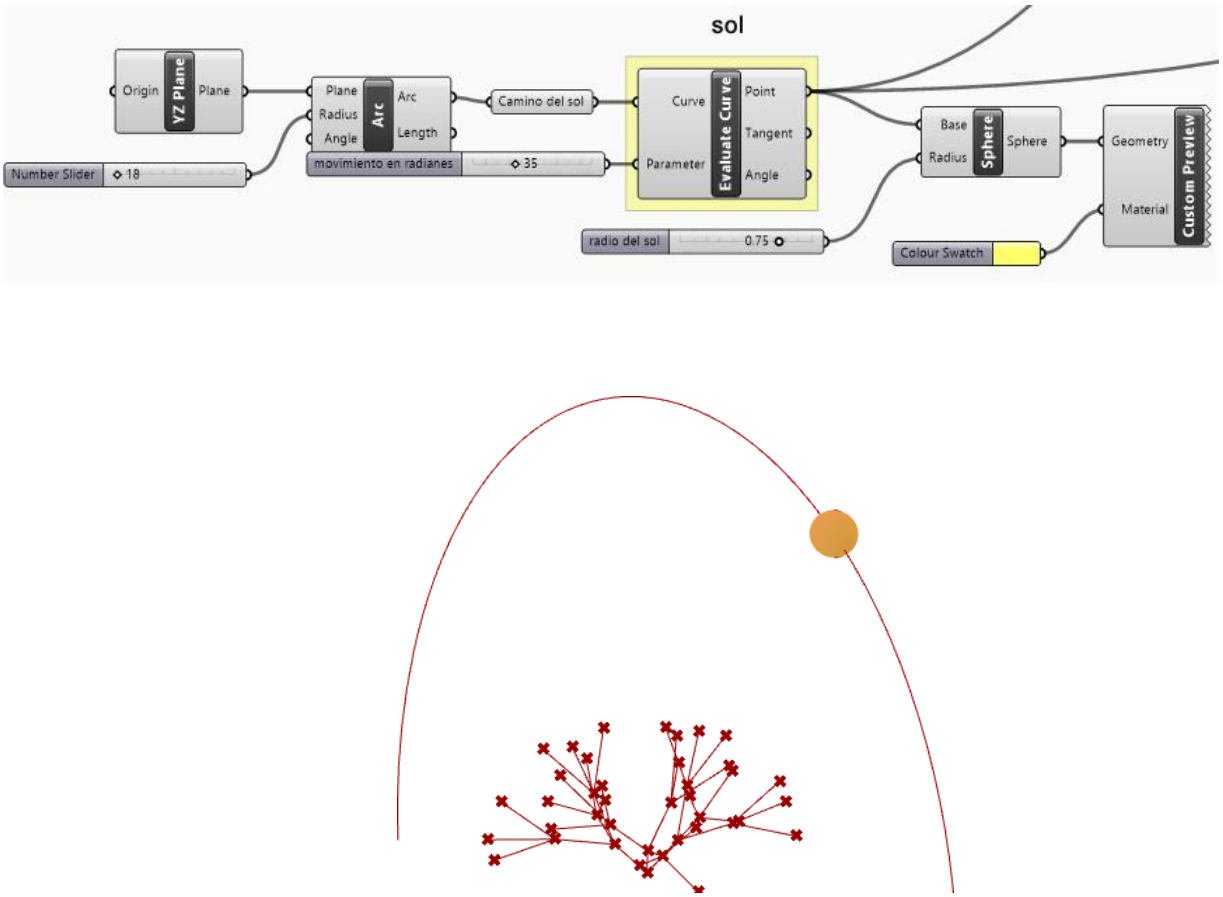


Fig. 39 Representación del sol, elemento externo generador y modificador de geometría

Grupo B – Amplitud de Exclusión Vectorial

Para poder realizar una exclusión de datos. Porque la superficie y el punto inicial generador deberán ser clonados, con el propósito que solo afecte a los Puntos Finales de Simulación Obtenidos mediante la Simulación Física de Kangaroo.

El componente "Curve" representa la superficie inicial y es una réplica de la superficie generadora inicial. El "Punto Atractor" es una instancia del "Punto Inicial Generador 3D", lo que nos permite establecer una relación vectorial en función del sol. Este "Punto Atractor Modificador" se utiliza para asegurar que siempre haya un cambio relativo al desplazamiento del sol, pero sin afectar la superficie en sí misma. Por esta razón, se incorpora el componente "Project Point".

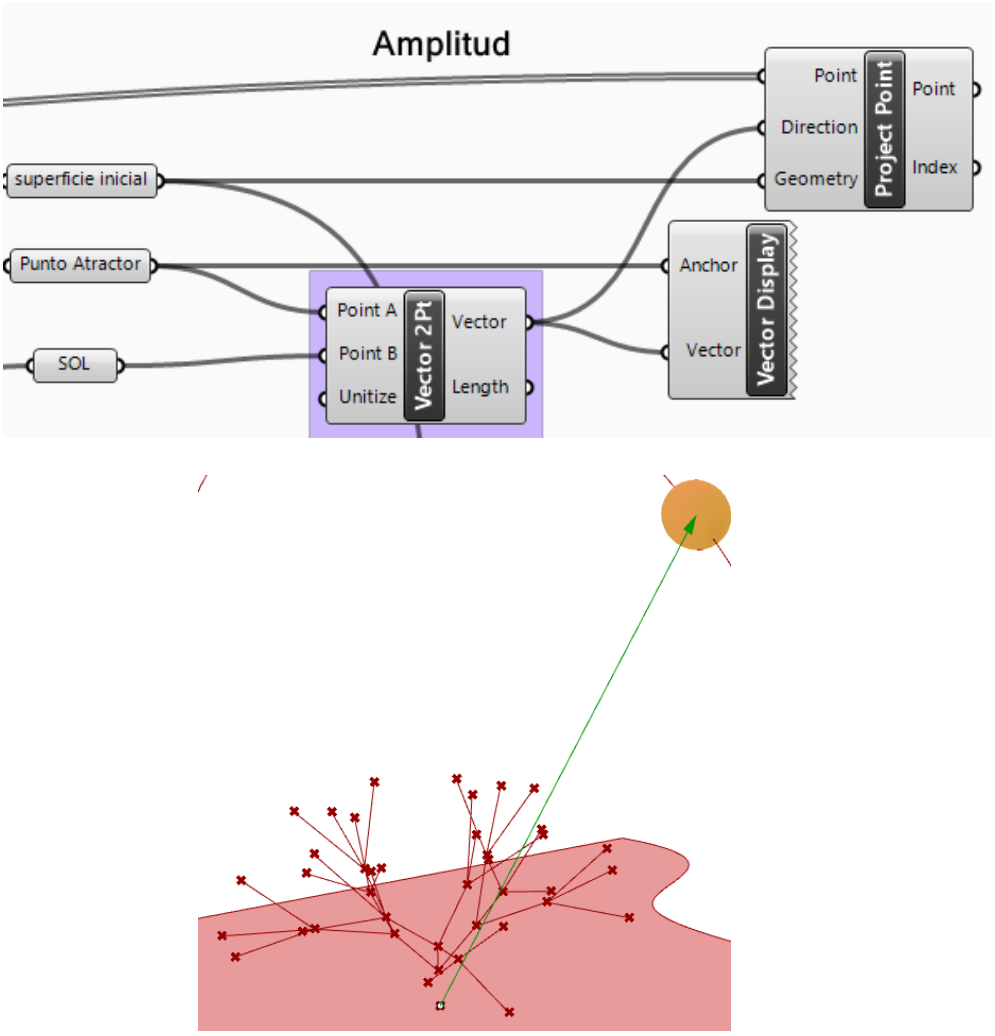


Fig. 40 Representación Geométrica Vectorial de elementos generadores iniciales. Fuente; Elaboración Propia

Grupo C – Eliminación de Geometría vectorial no -generadora

Exclusión Geométrica en Superficie – No crecimiento en Superficie

Al conectar la salida de datos de los "Puntos Finales de Simulación" con la entrada de datos "P" del componente "Point Brep", y la "Superficie de Exclusión Geométrica" con la entrada "B", se proyectan los puntos modificados por las fuerzas sobre la superficie. El componente "Dispatch" seleccionará aquellos puntos que generan o no generan geometría en relación con los puntos proyectados en la superficie. (Fig.41)

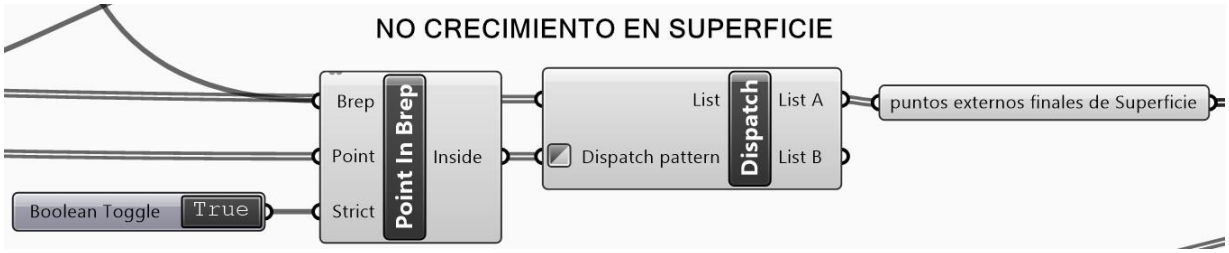


Fig. 41 Eliminación de la Geometría en la Superficie. Fuente: Elaboración Propia

Grupo D Producción Geométrica

Para desarrollar un dominio numérico variable de control de escala, es necesario establecer una relación entre el sol (Eval) y los Puntos Externos Finales de Superficie, que determinara el cambio de una escala mayor, cuando el sol se encuentre más próximo al sistema de ramificación generador, y una escala menor cuando el sol se encuentre más lejos.

El componente "Distance" facilita una relación de distancia entre el punto Sol y los "Puntos Externos Finales de Superficie", lo que nos permite limitar la variabilidad de la escala. Además, emplearemos "Bounds" para establecer el límite de distancia que varía según un parámetro numérico. El componente "Dominio" nos ayuda a definir límites máximos de escala mediante parámetros numéricos. Finalmente, para representar la variabilidad de la escala, incorporaremos el componente "Sphere" con el fin de visualizar el comportamiento y la representación formal final, así como para conservar las "Ramas Finales de Simulación". (Fig.42)

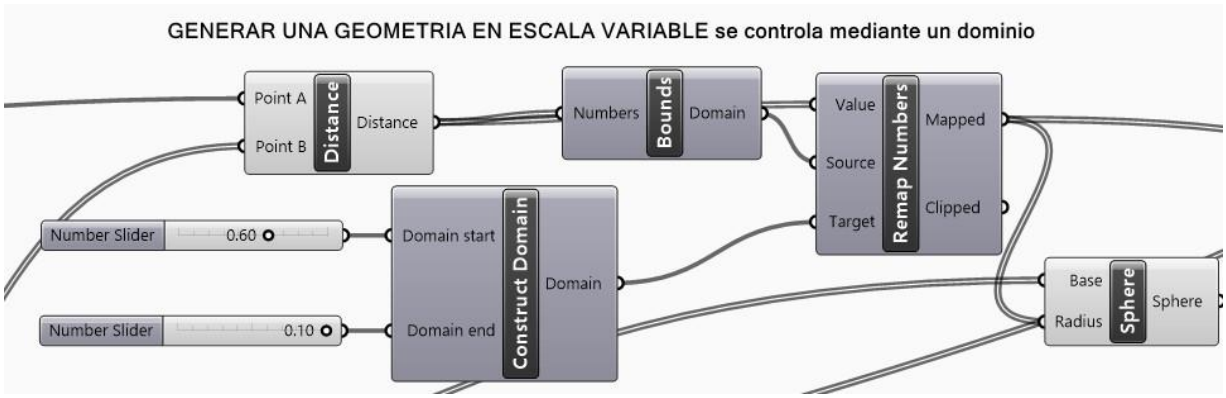
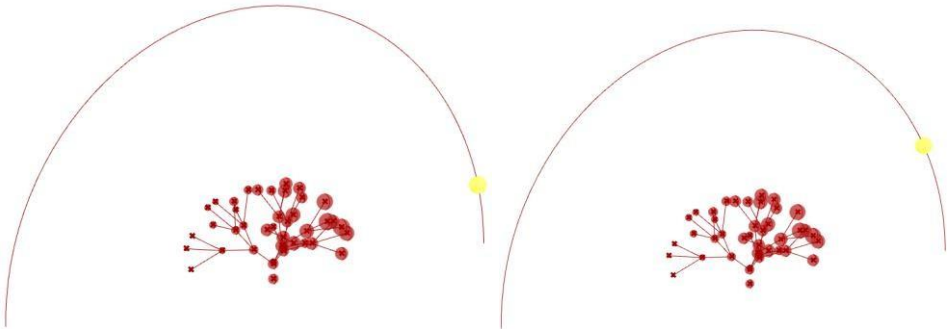
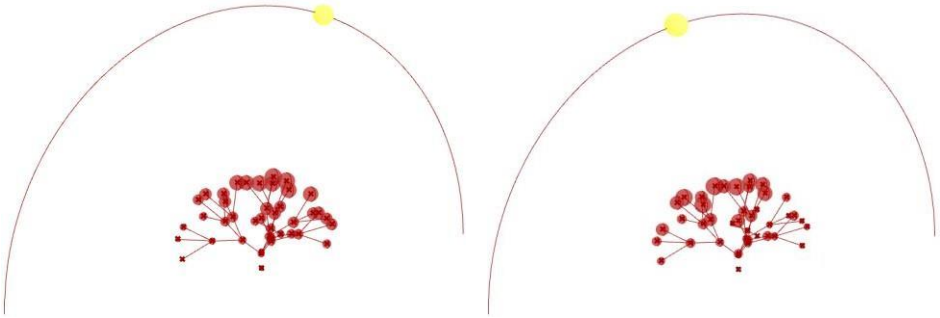


Fig. 42 Los componentes Dist, Bounds, Dom y Remap, generadores de la variación numérica en escala. Fuente: Elaboración Propia



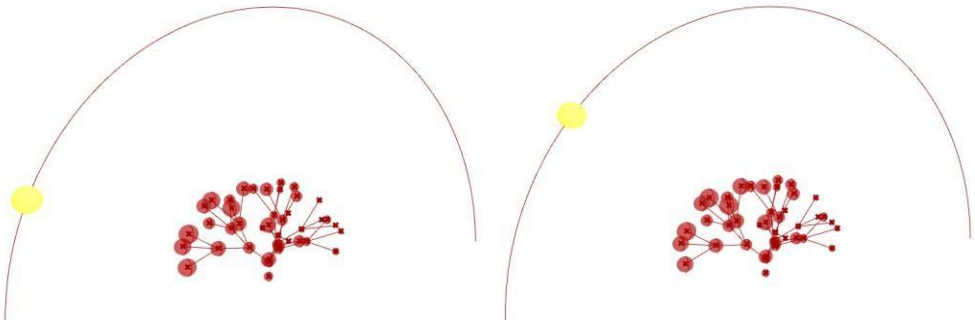
Sol = 5 π Rad

Sol = 10 π Rad



Sol = 25 π Rad

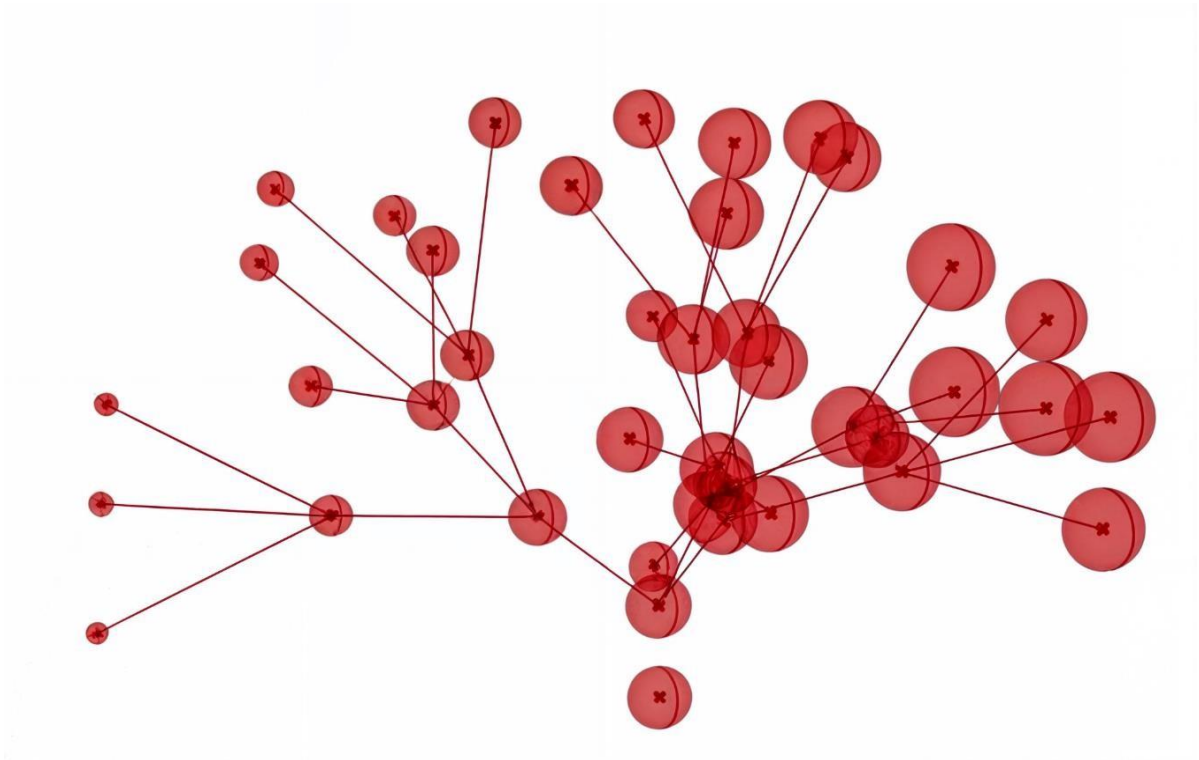
Sol = 35 π Rad



Sol = 45 π Rad

Sol = 50 π Rad

Fig. 43 variabilidad en escala de elementos geométricos finales del sistema + Etapa de Diseño.
Fuente: Elaboración Propia



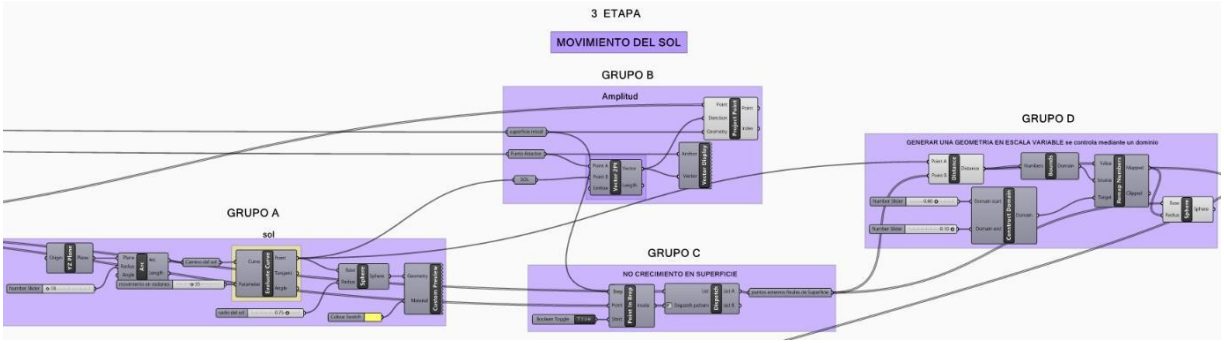


Fig. 44 Elemento generativo variable en escala, en función del desplazamiento y la posición del sol que afecta al sistema de ramificación fractal. Fuente: Elaboración Propia

3.1.6 Cuarta Etapa de Diseño

En esta fase final de diseño, vamos a representar digitalmente nuestro sistema mediante la incorporación de geometría en forma de mallas. Las mallas geométricas nos permitirán imitar la configuración formal final del organismo digital.

A partir de este momento nos limitaremos a generar únicamente la geometría de representación final, imitando la configuración morfológica del organismo natural a través de un algoritmo de programación grafica (Grasshopper) y un plugin extra cocoon, que nos permita optimizar la geometría final.

Transformación y Optimización de Ramificaciones Finales

Grupo A - Generar Mallas

Emplearemos el componente "Divide Curve", el cual nos posibilita dividir las líneas en tres partes mediante la entrada "N" con un valor numérico de 3. A partir de estos puntos generados, derivados de las líneas finales simuladas, podremos crear curvas para el desarrollo de las mallas geométricas finales.

Necesitamos convertir estas curvas en polilíneas utilizando el componente "Polyline", que estará activado con la opción "Flatten" representada por una flecha hacia abajo. El componente "Curve Charge", que es la curva de carga, nos permitirá relacionar las directrices, es decir, las ramas finales que ahora son polilíneas, las cuales deben conectarse al componente "Remap". El "Curve Charge" se conectará a la entrada "C" del componente "Cocoon", y restringiremos la lectura de datos a un elemento unitario, por lo que activaremos la condición "flatten". (Fig, 45)

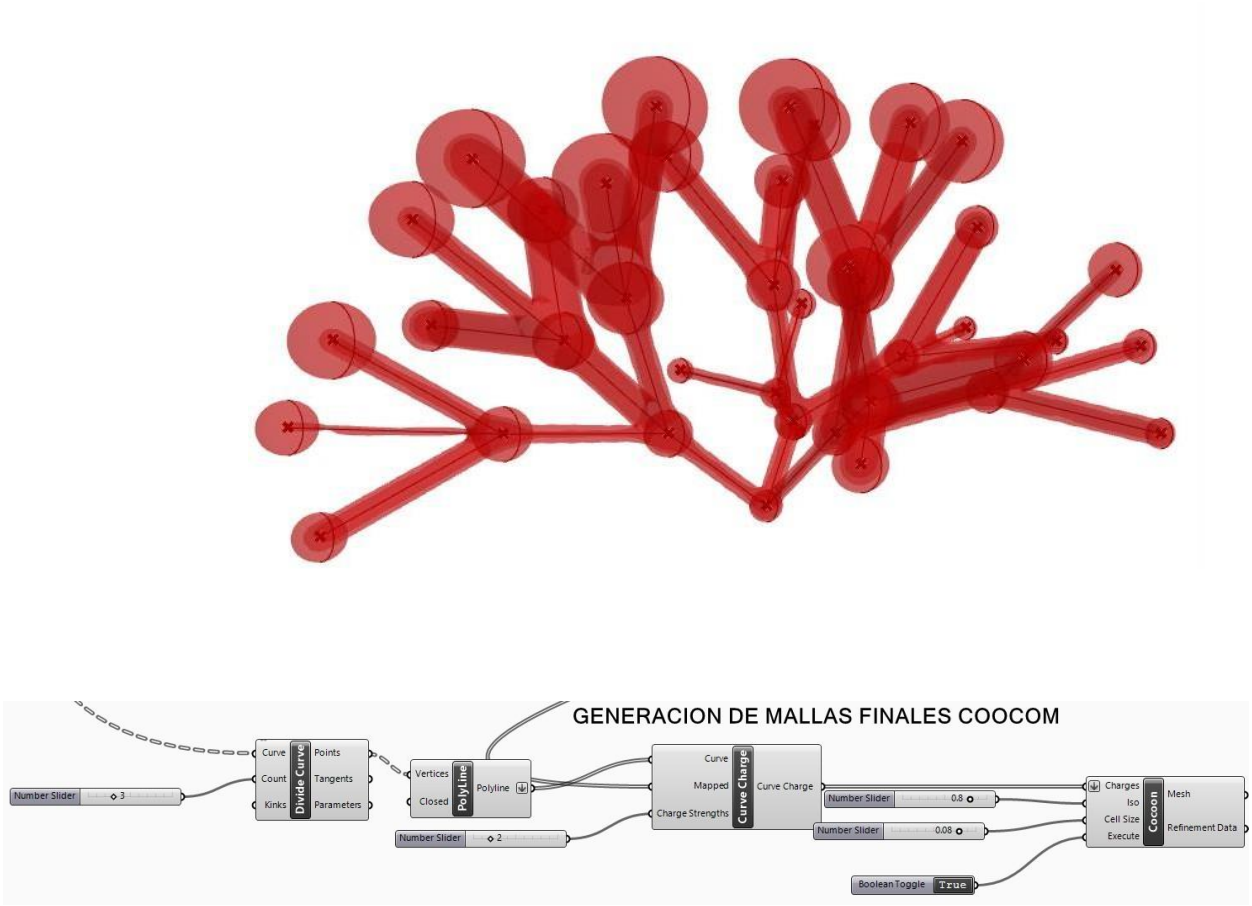


Fig. 45 Generador primario de mallas geométricas.

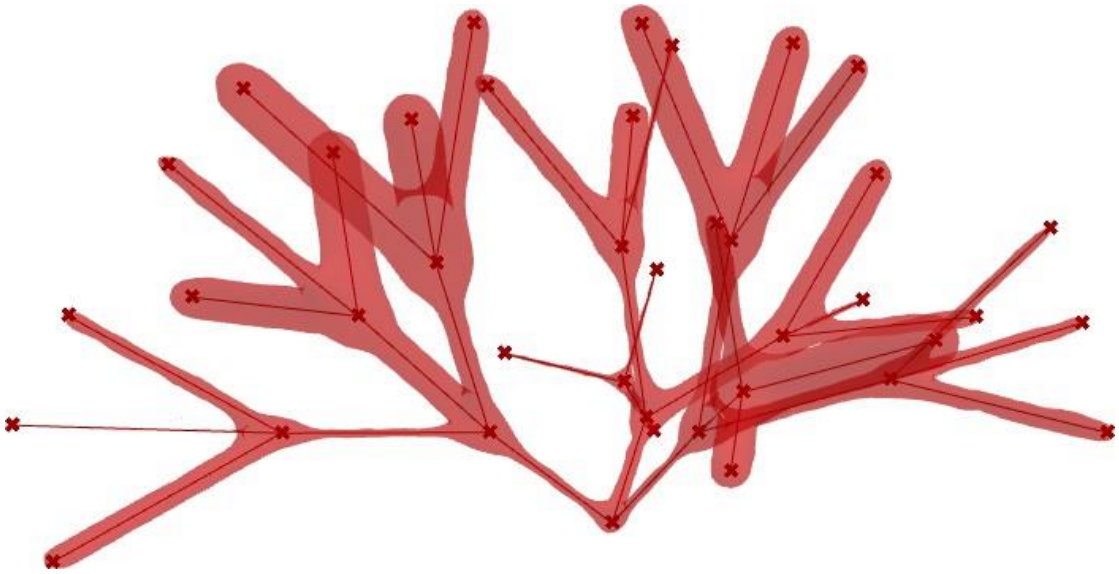
Optimización Final de Geometría

La optimización referida a la generación de geometría es un proceso que nos permitirá producir mallas más complejas con relación a sus divisiones.

Mallas Finales de Optimización.

Para optimizar la generación primaria de las mallas geométricas, emplearemos el componente "Refine" de Cocoon en Grasshopper.

Hemos mejorado la geometría de las mallas finales utilizando el componente Cocoon, lo que nos brinda un mayor control y una calidad superior en la resolución y presentación del modelado en 3D. Estas mallas finales se generan a partir de los primeros ciclos de ramificación (utilizando Anemone), la aplicación de fuerzas para modificar el sistema (con Kangaroo), la variación de escala (cantidad de materia) con relación al movimiento del sol (mediante Grasshopper + Rhino), y la optimización de la geometría a través de Cocoon. Hemos analizado e interpretado algunos de los comportamientos y procesos generativos relacionados con la forma de un sistema orgánico natural. (Fig.46)



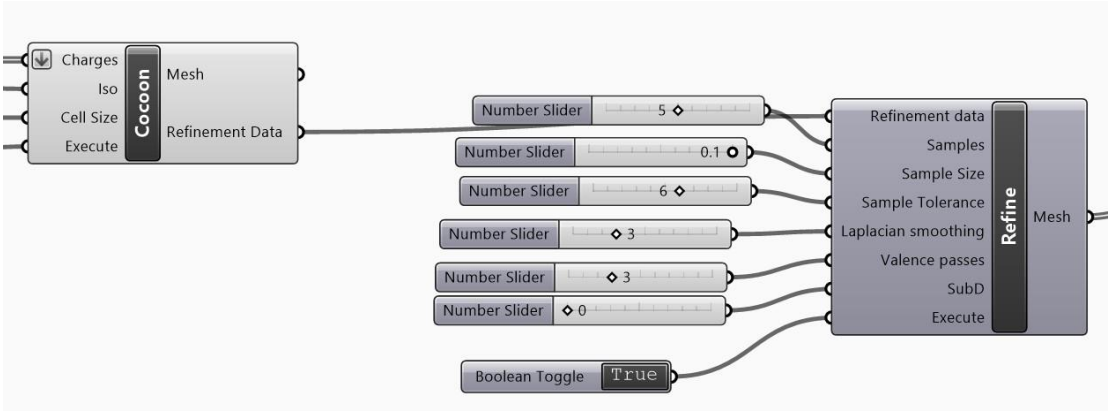


Fig. 46 componente Refine, para generar las mallas secundarias finales optimizadas.

Fuente: Elaboración Propia

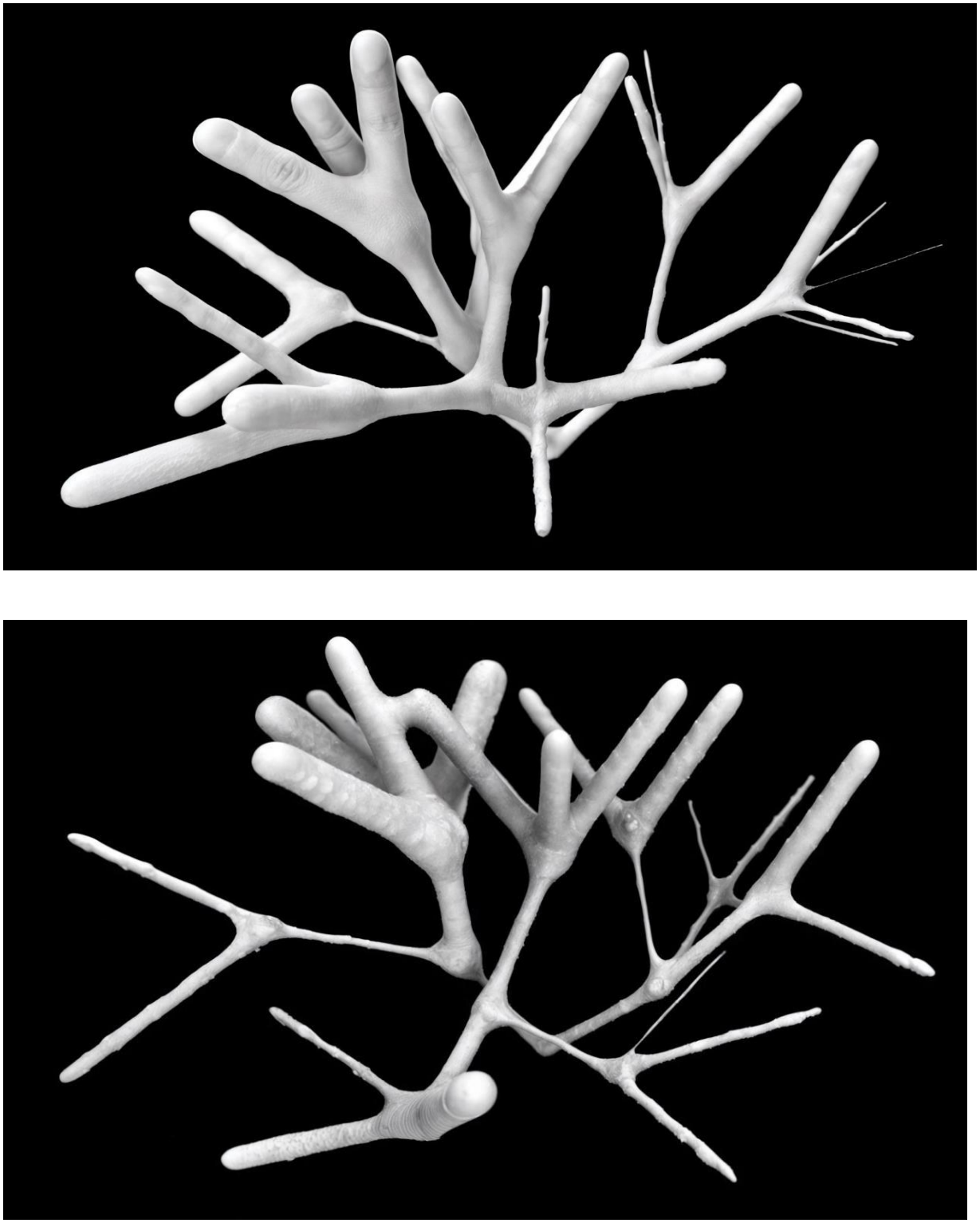
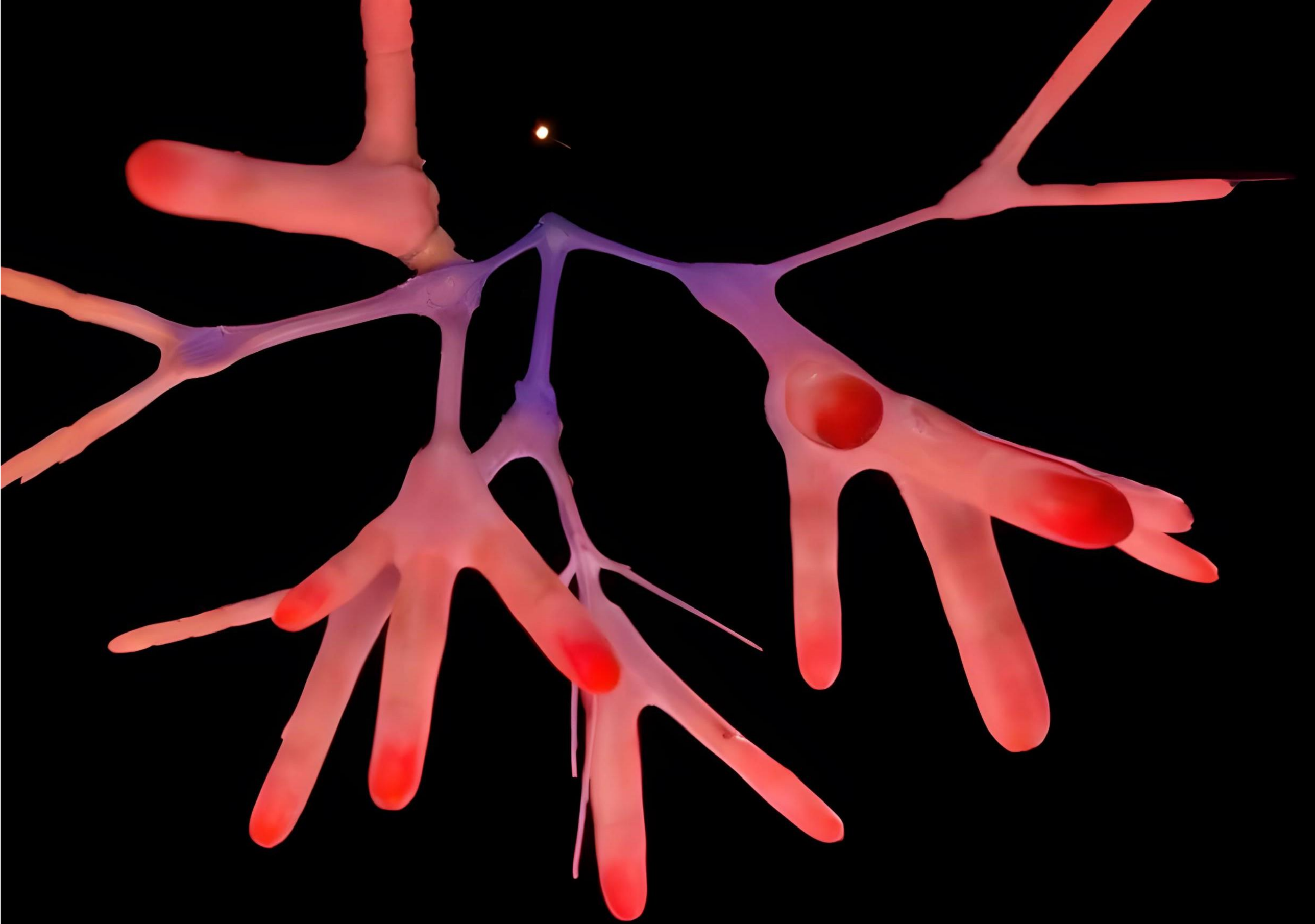


Fig. 47 Figura interacción a los 14s y 36s . Fuente: Elaboración propia



SISTEMA DE CRECIMIENTO DIFERENCIAL

3.2 Crecimiento Diferencial

Para investigar y crear un modelo relacionado con el crecimiento diferencial, se deberá estudiar como los organismos crecen y desarrollan sus estructuras. Esto incluye la comprensión de la división celular, la diferenciación celular, y la morfogénesis. Se Comprende cómo las fuerzas físicas, como la tensión superficial y la presión interna, influyen en el crecimiento y la forma de los organismos.

En nuestro caso de estudio, se ha escogido al coral lechuga, prestando especial atención a cómo las diferentes partes del coral crecen y se expanden. También se estudiará cómo la coral lechuga crece de manera que evita colisiones entre sus propias ramas, utilizando el espacio disponible de manera eficiente.

3.2.1 Coral de lechuga (Lobophytum)

El Coral de Lechuga, perteneciente al género Lobophytum spp., es una fascinante especie de coral blando que se encuentra comúnmente en los arrecifes de coral tropicales. Este coral, conocido por su forma distintiva y su apariencia exuberante, despierta el interés tanto de científicos como de aficionados a la naturaleza debido a su belleza y su papel en los ecosistemas marinos.

La estructura del Coral de Lechuga se caracteriza por una base carnosa y rizosa, desde la cual emergen numerosos pólipos que forman una apariencia similar a una lechuga o col rizada. Según Jones y Smith (2018), estos pólipos pueden variar en color, desde tonos suaves y pastel hasta colores vibrantes y llamativos, lo que contribuye a la diversidad visual de los arrecifes coralinos.

Fig. 49 Coral de lechuga (Lobophytum) . Fuente: Stan Bysshes Gallery

Fig. 51 Modelado coral lechuga. Fuente: Elaboración Propia

3.2.2 Enfoques aplicados al Proceso de Diseño

Un algoritmo que simule este proceso de crecimiento diferencial, donde una línea se expande mientras intenta mantener una distancia constante consigo misma. Este tipo de algoritmo puede generar patrones visualmente interesantes y dinámicos que exhiben autoorganización y control intrínseco sobre su estructura.

Para implementar este algoritmo, serán los siguientes pasos:

1..- Definir la línea inicial: **RHINO + GRASSHOPPER**

2.- Definir el proceso de crecimiento: **GRASSHOPPER**

3.-Aplicar restricciones de distancia**: GRASSHOPPER**

4.-Actualizar la línea: **GRASSHOPPER**

5.-Iterar el proceso: **GRASSHOPPER + ANEM**

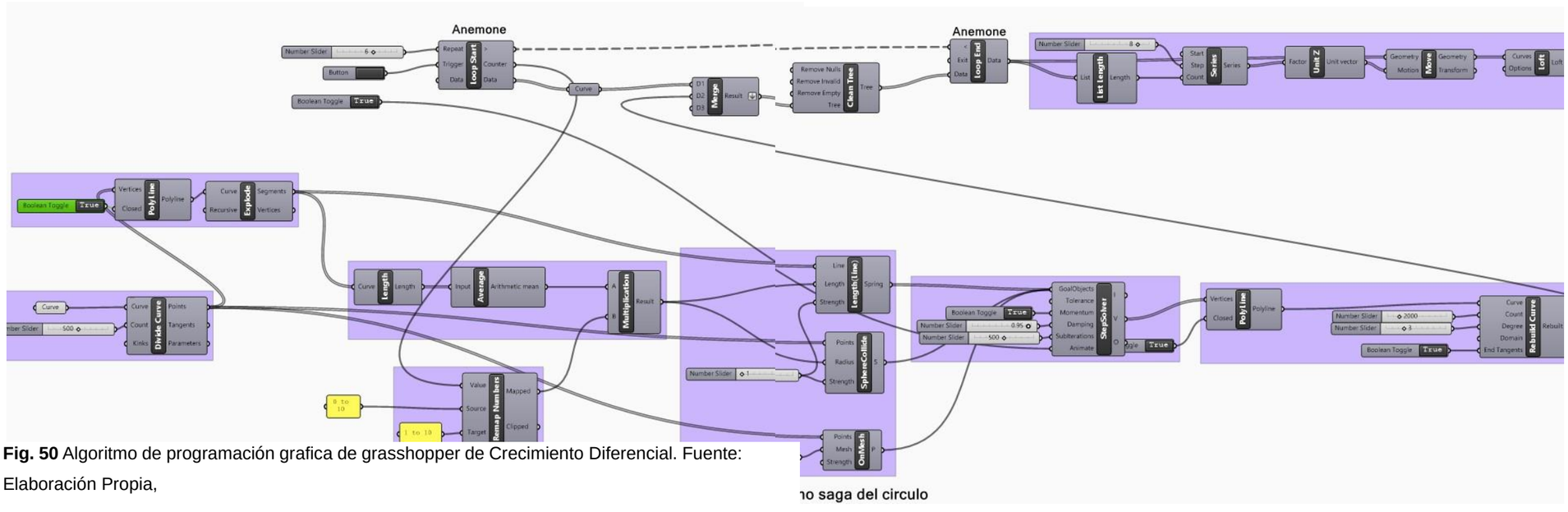


Fig. 50 Algoritmo de programación grafica de grasshopper de Crecimiento Diferencial. Fuente: Elaboración Propia,

3.2.3 Primera Etapa del Diseño

En el lienzo de Rhino, se dibujará una curva inicial que servirá como punto de partida para el crecimiento. Esta curva será referenciada en Grasshopper utilizando el comando "Curve" para manipularla y generar el crecimiento deseado. (Fig, 53)

Para dividir la curva en pequeñas partes, utilizaremos el componente "Divide Curve" en Grasshopper. Conectarás la entrada "C" de "Divide Curve" con el componente "Curve" que representa la curva inicial que dibujaste en Rhino. Luego, configurarás el parámetro "Count" en 500, lo que significa que la curva se dividirá en 500 segmentos**. Este proceso generará una serie de puntos a lo largo de la curva, cada uno representando un punto de división.** (Fig. 54)

En esta fase necesitamos simular la expansión de los segmentos líneas por lo que la curva será fragmentada.
Después de dividir la curva en segmentos utilizando el componente "Divide Curve", utilizaremos el componente "Polyline" para convertir estos segmentos en una polilínea. Es importante tener en cuenta que la polilínea resultante no estará cerrada inicialmente. Para cerrarla, utilizaremos un interruptor booleano que estableceremos como verdadero en la entrada "closed" del componente "Polyline".

Después de crear la polilínea cerrada utilizando el componente "Polyline", vamos a utilizar la herramienta Kangaroo para simular la expansión de los segmentos de línea. Para hacer esto, primero necesitaremos explotar la polilínea en sus segmentos individuales. Luego, utilizaremos el componente "Length (line)" para definir la longitud inicial de cada segmento de línea. (Fig. 55)
Para calcular la longitud promedio de los segmentos de línea de la polilínea y luego definir esa longitud promedio como la longitud inicial para todos los segmentos, seguiremos estos pasos:

Conectar la salida del componente Explode al componente "Length" para obtener las longitudes individuales de cada segmento de línea.
Utilizar el componente "Average" para calcular el promedio de todas estas longitudes.
Conectar la salida del componente "Average" al componente "Multiplication" junto con un parámetro numérico establecido en 1. (Fig.55)
Conectar la salida del componente "Multiplication" a la entrada "Length" del componente "Length (line)".

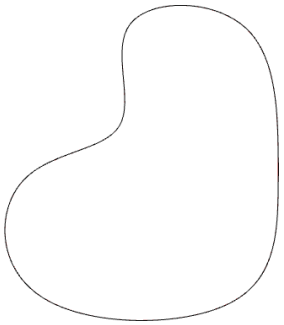


Fig.53 Curva de lienzo en Rhino. Fuente: Elaboración Propia

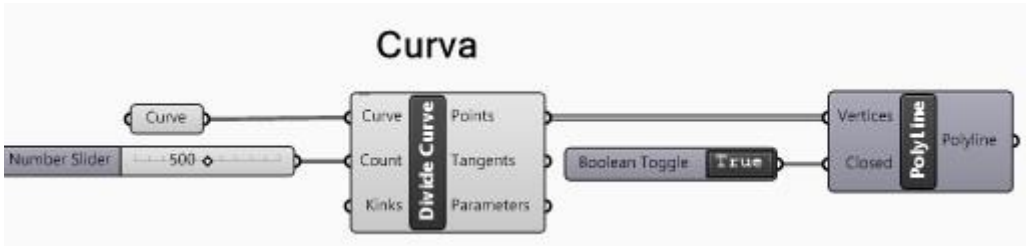


Fig. 54 Componente curve será divido en segmentos por Divide Curve. Fuente; Elaboración Propia

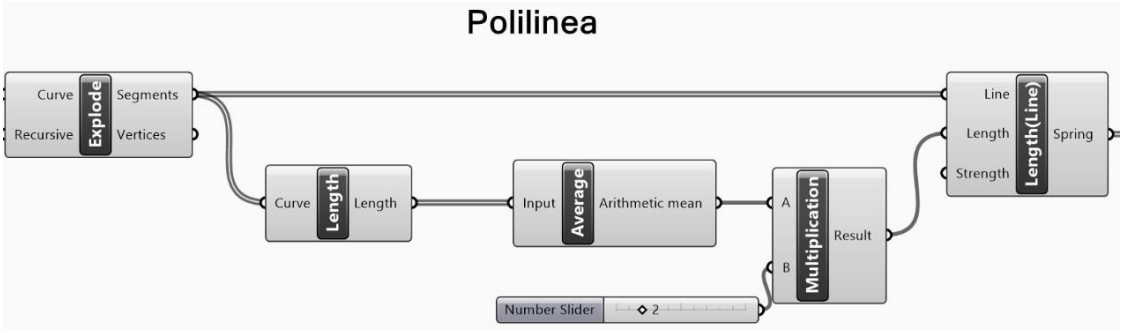


Fig. 55 definir la longitud inicial, se define la longitud promedio. Fuente: Elaboración Propia

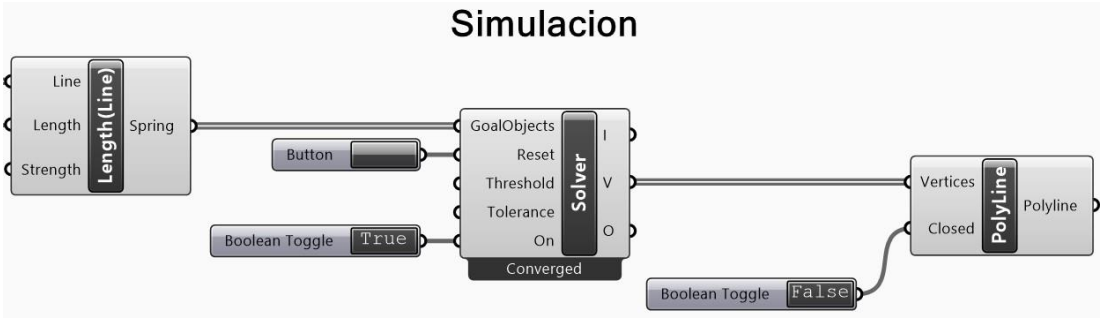


Fig. 56 Componente Solver de Kangaroo Simulación. Fuente: Elaboración Propia

Con estos pasos, estarás configurando la simulación en Kangaroo para que los segmentos de línea se expandan según las longitudes definidas inicialmente.

Conectar la salida "Length" del componente "Length (line)" a la entrada "GoalObjects" del componente "Solver".
Conectar el botón de reset o alternar al componente "Reset" del componente "Solver".
Conectar un interruptor booleano (Boolean Toggle) configurado como "True/False" a la entrada "On" del componente "Solver".
Utilizar la salida de los vértices del componente "Solver" como entrada para reconstruir una nueva polilínea. (Fig.56) (Fig. 57)
Para asegurar que la polilínea generada no esté cerrada, conectar el interruptor booleano (Boolean Toggle) a la entrada correspondiente del componente utilizado para reconstruir la polilínea.

Para evitar la colisión entre los puntos generados y garantizar que las líneas no se intercepten, utilizaremos el componente "SphereCollide" en Grasshopper. Aquí está la secuencia de pasos para configurar esto:
Conectar la salida de los puntos generados (posiblemente la salida de vértices del componente "Solver") al componente "Divide Curve". (Fig. 58)

Conectar la salida del componente "Divide Curve" al componente "SphereCollide" para proporcionar los puntos a este componente.
Utilizar el componente "Multiplication" para multiplicar el radio deseado por cada punto. Conectar la salida de este componente al parámetro "Radius" del componente "SphereCollide".
Conectar la salida del componente "SphereCollide" a la entrada "Goal Objects" del componente "Solver". (Fig.59)
De esta manera, estamos asegurando que cada punto generado esté rodeado por una esfera de radio específico, evitando que los puntos se crucen entre sí. Esto ayudará a mantener el orden y evitar la intersección de las líneas en la simulación.

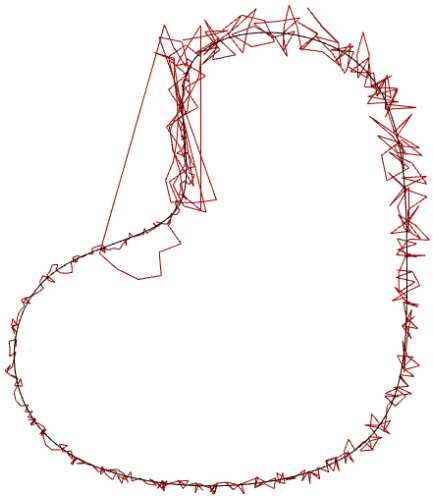


Fig. 57 líneas en expansión. Fuente: Elaboración Propia

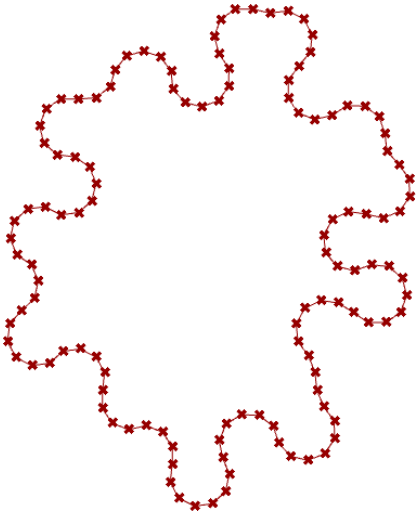


Fig. 58 Puntos generados sin la intercepción de estas. Fuente: Elaboración Propia

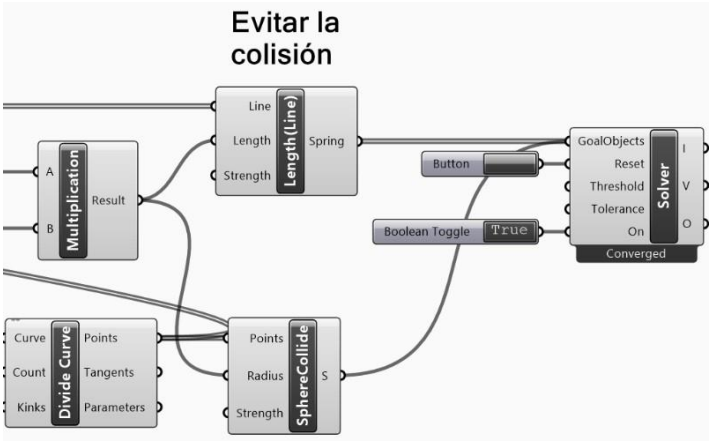


Fig. 59 Componente Sphere Collide. Fuente: Elaboración Propia

Al aumentar la fuerza del componente "Length (line)", estamos asegurando que la expansión de los segmentos de línea sea más pronunciada y tenga prioridad sobre la fuerza de las esferas de colisión, lo que ayudará a garantizar un mejor crecimiento en la simulación.

Para ajustar la fuerza del componente "Length (line)" y asegurarnos de que sea mayor que la fuerza del componente "SphereCollide", seguiremos estos pasos:
Conectar la salida del componente "Length (line)" al componente "Strength" del componente "Solver".
Configurar un parámetro numérico con el valor deseado (por ejemplo, 10) y conectarlo a la entrada "Strength" del componente "Solver". (Fig.60)

Para establecer una restricción circular en la expansión de las líneas generadas,
En el lienzo de Rhino, dibuja un círculo que servirá como límite para el crecimiento de las líneas.
En Grasshopper, utiliza el componente "Curve" para hacer referencia al círculo dibujado en Rhino.
Conecta la salida del componente "Curve" al componente "Divide Curve", para dividir el círculo en segmentos.
Utiliza el componente "Mesh" para generar una malla a partir del círculo. Conecta la salida de la malla al componente "OnMesh".
Conecta la salida de puntos del componente "Divide Curve" al componente "OnMesh". (Fig.61)
Conecta la salida del componente "OnMesh" a la entrada "Goal Objects" del componente "Solver".
De esta manera, estarás estableciendo una restricción circular en la expansión de las líneas generadas, asegurando que no crezcan más allá del límite definido por el círculo dibujado en Rhino. (Fig.62)

Para suavizar la curva resultante y mejorar su resolución, así como para ajustar las tangentes finales de la curva, podemos seguir estos pasos:
Conectar la salida de la polilínea al componente "Rebuild Curve". (Fig.63)
Configurar un parámetro numérico con el valor deseado (por ejemplo, 2000) y conectarlo a la entrada "Points" del componente "Rebuild Curve". Esto aumentará la resolución de la curva resultante.
Configurar otro parámetro numérico con el valor deseado (por ejemplo, 3) y conectarlo a la entrada "Degree" del componente "Rebuild Curve". Esto ajustará el grado de la curva, suavizándola.
Conectar un interruptor booleano (Boolean Toggle) a la entrada "End Tangents" del componente "Rebuild Curve". Este interruptor controlará si se ajustan las tangentes finales de la curva.
Configurar el interruptor booleano según sea necesario para ajustar las tangentes finales de la curva.

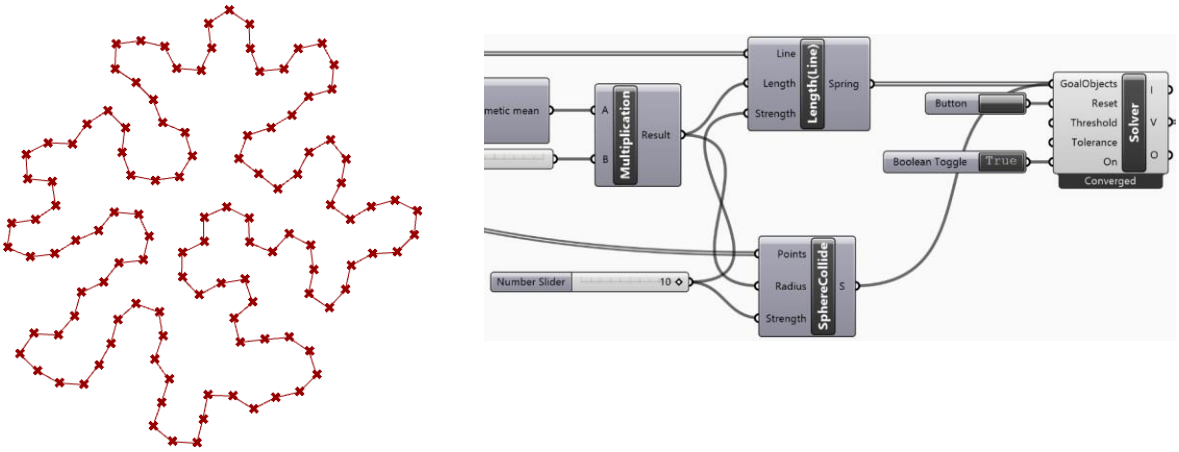


Fig. 60 Puntos generados sin la intercepción de estos. Fuente: Elaboración. Propia

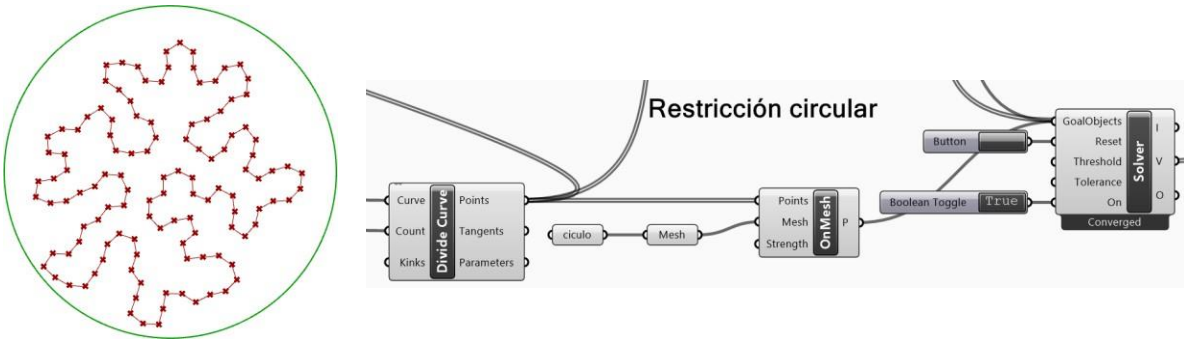


Fig. 61 Dibujado un circulo en el lienzo de Rhino + Componente On mesh como resultado georreferenciado el circulo. Elaboración: Propia

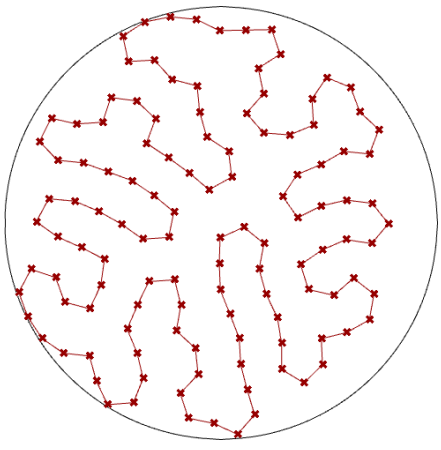


Fig. 62 Crecimiento Restringido. Fuente: Elaboración Propia

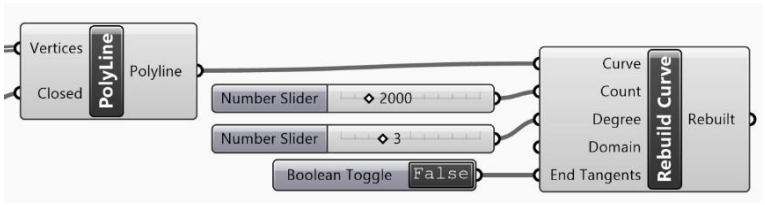


Fig. 63 Reconstruyendo la polilínea. Fuente: Elaboración Propia

Necesitamos un mejor control sobre la simulación, ya que en esta fase se aplicará anemone que hará un x número de repeticiones. Para cambiar el solver a Step Solver y conectar los componentes correspondientes, así como configurar los parámetros necesarios,
Reemplazar el componente "Solver" por el componente "Step Solver".
Conectar las salidas de los componentes "Length (Line)", "Sphere Collide" y "On Mesh" a la entrada "GoalObjects" del componente "Step Solver".
Conectar un interruptor booleano (Boolean Toggle) a la entrada "Momentum" del componente "Step Solver".
Configurar un parámetro numérico con el valor 0.95 y conectarlo a la entrada "Damping" del componente "Step Solver".
La entrada anímate se conectará con el componente toggle, al momento de cambiarlo a True estará animando correctamente. (Fig.64)

Se usará el componente anemone, lo que permitirá que el bucle se repita la cantidad de veces especificada.

Agrega el componente "Loop Start" al lienzo de Grasshopper.
Conecta un parámetro numérico con el valor 10 a la entrada "N" del componente "Loop Start". Esto determinará la cantidad de repeticiones del bucle.
Conecta un botón (Button) al componente "Loop Start" en la entrada "T" como disparador para iniciar el bucle.
Agrega el componente "Loop End" al lienzo de Grasshopper.
Conecta la salida "Do" del componente "Loop Start" a la entrada "Do" del componente "Loop End" para cerrar el bucle.
Conecta la salida "Counter" del componente "Loop Start" a la entrada "B" del componente "Multiplication". Esto permitirá que el bucle se repita la cantidad de veces especificada. (Fig. 65)

Para un mayor control sobre como utilizar el contador es decir Al remapear el valor del contador del bucle, puedes controlar el número de iteraciones y ajustar cómo cada iteración afecta la geometría final. Esto es especialmente útil cuando necesitas un control granular sobre cada paso del proceso de generación geométrica.

Para utilizar el componente "Remap" para tener un mayor control sobre el contador del bucle:
Agrega el componente "Remap" al lienzo de Grasshopper.
Conecta la salida "C" del componente "Loop Start" a la entrada "V" del componente "Remap". Esto proporcionará el valor del contador del bucle al componente "Remap".
Configura un parámetro numérico con el valor 10 y conéctalo a la entrada "S" del componente "Remap". Esto establecerá el rango de entrada deseado para el contador del bucle. (Fig.66)

Conecta la salida "S" del componente "Remap" al componente "Multiplication".

Con estos pasos, el componente "Remap" ajustará el valor del contador del bucle para que esté dentro del rango especificado antes de pasarlo al componente "Multiplication".

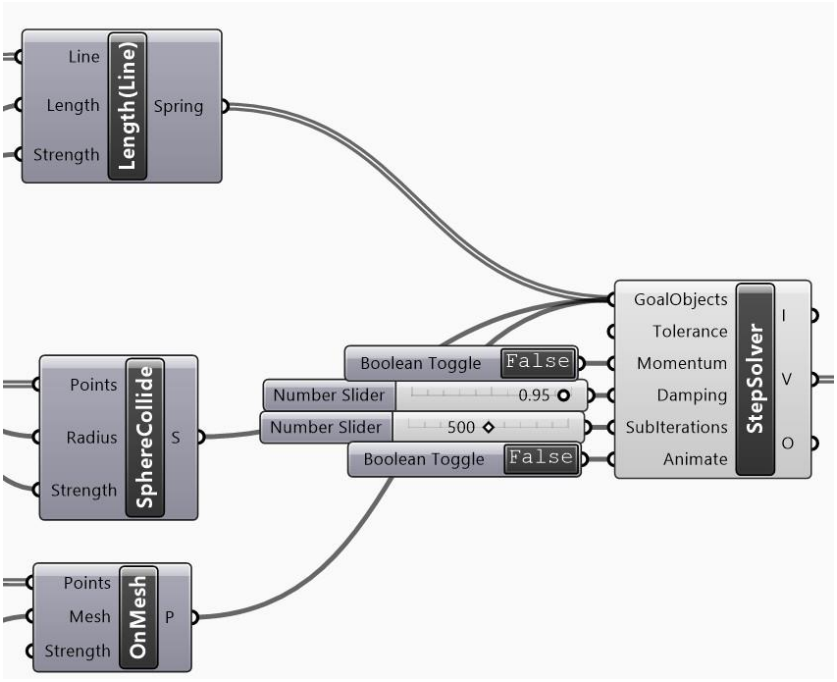


Fig. 64 Reemplazar el Step por el Step Solver de Kangaroo. Fuente: Elaboración Propia

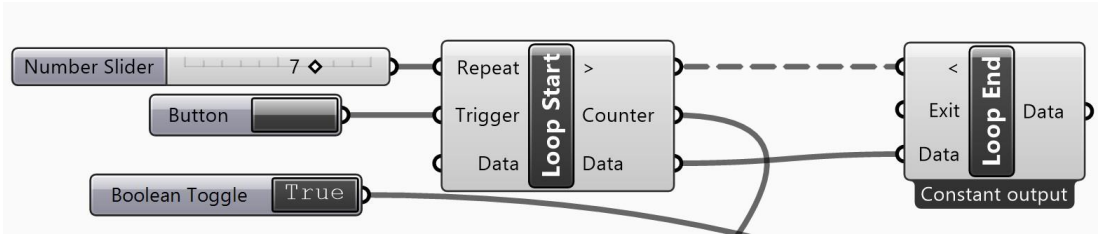


Fig. 65 Plugin Anemone. Fuente: Elaboración Propia

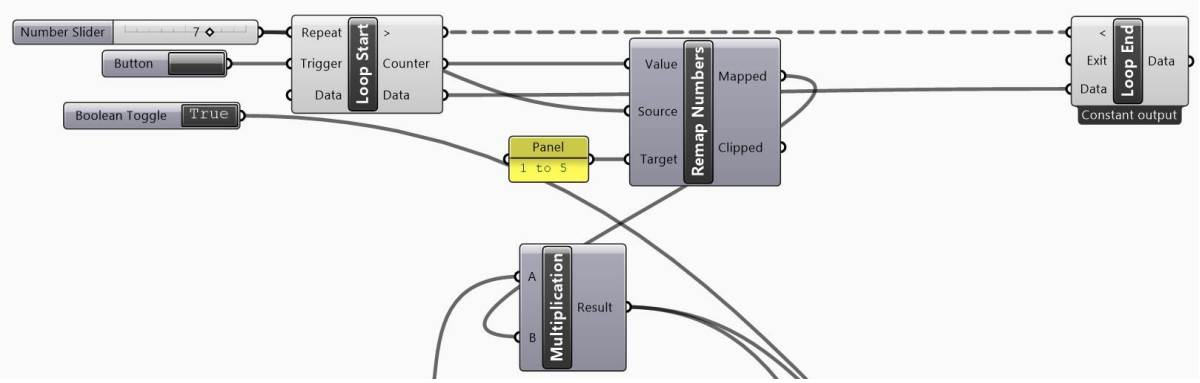


Fig. 66 Componente Remap Numbers

Esta simulando el crecimiento de ramas de coral en un entorno computacional. Cada iteración del bucle representa una nueva rama generada. Siguiendo estos pasos, puedes combinar todas las ramas generadas en una sola curva reconstruida, permitiendo así un análisis y visualización más claros del patrón de crecimiento.

Conecta la salida "Do" del componente "Loop Start" al componente "Curve". Esto proporcionará la curva generada en cada iteración del bucle.
Conecta la salida "Curve" del componente "Curve" al componente "Merge". Esto agregará las curvas generadas en cada iteración al componente "Merge".
Conecta la salida "Merge" al componente "Flatten" para aplanar el árbol de datos.
Conecta la salida del componente "Flatten" al componente "Rebuild Curve".
Finalmente, conecta la salida "Do" del componente "Loop End" al componente "Rebuild Curve" para cerrar el bucle.
Con estos pasos, las curvas generadas en cada iteración del bucle se agregarán correctamente al componente "Merge" y luego se aplanarán antes de pasarlas al componente "Rebuild Curve". Esto te permitirá visualizar y trabajar con las curvas generadas de manera efectiva. (Fig.67)

Para hacer que las curvas sean tridimensionales y se muevan en la dirección Z se hace lo siguiente:
Conecta la salida "Do" del componente "Loop End" al componente "Move". Esto enviará las curvas generadas en cada iteración del bucle al componente "Move".
Conecta la salida del componente "Loop End" al componente "List Length". Esto determinará cuántas curvas se generaron en total durante el bucle.
Conecta la salida del componente "List Length" a la entrada "C" del componente "Series". Esto proporcionará el número total de curvas al componente "Series".
Configura el componente "Series" para generar una serie de números desde 0 hasta el número total de curvas (menos 1, si es necesario) en incrementos de 1. Conecta la salida de "Series" al componente "Z" del componente "Move". (Fig.68)
Con estos pasos, las curvas se moverán en la dirección Z, creando así curvas tridimensionales. Cada curva se moverá en una cantidad diferente en función de su posición en la lista de curvas generadas durante el bucle.

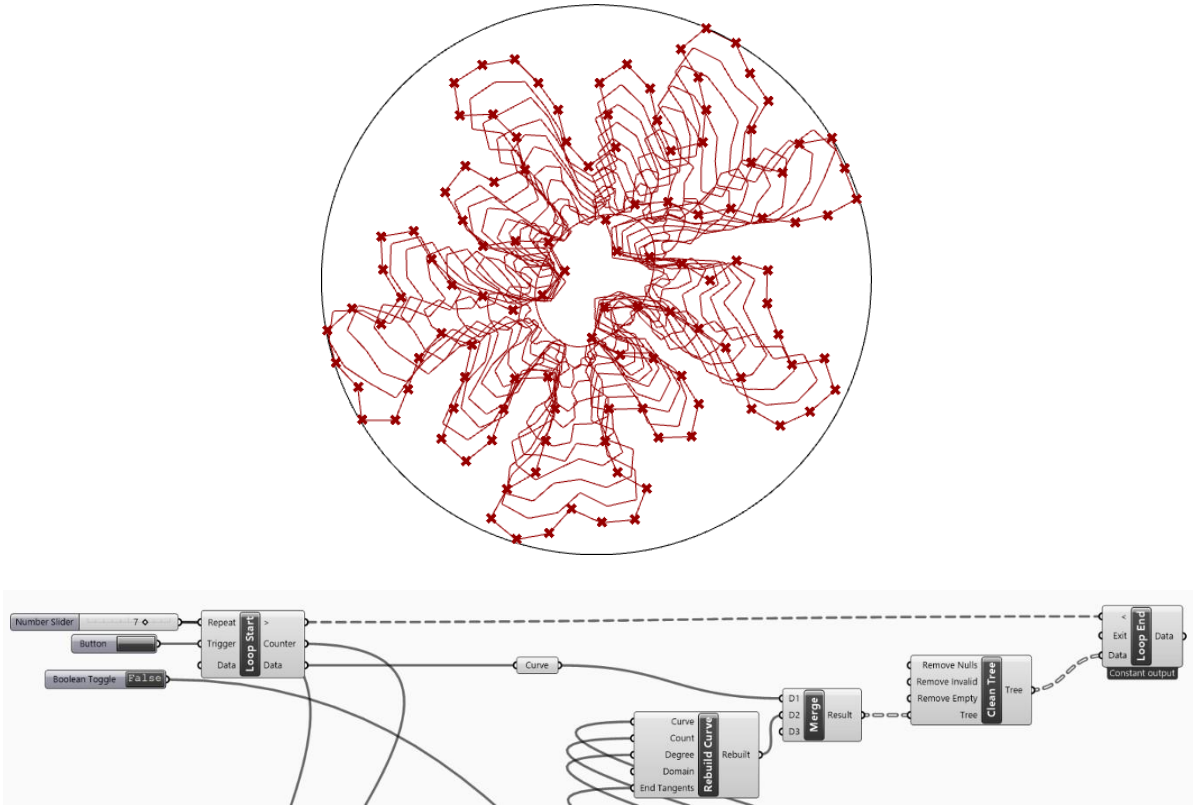


Fig. 67 Curvas generadas por el componente Merge. Fuente: Elaboración Propia

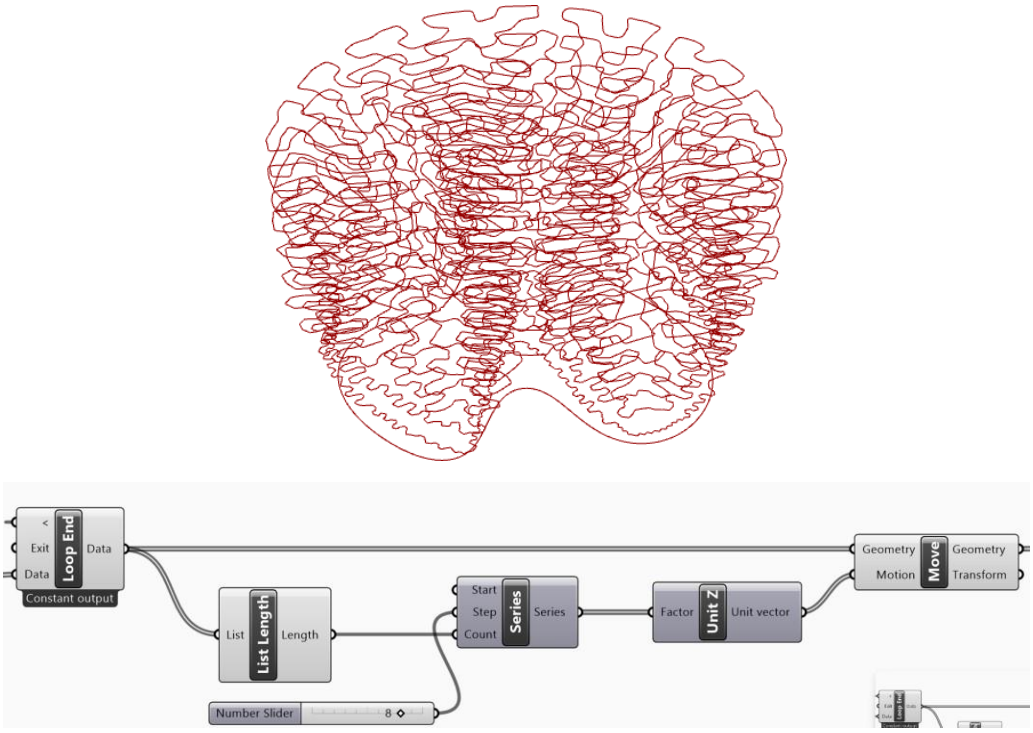


Fig. 68 Curvas tridimensionales por el componente series. Fuente: Elaboración Propia

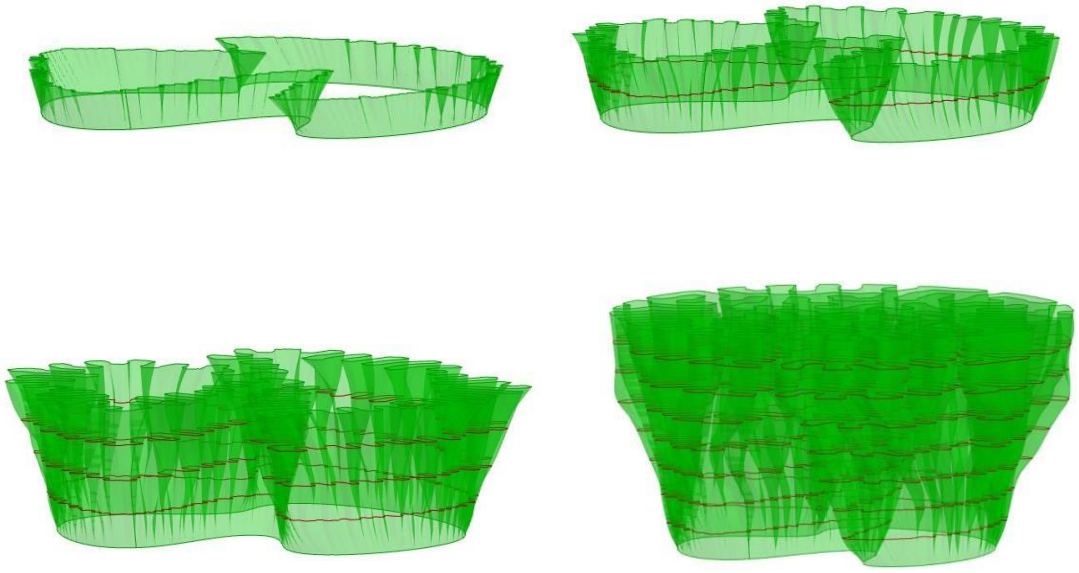


Fig. 68 Simulación de Crecimiento del coral lechuga

Fig. 69 Código QR, Simulación de crecimiento diferencial

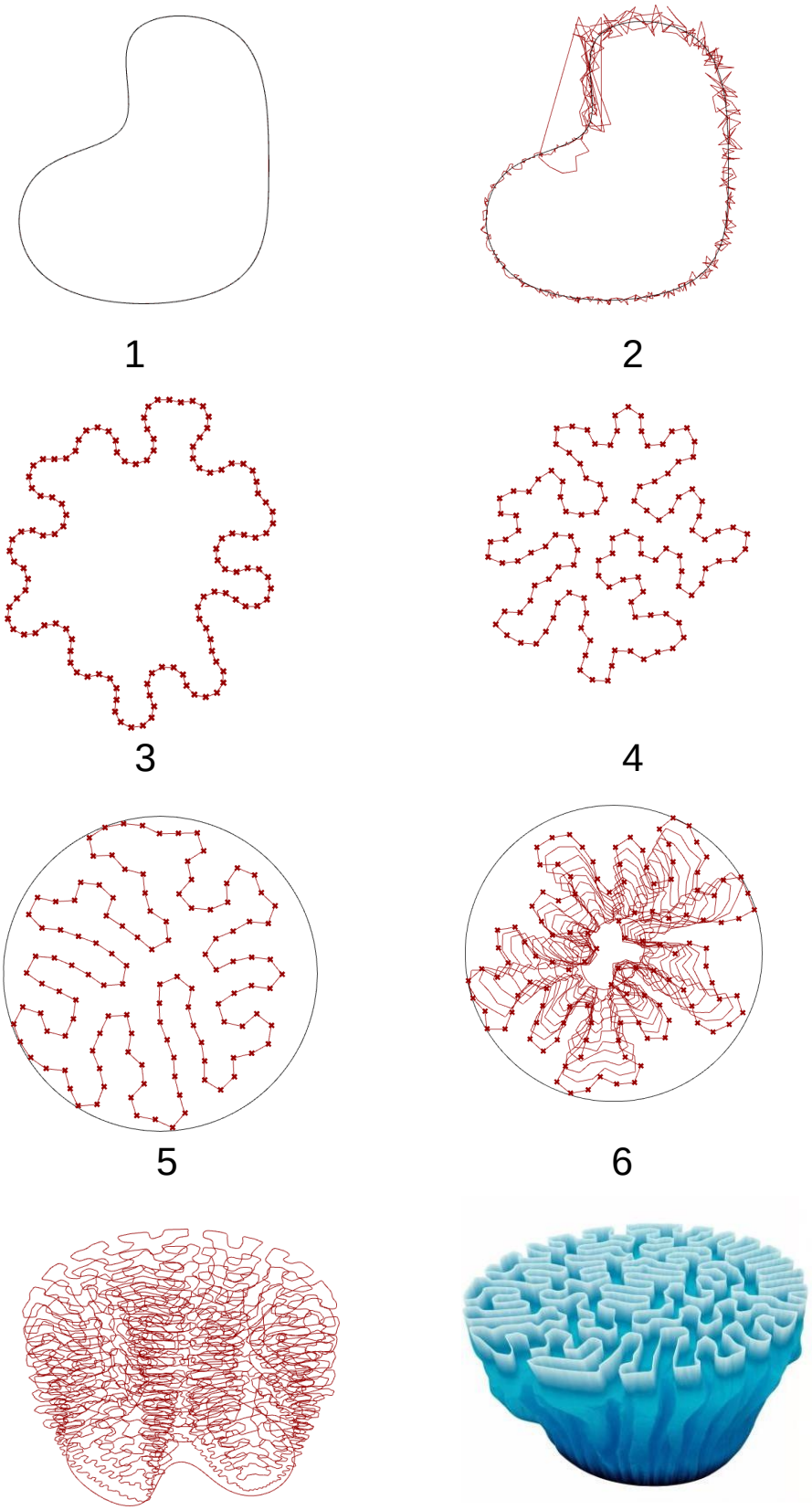


Fig.70 Etapas del proceso de crecimiento diferencial del sistema geométrico paramétrico

SISTEMA DE CRECIMIENTO Autómata Celular

3.3 Sistema de Crecimiento Autómata Celulares

Los autómatas celulares son sistemas discretos que consisten en una red de células, cada una con un estado que evoluciona según reglas específicas y la influencia de sus vecinas.

En el contexto del diseño arquitectónico, los autores proponen una adaptación de los autómatas celulares que permita explorar diferentes configuraciones espaciales y generar soluciones innovadoras para problemas de diseño. Esto se logra mediante la definición de reglas que gobiernen la interacción entre las células, representando así diferentes aspectos del diseño arquitectónico, como la distribución de espacios, la circulación, la iluminación, entre otros.

3.3.1 Enfoque aplicado al Diseño

El algoritmo propuesto se inspira en el "Juego de la Vida" de Conway, un autómata celular que utiliza reglas simples para determinar el estado de cada célula en función de sus vecinas. En nuestro caso, se establecen múltiples curvas que actúan como semillas para el crecimiento de una forma dentro del autómata celular. Cada célula dentro de la cuadrícula del autómata representa una unidad espacial, y su estado evoluciona en cada generación de acuerdo con las reglas del Juego de la Vida.

El proceso implica registrar cada generación de células en dos capas separadas: una para las células activas, que están vivas, y otra para las células inactivas, que están muertas. Esto permite seguir el desarrollo y la evolución de la forma a lo largo del tiempo, así como realizar ajustes y modificaciones en generaciones futuras para obtener la forma deseada.

Al combinar la lógica del Juego de la Vida con la representación de curvas y formas, el algoritmo ofrece una manera creativa y dinámica de explorar y generar diseño. Cada generación de células refleja una iteración en el proceso de diseño, y las formas resultantes, en este caso, representan el coral aislado.

En este caso de estudio, comenzaremos con el desarrollo del algoritmo y definiremos las funciones necesarias para encontrar el patrón del coral aislado, que es diferente a los dos casos anteriores.

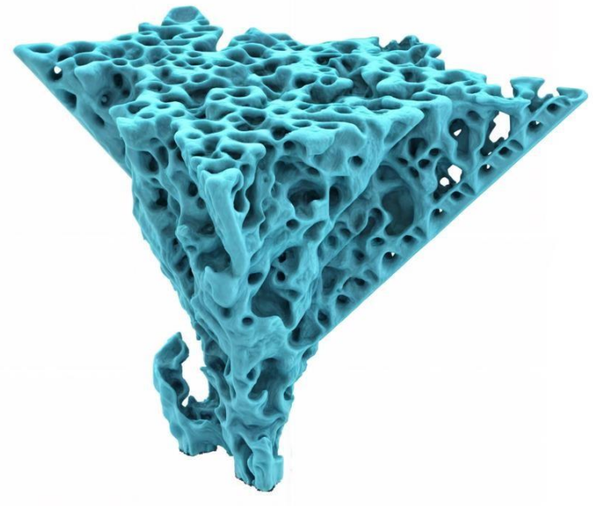


Fig. 72 Modelo coral aislado. Fuente: Elaboración Propia

1. **GRUPO A** Establecer un patrón – puntos que se interceptan
2. **GRUPO B** Direcciones para dispositivos Móviles
3. **GRUPO C** Lista de puntos muertos
4. **GRUPO D** Cuantos vecinos tiene los puntos muertos

5 **GRUPO E** Regalas de Conway

6 **GRUPO F** Automatización

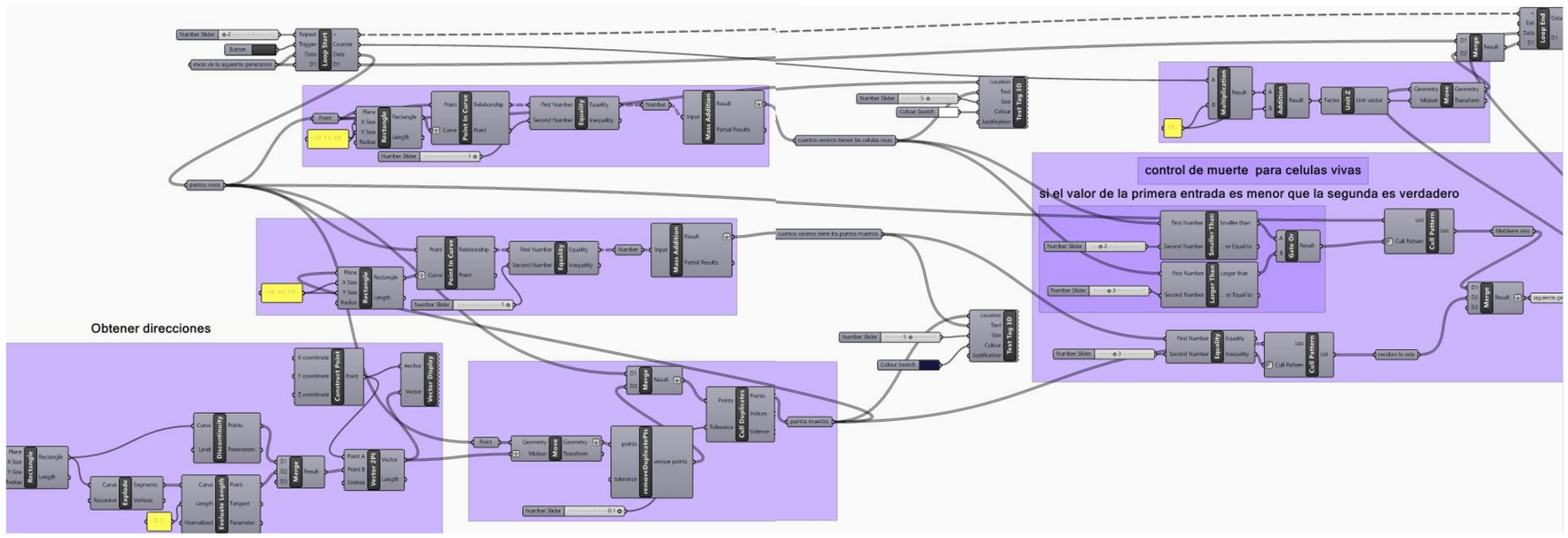


Fig. 71 Algoritmo de programación grafica de grasshopper Automata Celular. Fuente: Elaboración Propia,

Grupo A Establecer un patrón – puntos que se interceptan

El primer paso es establecer el tamaño de la célula en el autómata celular. En Grasshopper, puedes crear un punto y luego moverlo a una distancia específica para crear un patrón que servirá como base para la estructura del autómata celular. (Fig.72)

Para determinar cuántos vecinos tiene cada punto en Grasshopper, necesitamos definir primero las reglas de vecindad que queremos seguir. En el contexto del "Juego de la Vida" de Conway, los puntos o células tienen ocho vecinos potenciales (arriba, abajo, izquierda, derecha y diagonales).

Definir el tamaño del rectángulo: Conecta un panel que contenga el rango deseado (por ejemplo, -100 a 100) a las entradas X e Y del componente "Rectangle". Esto definirá el tamaño del rectángulo alrededor de cada punto. (Fig. 73)
Este enfoque te permitirá calcular los vecinos de cada punto y crear rectángulos alrededor de ellos para facilitar la verificación de su estado (vivo o muerto) en el contexto del "Juego de la Vida" de Conway en Grasshopper.

Determinar cuántos puntos interceptan correctamente cada rectángulo y obtener un recuento total de puntos correctos en todos los rectángulos.
Para determinar cuántos puntos interceptan correctamente un rectángulo. (Fig. 74)
Conectar los puntos a las curvas y luego a los rectángulos: Utiliza el componente "Point on Curve" para conectar los puntos a las curvas y luego conecta estas curvas al componente "Rectangle". Asegúrate de que la entrada C del componente "Point on Curve" esté en "Graft", lo que significa que cada punto se conectará correctamente a su respectivo rectángulo.
Determinar la coincidencia de puntos: Conecta el componente "Equals" al componente "Point on Curve" para determinar si los puntos interceptan correctamente el rectángulo. La salida de "Equals" será True si los puntos interceptan correctamente y False si no lo hacen.
Convertir True/False a valores numéricos: Conecta el componente "Numbers" al componente "Equals" para convertir los valores True/False en 1/0 respectivamente. Esto te dará una representación numérica de cuántos puntos interceptan correctamente cada rectángulo.
Sumar las respuestas: Utiliza el componente "Mass Addition" para sumar todas las respuestas numéricas, es decir, contar cuántos puntos interceptan correctamente en total.

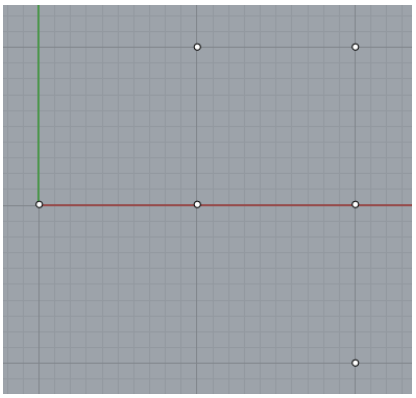


Fig.72 Determina un patrón en Rhino. Fuente: Elaboración Propia

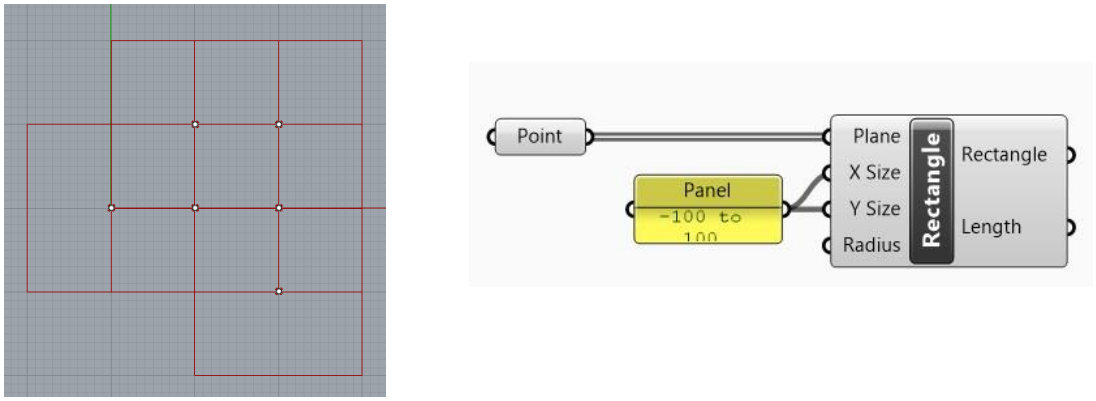


Fig. 73 Tamaño del Rectángulo. Fuente: Elaboración Propia

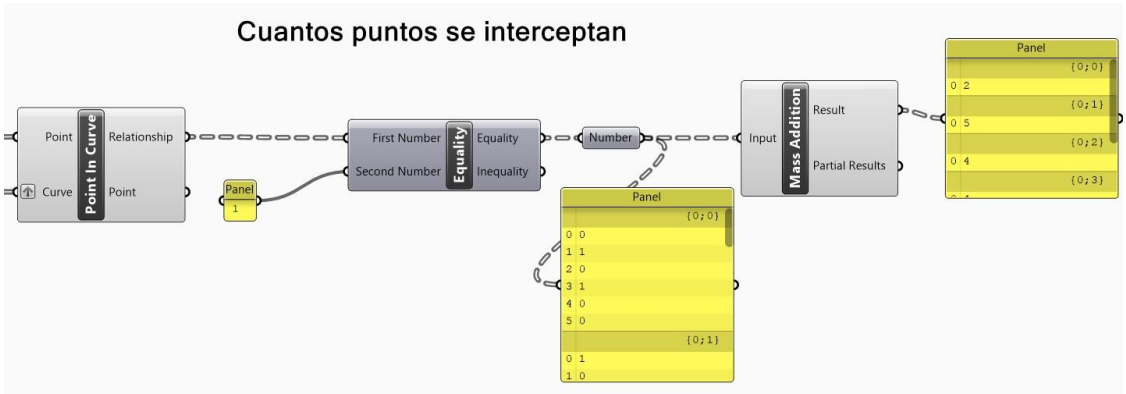


Fig. 74 Definir cuantos puntos se interceptan. Fuente: Elaboración Propia

Grupo B Direcciones para dispositivos Móviles

Proceso para Crear Puntos y Dividir un Rectángulo en Grasshopper

Este procedimiento te permite crear y manipular un rectángulo en Grasshopper, dividiendo sus lados en segmentos y evaluando los puntos medios de cada segmento.

Crear puntos alrededor del rectángulo: Utiliza el componente "Rectangle" con un parámetro de (-100 a 10) para crear un rectángulo. Luego, utiliza el componente "Construct Point" para crear puntos en las esquinas del rectángulo. Estos componentes se construirán en la coordenada 0.0.0.

Dividir el rectángulo en segmentos: Conecta el componente "Rectangle" al componente "Explode". Esto dividirá el rectángulo en 4 segmentos, pero como tienes 5 vértices, utiliza el componente "Discontinuity" conectado al rectángulo para obtener 4 puntos en lugar de 5.

Evaluar la longitud de los segmentos: Conecta el componente "Evaluate Length" al componente "Explode" en la salida S. Esto te dará la longitud de cada segmento, y necesitas que sea 0.5 para obtener los puntos medios.

Obtener los puntos medios: Utiliza el componente "Merge" para combinar las salidas P de los componentes "Evaluate Length" y "Discontinuity". Esto te dará los puntos medios de cada segmento del rectángulo.

Se necesita crear vectores alrededor del rectángulo en Grasshopper, lo que te ayudará a visualizar la distribución y orientación de los vecinos alrededor de un punto específico.

Para crear vectores alrededor del rectángulo y visualizarlos.

Crear puntos alrededor del rectángulo: Utiliza el componente "Rectangle" con un parámetro de (-100 a 10) para crear un rectángulo. Luego, utiliza el componente "Construct Point" para crear puntos en las esquinas del rectángulo y los puntos medios de los segmentos, como ya se mencionó anteriormente.

Crear vectores: Utiliza el componente "Vector 2Pt" para crear vectores entre los puntos medios y las esquinas del rectángulo. Conecta dos componentes "Construct Point" a "Vector 2Pt" para especificar los puntos de inicio y fin de cada vector. Luego, utiliza el componente "Merge" para combinar los vectores generados, lo que te dará un total de 8 vectores.

Visualizar los vectores: Para visualizar los vectores, puedes utilizar el componente "Vector Display" (a menudo abreviado como "V") conectado a la salida de los vectores. Esto te permitirá ver la dirección y la longitud de cada vector en la ventana de Grasshopper.

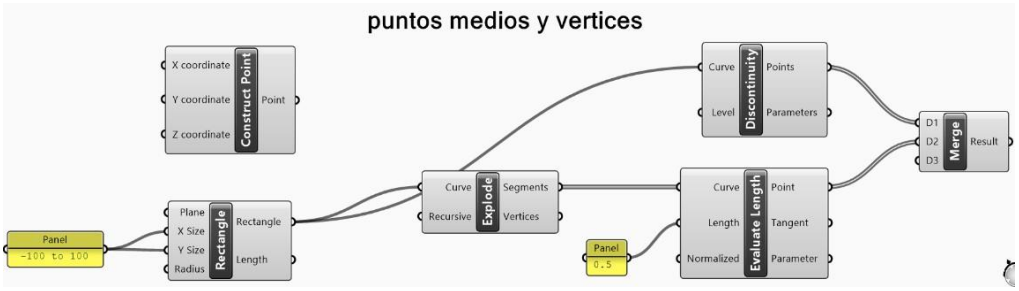


Fig. 75 Definir puntos medios de los vértices. Fuente: Elaboración Propia

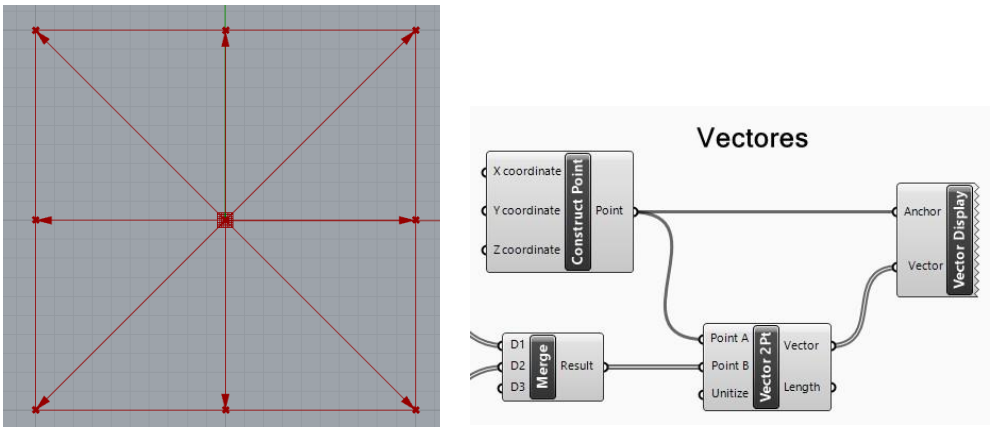


Fig. 76 Visualización de Vectores. Fuente: Elaboración Propia

Grupo C Obtención de Puntos Muertos

Mover Puntos en Ocho Direcciones y Eliminar Duplicados en Grasshopper

Este enfoque es útil para distribuir puntos de manera uniforme y asegurarse de que no haya redundancia en los datos de puntos.

Crear puntos alrededor del rectángulo: Como mencionaste anteriormente, has creado puntos alrededor del rectángulo utilizando el componente "Construct Point".

Crear vectores: Utiliza el componente "Vector 2Pt" para crear vectores entre los puntos medios y las esquinas del rectángulo, como lo hiciste anteriormente.

Mover puntos en ocho direcciones diferentes: Conecta el componente "Move" al grupo de puntos (Grupo A) que creaste anteriormente. Conecta la entrada T del componente "Move" al grupo de vectores (Grupo B) para especificar las direcciones y distancias en las que deseas mover los puntos. Esto moverá los puntos en ocho direcciones diferentes según los vectores proporcionados.

Aplanar la salida de puntos: Conecta la salida G del componente "Move" al componente "Flatten". Esto asegurará que todos los puntos estén en una sola lista, facilitando el procesamiento posterior. (Fig.79)

Eliminar puntos duplicados: Utiliza el componente "RemoveDuplicatePts" para eliminar puntos duplicados de la lista. Esto garantizará que cada punto sea único y evitará la duplicación de resultados.

Para obtener una lista de puntos que potencialmente pueden cobrar vida, es decir, aquellos puntos que no están ocupados por puntos vivos.

Eliminar puntos duplicados de los puntos vivos: Conecta el componente "RemoveDuplicatePts" a la salida de los puntos vivos. Luego, conecta el componente "CullDuplicate" a la salida del componente "RemoveDuplicatePts". Esto te dará una lista de puntos vivos sin duplicados.

Crear una lista de puntos muertos: Utiliza el componente "Merge" para combinar la lista de puntos vivos (sin duplicados) con la lista de todos los puntos generados. Conecta el componente "Point" a la salida del componente "CullDuplicate". Esto te dará una lista combinada de todos los puntos, con los puntos vivos eliminados.

la lista de puntos potencialmente vivos: Conecta la salida del componente "CullPattern" al parámetro "Flatten" del componente Merge. Esto asegurará que todos los puntos estén en una sola lista, facilitando su uso posterior. (Fig. 80)

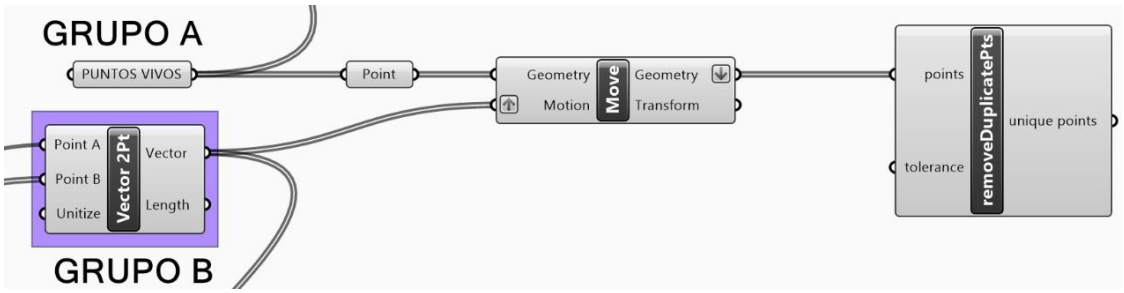
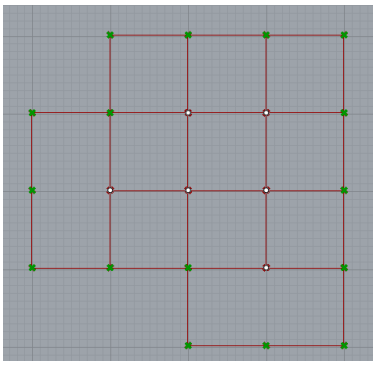


Fig. 79 Eliminación de Puntos Duplicados. Fuente: Elaboración Propia



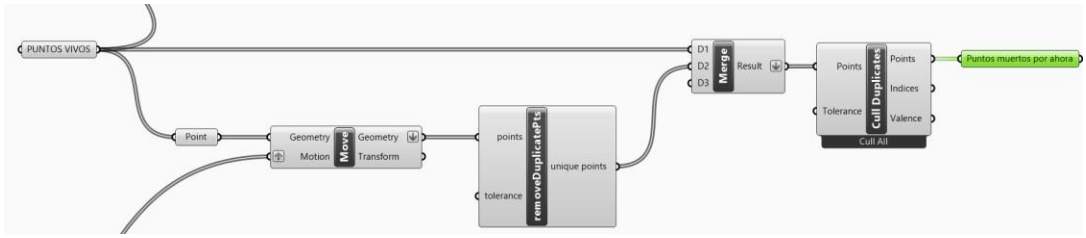


Fig. 80 Obtención de Puntos Muertos. Fuente: Elaboración Propia

Grupo D Cuantos Vecinos tiene cada Punto Muerto

Se encuentra configurando un sistema en Grasshopper que puede diferenciar entre dos estados distintos de puntos: vivos y muertos. Esta diferenciación es esencial en esta aplicación de simulación y modelado paramétrico, permitiendo un control más preciso y una mayor flexibilidad en la manipulación de los datos de puntos.

Copia el Grupo A y elimina el componente Point. Conecta el componente Point Curve al componente Point (puntos vivos) y conecta la entrada Point del componente Rectangle al componente Point (puntos muertos) dentro del mismo grupo. (Fig.81) (Fig.82)

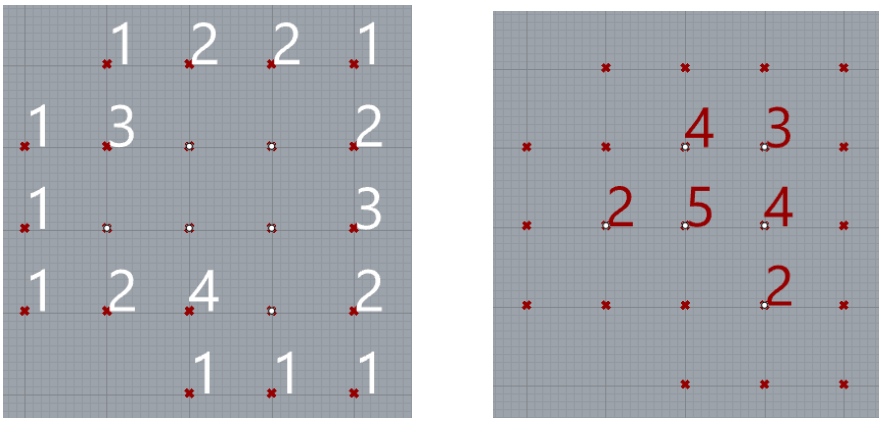


Fig. 81 Cuantos vecinos tiene cada punto muerto. Fuente: Elaboración Propia

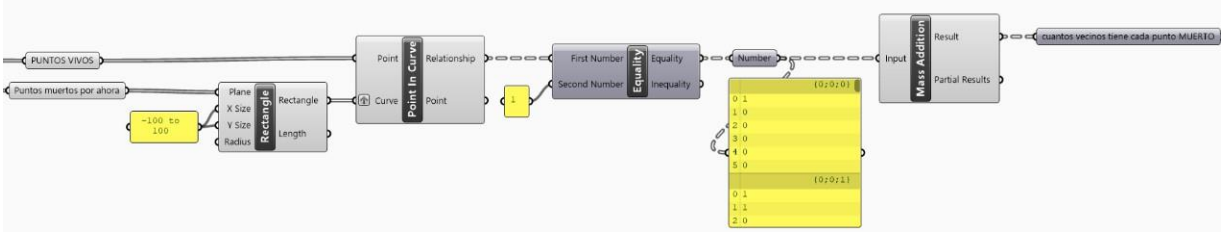


Fig. 82 Cuantos vecinos tiene cada punto muerto y cuantos vecinos tiene cada punto vivo. Fuente: Elaboración Propia

Grupo E

Este es el momento de introducir reglas sobre cómo debe comportarse los puntos, las reglas que obtendremos son del Juego de la vida Conway estas son los cuatros reglas.

1.- Cualquier célula viva con menos de dos vecinas vivas muere, como si se tratara de una subpoblación. (Fig.83)
2.-Cualquier célula viva con dos o tres vecinas vivas vive hasta la siguiente generación.
3.-Cualquier célula viva con más de tres vecinas vivas muere, como si fuera por superpoblación. (Fig.84)
4.-Cualquier célula muerta con exactamente tres vecinas vivas se convierte en una célula viva, como si se reprodujera.

El componente Points, que determina cuántos vecinos tiene cada punto vivo, se conectará al componente Smaller Than. La entrada A del componente Smaller Than recibirá el número de vecinos de cada punto vivo, mientras que la entrada B será un parámetro numérico establecido en 2. Si el número de vecinos es menor que 2, se aplicará la primera regla del Juego de la Vida.

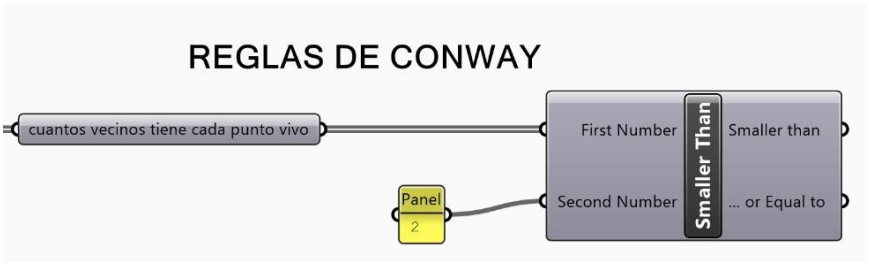


Fig. 83 Cualquier célula viva con menos de dos vecinas vivas muere, como si se tratara de una subpoblación. Fuente: Elaboración Propia

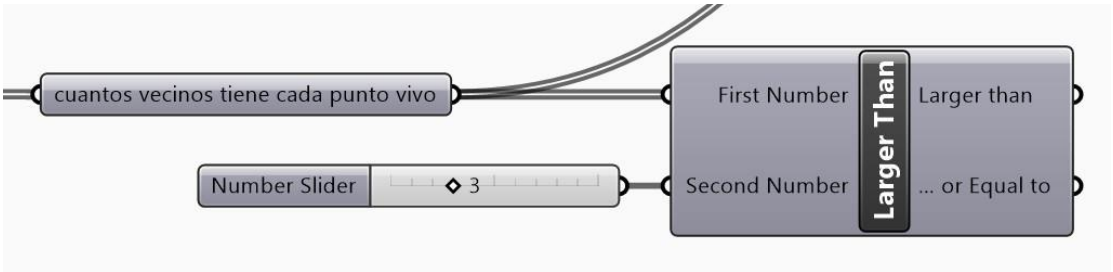


Fig. 84 Cualquier célula viva con más de tres vecinas vivas muere, como si se tratara de una superpoblación. Fuente: Elaboración Propia

El componente Points, que determina cuántos vecinos tiene cada punto vivo, se conectará al componente Larger Than. La entrada A del componente Larger Than recibirá el número de vecinos de cada punto vivo, mientras que la entrada B será un parámetro numérico establecido en 3. Si el número de vecinos es mayor que 3**, se aplicará la tercera regla del Juego de la Vida**. (Fig. 85)

Estos pasos siguientes garantizará que solo se conserven los puntos que tienen respuestas verdaderas para la siguiente generación.

El componente "Gate or" se conectará a la entrada P del componente "Cull". Las respuestas de "Gate or" son falsas y verdaderas, y de las cuales hay 3 verdaderas y 3 falsas. Solo necesitamos las verdaderas para la siguiente generación. Por lo tanto, la entrada L del componente "Cull" se conectará a los puntos vivos. (Fig.85)

Para aplicar la cuarta regla del Juego de la Vida, donde una célula muerta con exactamente tres vecinos vivos se convierte en una célula viva, seguimos estos pasos:

Conectamos el componente "Point" (cuantos vecinos tiene cada punto muerto) al componente "Equals" con un parámetro numérico de 3. Esto nos dará una lista de puntos muertos que tienen exactamente tres vecinos vivos. (Fig.86)

Luego, utilizamos el componente "Cull Pattern" para eliminar estos puntos de la lista de puntos muertos. Conectamos la salida del componente "Equals" al parámetro "Pattern" del componente "Cull Pattern". Esto asegurará que solo se conserven los puntos muertos que no tienen exactamente tres vecinos vivos, ya que queremos eliminar aquellos que podrían convertirse en células vivas según la regla.(Fig.87)

De esta manera, obtenemos una lista de puntos que representan la siguiente generación en el Juego de la Vida.

Conectamos los dos componentes "Cull Pattern" con sus respectivos componentes "Point" (que representan los puntos que deben mantenerse vivos o revivir).

Conectamos la salida de ambos componentes "Cull Pattern" al componente "Merge".

Aseguramos que el parámetro "Flatten" esté activado en el componente "Merge" para aplanar la lista de puntos resultante.(Fig.88)

La salida del componente "Merge" será el componente "Point" que representa la siguiente generación.

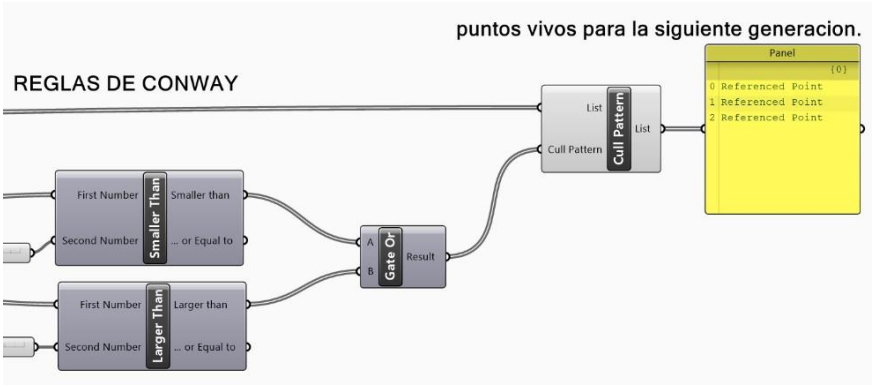


Fig.85 Puntos vivos para la siguiente generación. Fuente: Elaboración Propia

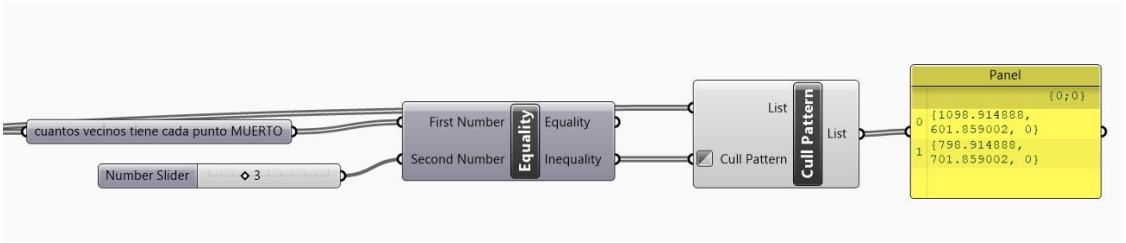


Fig. 86 Cualquier célula muerta con exactamente 3 vecinos se convierte en una célula viva, Fuente: Elaboración Propia

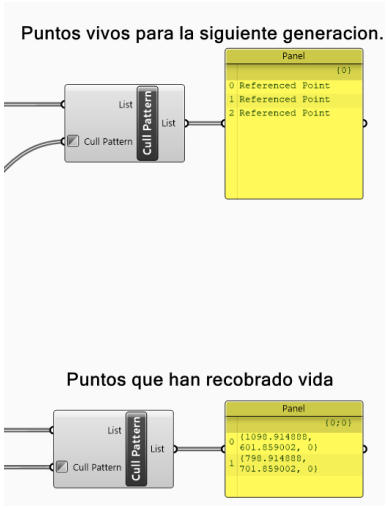


Fig. 87 Puntos vivos

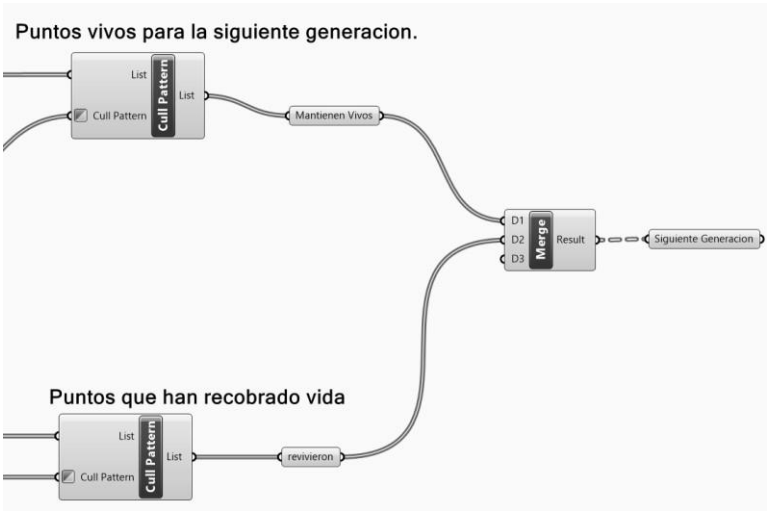


Fig. 88 Puntos vivos para la siguiente generación**.**

Fuente: Elaboración Propia

Grupo F – Automatización

Estos pasos detallan cómo configurar un bucle iterativo en Grasshopper utilizando Anemone para gestionar y manipular puntos.

Crear puntos de la generación inicial: Coloca un componente "Point" antes del Grupo A para representar la generación inicial de puntos. Estos puntos pueden ser georeferenciados desde Rhino.
Limpiar valores en el Grupo A: Dentro del Grupo A, coloca un componente puntos vivos y con click derecho "Clear Values" antes de cualquier otro componente para asegurarte de que los valores estén limpios y listos para el proceso.
Configurar el bucle Anemone: Utiliza el componente "Loop Start" para iniciar el bucle. En la entrada N, coloca un parámetro numérico de 10 para especificar el número de iteraciones que deseas. En la entrada T, proporciona un disparador adecuado para iniciar el bucle.
Conectar el proceso dentro del bucle: Conecta el componente "Point" de la generación inicial a la entrada DO del bucle Anemone. Luego, conecta el flujo de trabajo descrito anteriormente (proceso dentro del Grupo A) que incluye la manipulación de los puntos vivos y todas las operaciones necesarias.
Finalizar el bucle: Conecta la salida del bucle Anemone al componente "Point" que representa la siguiente generación. Esto asegurará que el proceso se repita según lo especificado por el bucle Anemone.

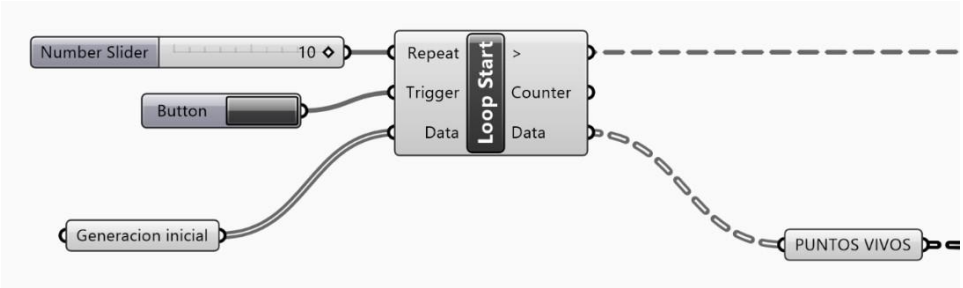


Fig. 89 Inicio bucle Anemone. Fuente: Elaboración Propia

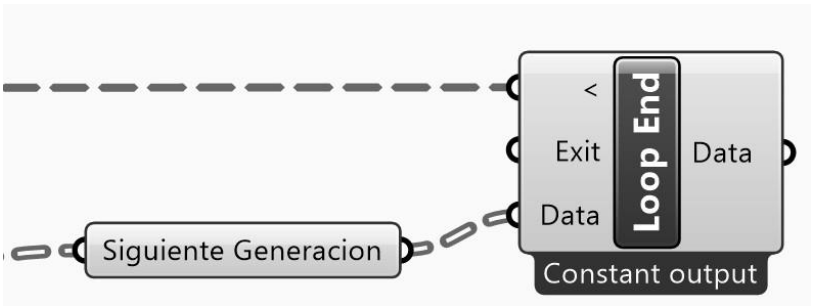


Fig. 90 Final del bucle Anemone. Fuente: Elaboración Propia

En esta fase, volveremos al patrón inicial (Grupo A) que se dibujó en Rhino. Ejecutaremos el algoritmo utilizando estos puntos, que se encontrarán georreferenciados en el componente "Curve" (generación inicial) que ya ha sido creado. Al hacer clic en el botón, el algoritmo comenzará a funcionar.

Detalles del Proceso

Volver al Patrón Inicial (Grupo A):

El patrón inicial fue dibujado en Rhino y será la base de nuestra simulación. (Fig. 01)

Georreferenciar Puntos en el Componente "Curve":

Asegúrate de que los puntos estén correctamente georreferenciados en el componente "Curve", que representa en inicio de la siguiente generacion.

Ejecutar el Algoritmo:

Con el patrón inicial configurado y los puntos georreferenciados, haz clic en el botón para iniciar la ejecución del algoritmo.

Este paso pondrá en funcionamiento el algoritmo, permitiendo que comience la iteración y manipulación de los puntos según lo especificado.

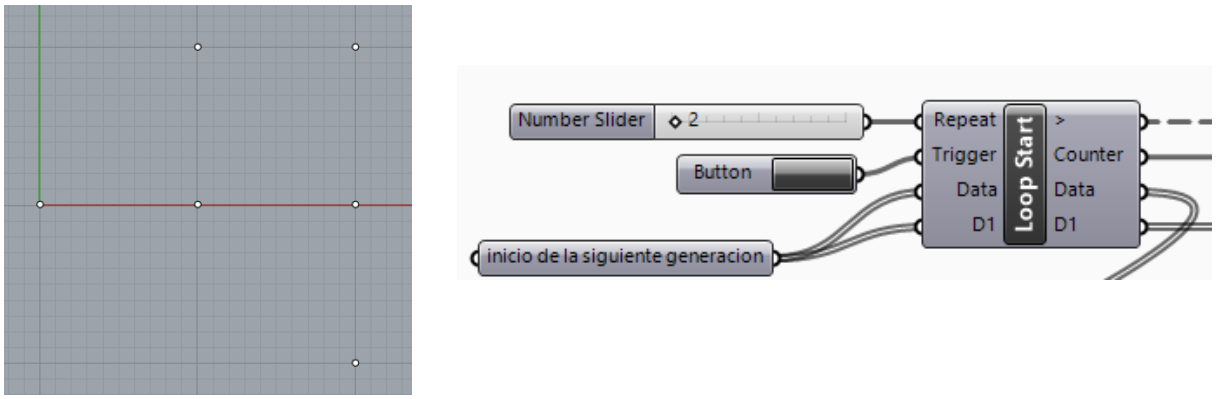


Fig. 91 Patron inicial

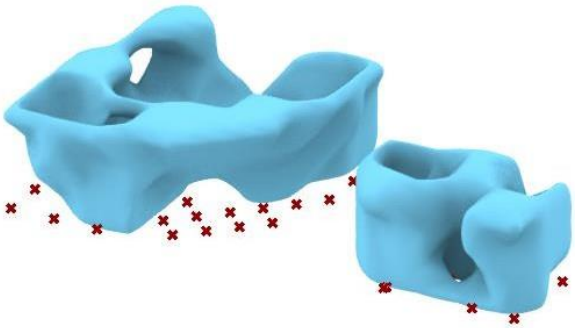


Fig. 92 Resultado del Algoritmo autómata celular. Fuente Elaboración Propia

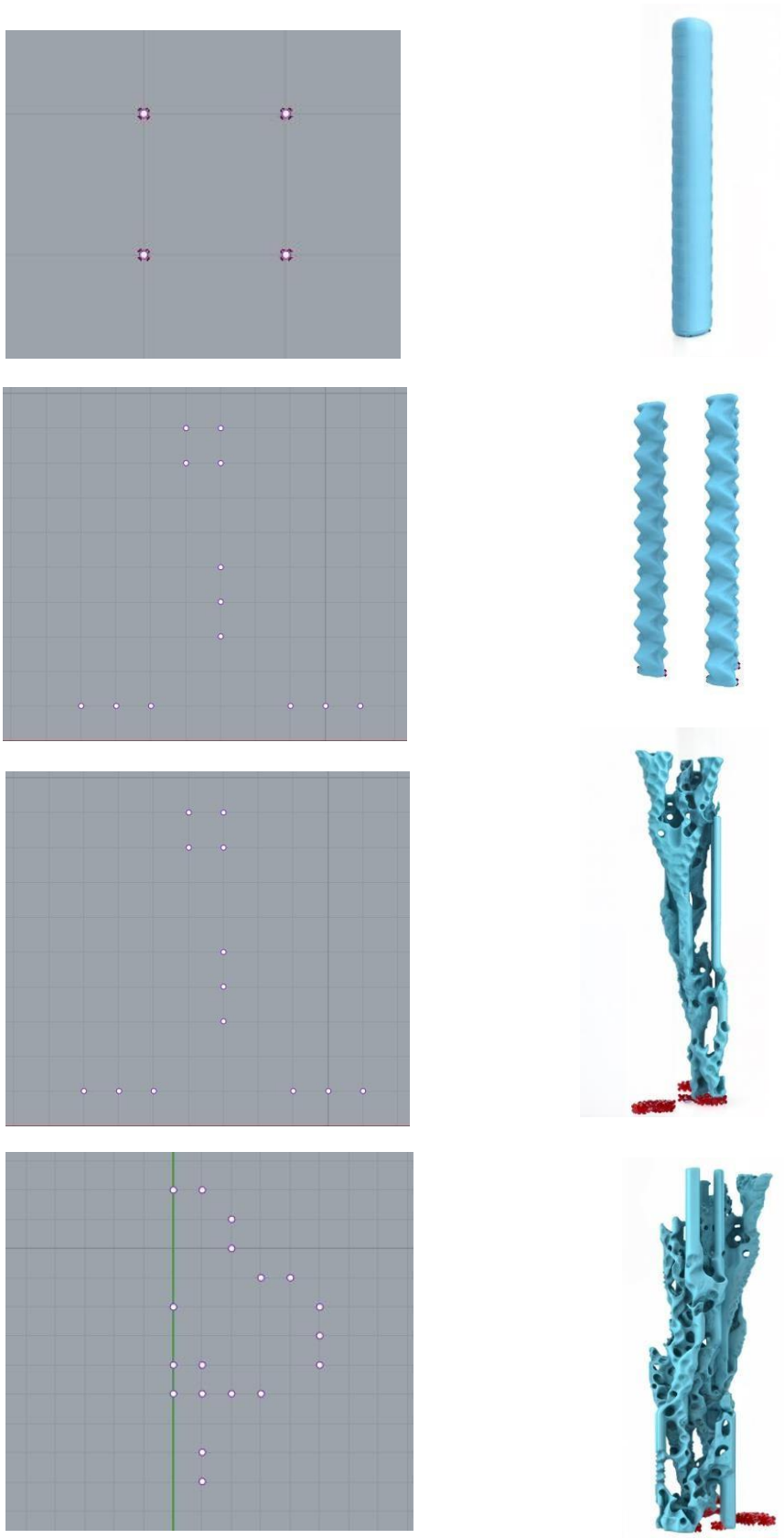


Fig. 93 Variabilidad geométrica a partir de los patrones, basado en un algoritmo de programación grafica grashopper. Fuente: Elaboración Propia

3.3.2 Representación del coral Cuerno

En esta fase exploraremos el patrón el coral cuerno de alce (Acropora palmata), es conocido por su apariencia ramificada y ramificaciones irregulares. intentar diseñar reglas que fomenten el crecimiento de estructuras ramificadas en la simulación del juego de Conway, como reglas que favorezcan la proliferación de células en direcciones específicas o reglas que simulen la competencia por el espacio entre las células.

El coral cuerno de alce (Acropora palmata) y el Juego de la Vida de Conway tienen en común la manifestación de patrones complejos a partir de reglas simples.
Crecimiento y Formación de Patrones:
Acropora palmata: El crecimiento del coral cuerno de alce se basa en procesos biológicos que siguen ciertas reglas naturales, como la forma en que sus ramas se extienden y se ramifican. Estas reglas biológicas, aunque simples en su base, pueden producir estructuras muy complejas y diversas.
Juego de la Vida de Conway: Este juego es un autómata celular que sigue reglas simples para determinar el estado de cada célula en una cuadrícula. A pesar de la simplicidad de las reglas, puede generar patrones increíblemente complejos y variados a lo largo del tiempo.

Fig. 94 Un grupo de coral cuerno de alce en Pickles Reef por Jennifer Adler

Para encontrar el patrón de crecimiento del coral cuerno de alce (Acropora palmata), se desarrolló un algoritmo de autómata celular bidimensional. Aunque los detalles específicos del algoritmo no están descritos en este capítulo, surgió como base para la aplicación de juegos infantiles (capítulo 4). El algoritmo se basó en el patrón de una "nave espacial" que se explicó en el segundo capítulo. Este patrón fue modificado iterativamente hasta obtener el patrón deseado. Una vez identificado el patrón adecuado, se ejecutó en el algoritmo previamente descrito, obteniendo así una simulación precisa del crecimiento del coral cuerno de alce (Fig. 95)

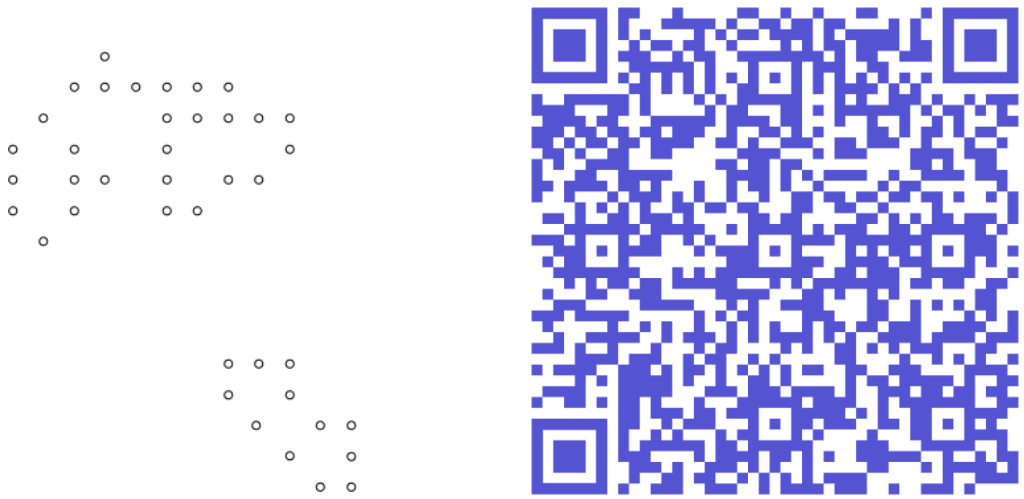


Fig. 95 Patrón del coral cuerno y QR. Fuente: Elaboración Propia

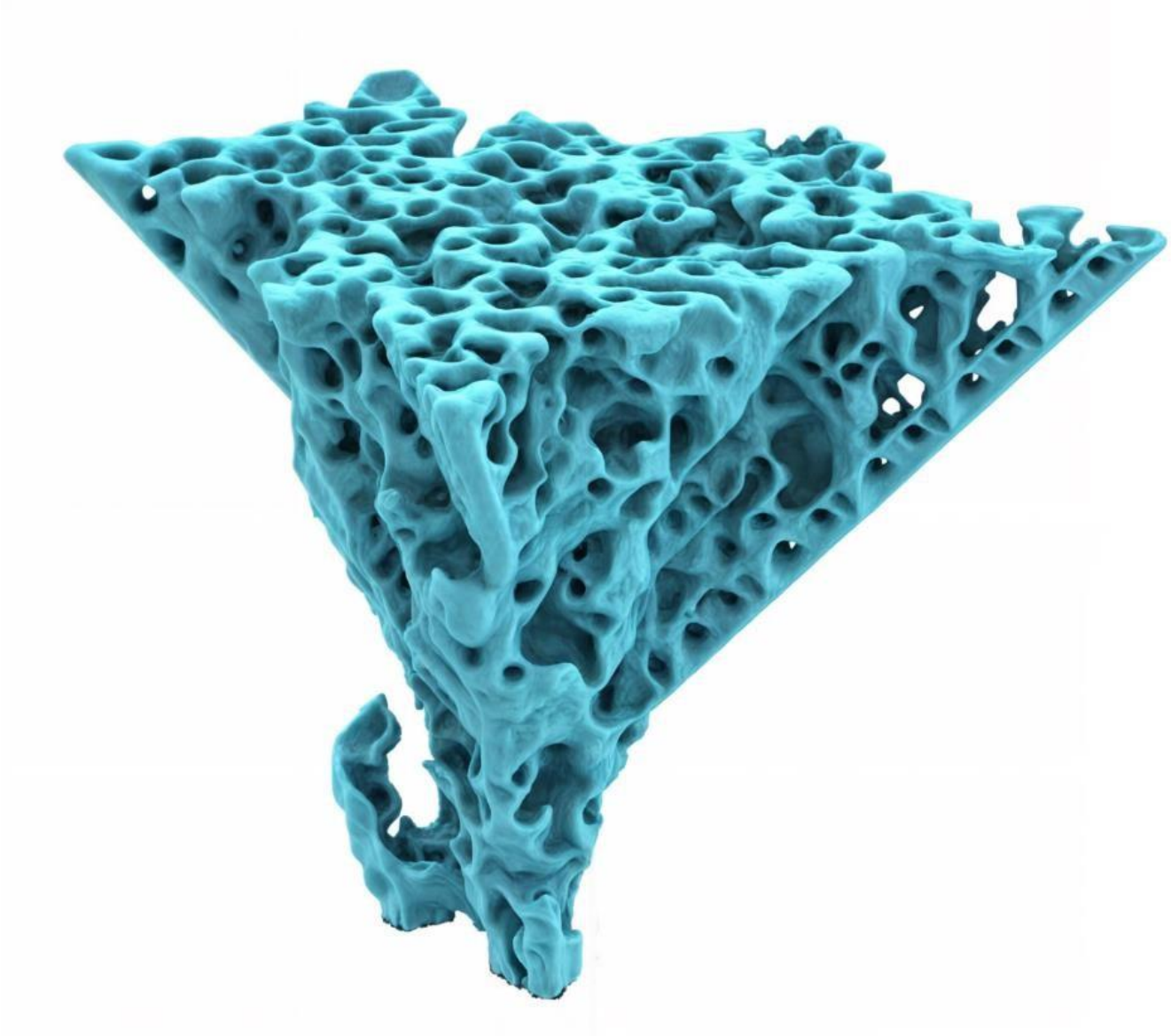
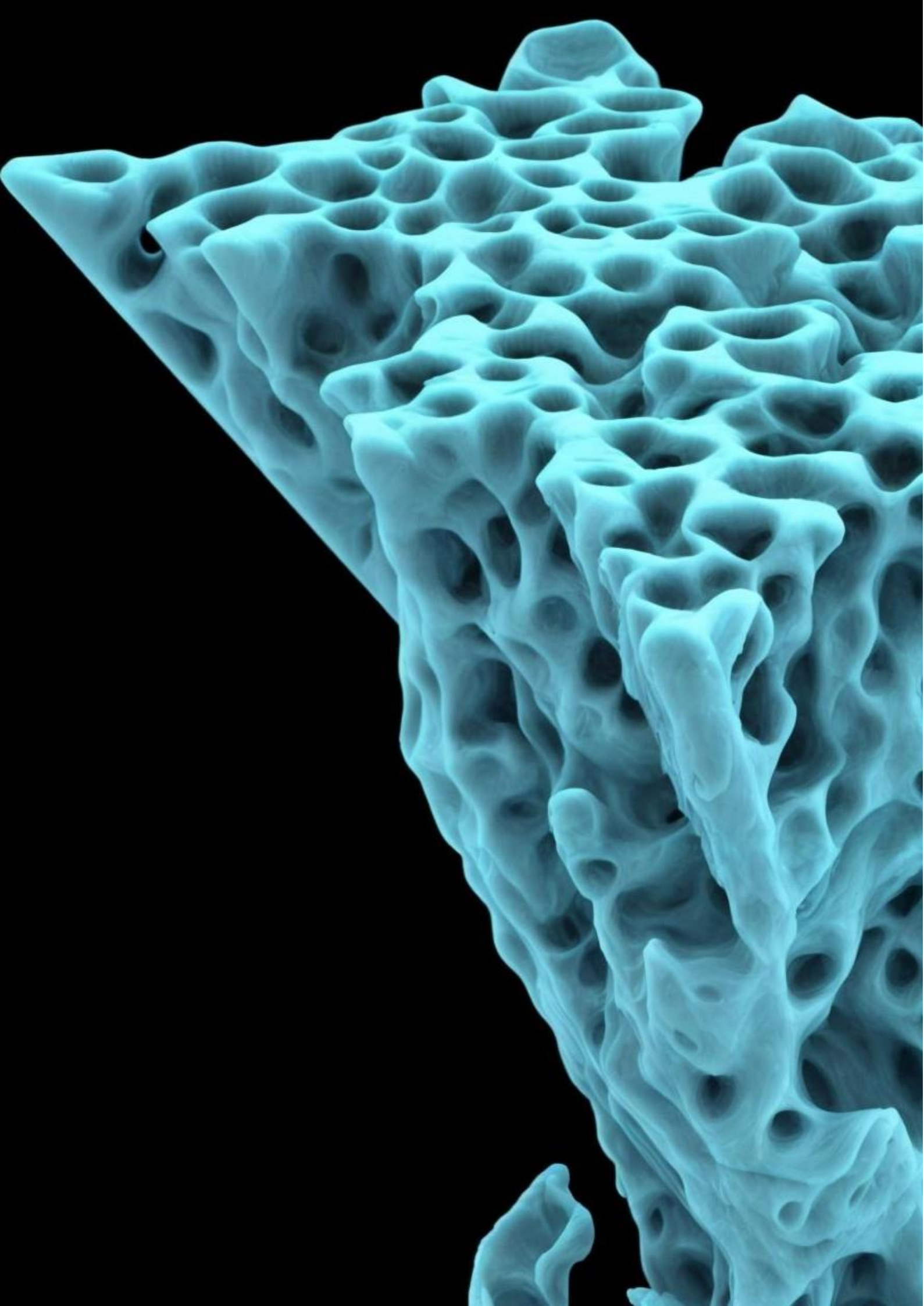


Fig. 96 Modelo tridimensional coral de cuerno. Fuente: Elaboración Propia



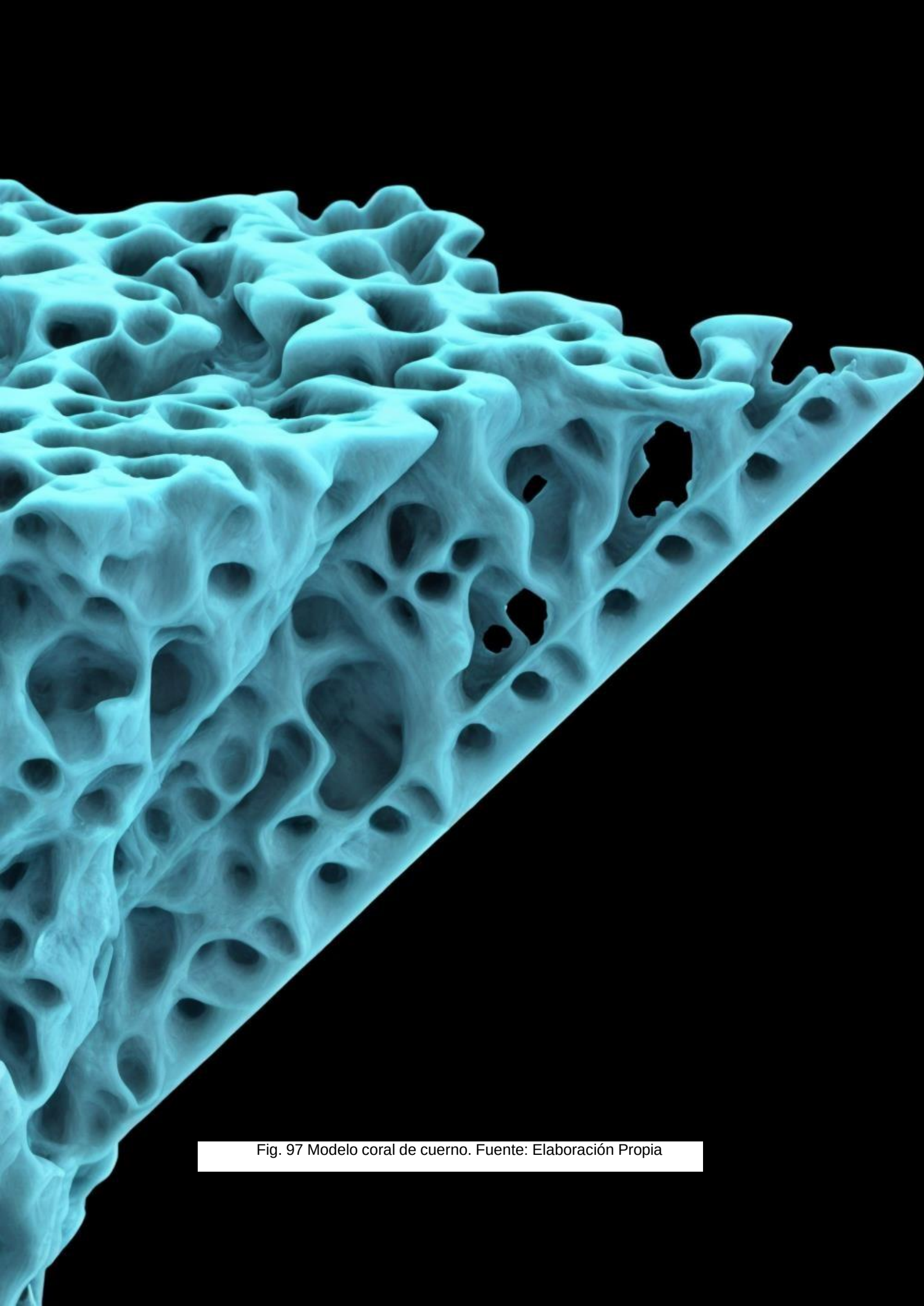


Fig. 97 Modelo coral de cuerno. Fuente: Elaboración Propia

CAPITULO 4

APLICACIÓN A JUEGOS INFANTILES

4 APLICACIÓN A JUEGOS INFANTILES

EN LA ACTUALIDAD

Actualmente, existe una crisis de imaginación; los niños de hoy esperan ser entretenidos por lo que tienen frente a ellos, sin hacer ningún esfuerzo. El poder de la imaginación se ha perdido, ya no pueden transformar algo sin vida o sin sentido en cualquier cosa que su mente pueda concebir, dándole significado a través de su imaginación. Aldo van Eyck previó este modelo educativo e intentó oponerse a él. En los parques infantiles contemporáneos, los elementos tienen formas perfectamente reconocibles, lo que impide que el niño imagine que un arco pueda ser un puente, una casa, un árbol o cualquier otra cosa, y hace que el elefante de hierro y madera no pueda ser otra cosa que eso. Además, estas figuras a menudo se fijan en el inconsciente del niño como la propia figura que representan: pocos niños habrán visto un elefante real antes de ver el elemento del juego en su parque.

4.1 Imaginación y Aprendizaje

Resulta esencial de hablar de la imaginacion y el asombro como herramientas fundamentales para comprender el mecanismo de diseño de los parques infantiles. El desarrollo cognitivo un ñiño es esencial de la manera que se relaciona con el entorno proximo en sus primeros años de vida . Asimismo el juego es fundamental para el desarrollo cerebral

Un concepto Muy interante en relacion al aprendizaje y teorias pedagogicas: el ñiño ha de aprender dentro hacia afuera, como defiende el neuropsiquiatra Daniel J. Siegel. (Fig.1)

Las diversas teorias pedagogicas como las Pestalozzi y Montesori , ponen el foco que es el niño quien es motor del aprendizaje. Y una de las principales herramientas que utiliza para ello es la imaginacion. Los parques infantiles de Aldo van Eyck fomenta que sea el niño quien busque su propia experiencia de forma activa, atraves de estructura minima. Gracias a esta es plasmada en los parques infantiles con elementos de juegos abstractos y sencillos.(Fig.2)

Estudios demuestran que el juego libre no estructurado facilita el desarrollo de capacidades de resolucion de problemas en la edad adulta.

Figura 1. Concepto Junk Playground Fuente: Extraído Tesis Jaime Álvarez Santana,2017

Figura 2. Parque.Simonskerkestraat año 1966 Estructuras metálicas con aspersores de agua. Fuente: Extraído de la tesis Jaime Álvarez Santana, 2017

4.1.1 Montessori en la Arquitectura

"Una actitud más justa y caritativa sería crear un ambiente adecuado en el que el niño estuviera libre de la opresión de los adultos, donde realmente pudiera prepararse para la vida. La escuela debería sentirse como un refugio frente a la tormenta o como un oasis en medio del desierto, como un refugio seguro para su espíritu". (María Montessori)

A principios del siglo XX, María Montessori inició su enfoque pedagógico científico, que se centra en la estimulación para favorecer el desarrollo cerebral durante la infancia, respetando la individualidad, autonomía, autoestima y confianza en los niños.
Una parte fundamental de su filosofía es la creación de espacios que permitan la libertad para desarrollar la mente, pensar y crear, donde las texturas, tamaños y colores potencian las cualidades de los niños a través de la estimulación.
En la actualidad, este método es ampliamente utilizado en los espacios educativos infantiles para mejorar el aprendizaje.

Las características especiales que los espacios Montessori comparten son las siguientes:

Simplicidad, organización y silencio
Es fundamental crear un ambiente tranquilo y acogedor que favorezca la toma de decisiones y la concentración, evitando un espacio con aparatos electrónicos que puedan crear confusión.
Seguridad.
El espacio debe ser explorable y seguro. En estas edades el descubrimiento se hace a través del juego, por eso se debe potenciar que sea de manera autónoma, diseñando estancias adaptadas a sus usuarios.
Accesibilidad y ergonomía
El entorno está diseñado para poder desarrollar las actividades diarias. Disponer todo de forma accesible y segura es la clave para poder alcanzar objetos, coger herramientas, abrir cajones, etc. sin depender de los mayores y estimular así el conocimiento.

Una de las actividades más importantes es la hora de ir al baño. El acceso al mismo debe ser independiente para adquirir buenos hábitos. Saber la altura y la disposición es importante a la hora de diseñarlos. (Fig. 3)

Estimulación del tacto y la vista.

Juego y arquitectura forman un tándem en el diseño. Mediante espacios abiertos, diferentes revestimientos, acabados y texturas, se invita a los infantes a ser protagonistas tanto en lo social como en lo físico e interactuar con su entorno.

Se recomienda priorizar los colores claros y la luz natural.

Decoración y accesorios.

Las ilustraciones y los dibujos despiertan el interés de los niños por el arte, pero sólo funcionan si están a la altura adecuada,

Los estantes y percheros deben permitir colgar mochilas, cazadoras y paraguas sin la ayuda de un adulto y facilitan la organización del espacio.

Las alfombras al pie de la cama o del sofá posibilitan no pisar un suelo frío al despertarse y permiten pisar con firmeza y seguridad.

Fomentar la curiosidad y descubrimiento.

Las escuelas infantiles ubicadas normalmente en las ciudades no disponen de grandes espacios exteriores, pero pueden ser igual de estimulantes y creativas si se crean espacios que suplan esas carencias. Emplear paredes móviles que dividan o unan las estancias, utilizar elementos transparentes para crear comunicación visual y fomentar la colaboración, son algunos de los factores que tener en cuenta a la hora de diseñar las aulas.

Fig. 3 Como estimular la autonomía en el niño. Fuente: Cuchi Moveis

4.1.2 Antropometría y ergonomía en los niños

La antropometría se refiere al estudio del desarrollo físico del ser humano. En los primeros años de vida, especialmente entre los 0 y 2 años, los niños experimentan un crecimiento acelerado, duplicando aproximadamente su tamaño inicial. Debido a este rápido desarrollo, los juegos en los parques infantiles no suelen estar diseñados para este grupo de edad. En contraste, los niños a partir de los 5 años crecen a un ritmo más gradual y desarrollan un dominio del espacio más equilibrado. (Fig. 4) y (Fig. 5)

Desde el punto de vista de la ergonomía, muchos juegos infantiles se diseñan utilizando datos antropométricos para que sus dimensiones sean adecuadas para los niños. Sin embargo, esto no es suficiente. La interacción física del niño con los juegos también depende de otros factores, como los materiales utilizados y el tipo de actividad o postura que implican. Estos aspectos deben ser considerados en relación con el nivel de desarrollo del niño para garantizar su seguridad y disfrute.

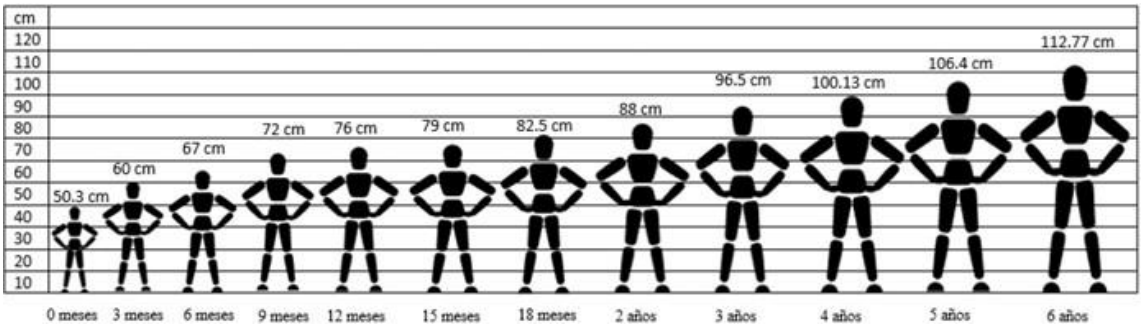


Fig. 4 Estatura de niños de 0 a 6 años. Fuente: Fundación Faustino Orbegozo 2009

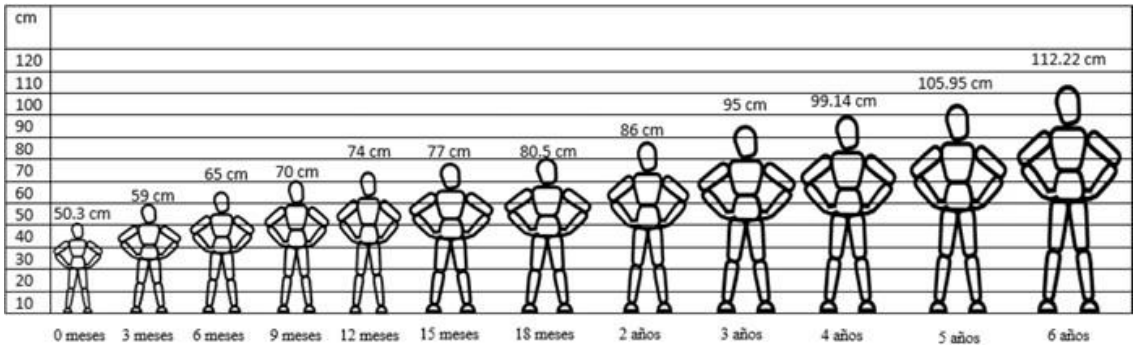


Fig. 5 Estatura de niñas de 0 a 6 años. Fuente: Fundación Faustino Orbegozo 2009

4.1.3 JUGAR CON AGUA

David Sobel, un destacado especialista en educación ambiental, ha escrito ampliamente sobre la importancia del juego en la naturaleza y su impacto en el desarrollo emocional y cognitivo de los niños. Sobel sostiene que la interacción con elementos naturales, como el agua, es fundamental para el crecimiento integral de los niños.

En su obra, Sobel destaca cómo el juego con agua proporciona una experiencia sensorial rica que no solo es agradable sino también educativa. A través del juego con agua, los niños exploran conceptos científicos básicos como la flotación, el flujo y la densidad, desarrollando así habilidades cognitivas importantes de manera lúdica y natural. Además, el agua tiene una capacidad única para capturar la atención y fomentar la concentración, lo que puede ser beneficioso para el desarrollo de la atención sostenida en los niños.

Sobel también subraya el aspecto emocional del juego con agua. La naturaleza relajante y terapéutica del agua puede ayudar a los niños a gestionar el estrés y las emociones. Jugar en un entorno natural, con acceso a arroyos, charcos y ríos, proporciona un espacio seguro donde los niños pueden expresar sus sentimientos y experimentar una sensación de calma y bienestar.

La conexión con el agua en el juego también fomenta el desarrollo social. Cuando los niños juegan juntos en entornos acuáticos, aprenden a cooperar, compartir y resolver problemas de manera conjunta. Estas interacciones son cruciales para el desarrollo de habilidades sociales y emocionales, como la empatía y la comunicación efectiva.

SISTEMA DE CRECIMIENTO Autómata Celular

Las primeras ideas para desarrollar este juego se centraron en la libertad de participación. El juego debe ser voluntario; si fuera obligatorio, perdería su cualidad de diversión. Además, el curso del juego no debe estar predeterminado, permitiendo así que los niños exploren y creen libremente. Sin embargo, se comprendió claramente que el juego debería estar regido por normas para mantener estructura y coherencia.
Otro elemento importante es el tiempo. En la actividad lúdica, la noción del tiempo queda suspendida: el tiempo se detiene, se dilata y se interrumpe. Los niños pierden la percepción del tiempo mientras juegan, inmersos en la experiencia.

Por ello, se pensó en un tablero donde los niños puedan dibujar patrones libremente, permitiéndoles plasmar su creatividad sin restricciones y sumergirse completamente en el juego. Sin embargo, este juego debe interactuar y responder al niño de manera dinámica. En lugar de un simple chorro de agua, la respuesta del tablero será más sofisticada.

Para el desarrollo del tablero se propone utilizar un algoritmo desarrollado en RHINO-GRASSHOPPER. El enfoque de diseño del algoritmo está basado en el juego de Conway, que ya estudiamos en el tercer capítulo (autómatas celulares). A diferencia de lo mencionado anteriormente, lo estudiaremos en dos dimensiones, empleando las reglas del "Juego de la Vida" de Conway.
El uso de estas reglas permitirá que el tablero responda dinámicamente a las interacciones de los niños, generando patrones acuáticos que evolucionan de manera impredecible y fascinante. Estas respuestas acuáticas no solo estimularán la creatividad y la imaginación de los niños, sino que también les ofrecerán una experiencia lúdica única, donde el agua se transforma en formas y cortinas, proporcionando una fantasía interactiva y educativa.

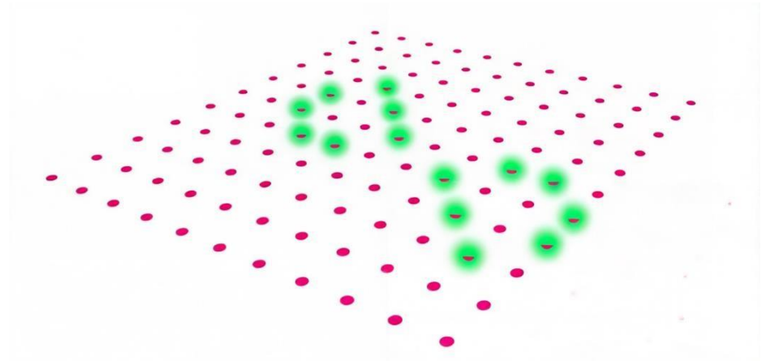
4.2 Autómatas Celulares: Juego de la Vida 2D

4.2.1 Enfoque de Diseño

El Juego de la Vida de Conway es un autómata celular que funciona en un entorno bidimensional. Este sistema puede visualizarse como una cuadrícula en la que cada celda puede estar en uno de dos estados: viva o muerta. Las celdas evolucionan a través de iteraciones basadas en un conjunto de reglas simples, las cuales determinan si una celda viva sigue viva, muere, o si una celda muerta cobra vida en la siguiente iteración. Las reglas son las siguientes:

Supervivencia: Una celda viva con 2 o 3 celdas vecinas vivas permanece viva para la siguiente generación.
Muerte por soledad: Una celda viva con menos de 2 celdas vecinas vivas muere en la siguiente generación.
Muerte por sobrepoblación: Una celda viva con más de 3 celdas vecinas vivas muere en la siguiente generación.
Reproducción: Una celda muerta con exactamente 3 celdas vecinas vivas vuelve a la vida en la siguiente generación.

Para entender mejor el funcionamiento, imaginemos una cuadrícula donde cada celda puede tener hasta 8 vecinas (las celdas adyacentes en horizontal, vertical y diagonal). En cada paso del tiempo (o generación), se evalúa el estado de todas las celdas de acuerdo con las reglas mencionadas. Este proceso se repite indefinidamente, lo que da lugar a patrones que pueden ser estáticos, oscilantes, o viajar a través de la cuadrícula.



1. **GRUPO A** Desarrollo de la Cuadricula
2. **GRUPO B** Automatización de la capa Active
3. **GRUPO C** Determinar los vecinos que se encuentren alrededor de los puntos
4. **GRUPO D** Reglas del juego Conway
5 **GRUPO E** Anemone

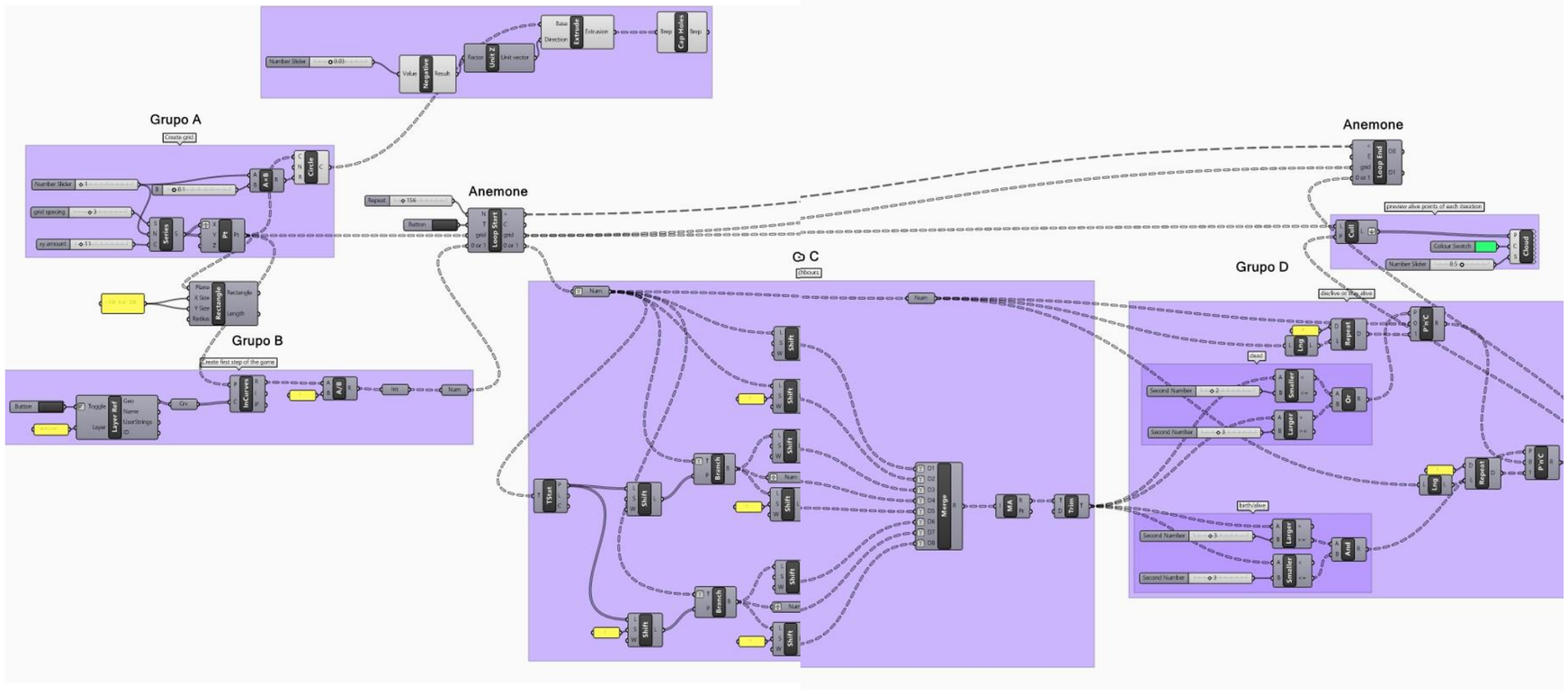


Fig. 5 Algoritmo de Programación grafica de grasshopper Automata celular **. Fuente;**
Elaboración Propia.

4.3 Enfoque de Diseño

Grupo A: Cuadricula

En este paso crearemos una cuadricula cual será nuestra área de trabajo. Para definir los puntos vivos y muertos.

Para crear una cuadrícula en Grasshopper para el Juego de la Vida de Conway, utilizaremos los componentes Construct Point y Series.

El componente Construct Point y colócalo en el lienzo de Grasshopper. Este componente te permite crear un punto en las coordenadas (0,0,0) por defecto.)

El componente Series en la entrada N colocará parámetro numérico 10 y en la entrada C se colocará un parámetro numérico 15) lo que obtenemos es 15 números que están espaciados por una distancia de 10.

La salida del Componente series se conectará a la entrada x, y del componente Construct Point y colocaremos Graft en la entrada en x). como resultado obtendremos una cuadricula completa con puntos en ambas direcciones x,y. (Fig.7)

En Rhino, se creará una capa llamada "Active". Se dibujarán unos rectángulos en esta capa. Los puntos que se encuentren dentro de los rectángulos serán considerados "vivos" y se moverán a una capa llamada "Vivos". Los puntos que estén fuera de los rectángulos serán considerados "muertos". Esta será tu condición inicial. (Fig. 8)

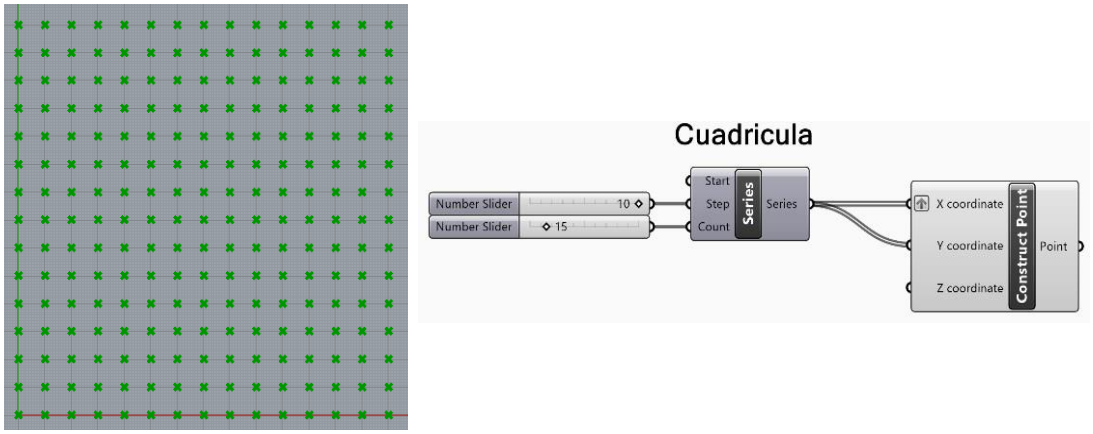


Fig. 7 Cuadricula Fuente: Elaboración Propia

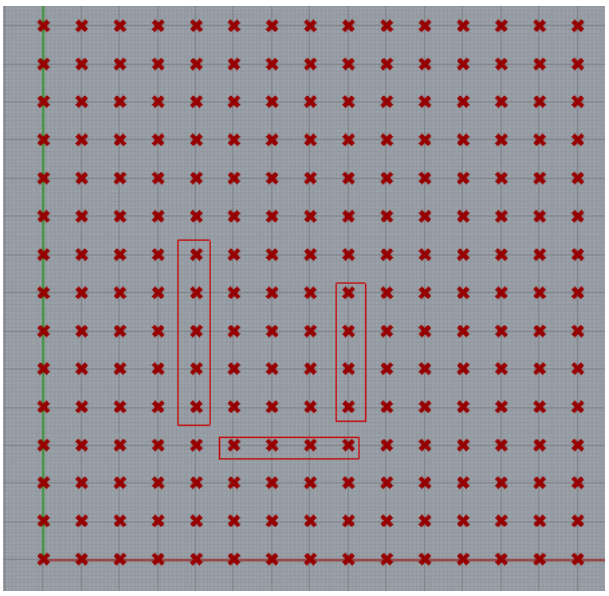


Fig. 8 Capa Active en Rhino. Fuente: Elaboración Propia

En este punto lo que vamos a determinar que los puntos vivos tengan valor 1 y los puntos muertos valor 0. estos serán los pasos.

Se usará el componente "Point in Curves" en Grasshopper, donde se conectarán los puntos y los rectángulos previamente creados en Rhino. Este componente determinará qué puntos están dentro de los rectángulos (vivos) y cuáles están fuera (muertos).

El componente "Point in Curves" en Grasshopper definirá las relaciones de los puntos con respecto a los rectángulos, asignando el valor 0 a los puntos que están fuera (muertos) y el valor 1 a los puntos que están dentro (vivos).

En Grasshopper, el componente "Divide" se configurará con un parámetro numérico de 2 en la entrada B. La entrada A estará conectada al componente "Point in Curves", lo que resultará en una lista de valores 1 y 0 que indican si los puntos están dentro (1) o fuera (0) de los rectángulos definidos previamente. (Fig. 9)

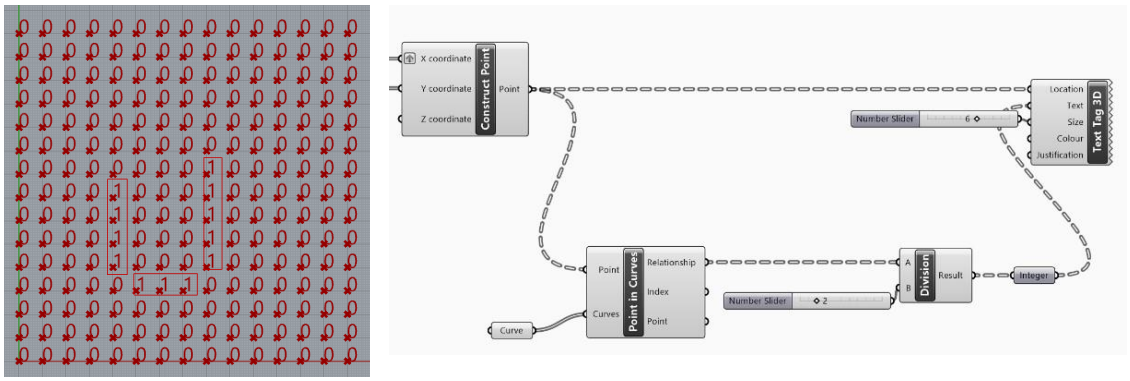


Fig. 9 Establecer valor 0 a puntos muertos y 1 puntos vivos. Fuente: Elaboración Propia

Grupo B: Automatización de la capa Active

Se necesita automatizar la capa active (el patrón que niño realizara)

Buscamos automatizar la activación de la capa "Active" en Rhino, de manera que, al iniciar el trabajo con los rectángulos, esta se active con solo presionar un botón. Para lograrlo, utilizaremos el componente "Layer Reference" en Grasshopper. Conectaremos un botón al puerto "Reference Toggle" y un panel con la palabra "Active" al puerto "Layer", ya que esta es la capa que estamos utilizando en Rhino. De esta forma, al presionar el botón, la capa "Active" se activará automáticamente para su uso. (Fig. 10) (Fig.11)

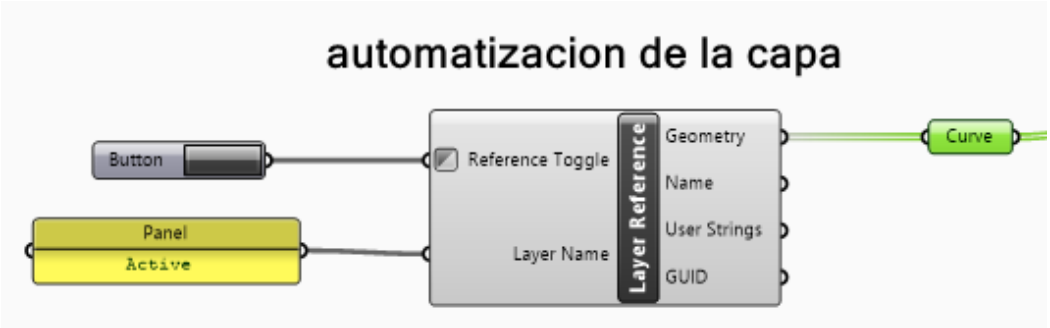


Fig. 10 Automatización de la capa Active. Fuente: Elaboración Propia

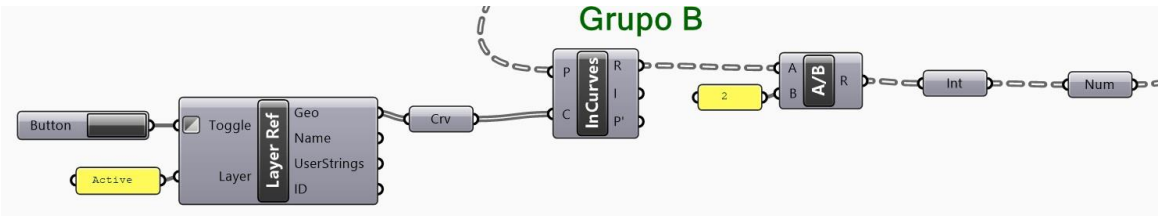


Fig.11 Enfoque de Diseño del Grupo B completo. Fuente: Elaboración Propia

Grupo C: Determinar los vecinos que se encuentra alrededor de puntos

Se necesita establecer los vecinos porque así se determinará cual de estos puntos permanecerán vivos o morirán.

Para realizar el cálculo de vecinos alrededor y obtener las rutas, sigue estos pasos:
Usa el componente Numbers y conéctalo a Shift List.
Calcula la cantidad de vecinos utilizando el componente Tree Branch, asegúrate de simplificar los datos.
Aunque obtendrás los datos deseados, también necesitas las rutas.
Utiliza Tree Statistics y conecta la salida P (Paths) a la entrada L (List) del componente Shift List.
Tree Branch se cloaca Maintain Paths, Este proceso te permitirá tanto calcular los vecinos como obtener las rutas necesarias para el análisis. (Fig. 11)

Para ajustar el cálculo de vecinos de 3,4,5 a 8 vecinos, serán estos pasos:
Usa el componente Tree Branch.
Conecta Tree Branch a la entrada List del componente Shift List.
Añade un componente Number para gestionar la cantidad de vecinos necesarios. (Fig.12)

Para ajustar el cálculo de vecinos de 6,7,8 vecinos, serán estos pasos:
Usa el componente Tree Branch y configúralo con Maintain Paths y Simplify para limpiar las listas.
Repite el paso anterior conectando Tree Branch a dos componentes Shift List y a un componente Number. (Fig. 13)

Para fusionar todos los vecinos en una lista:
Usa el componente Merge y conéctalo a todos los componentes que contienen los vecinos.
Aplica la opción Simplify para una mejor lectura de los datos.
Esto dará como resultado una lista con 225 ramas, cada una con 8 elementos, ya que el diseño de la cuadrícula es de 15x15, totalizando 225 celdas. (Fig.14)

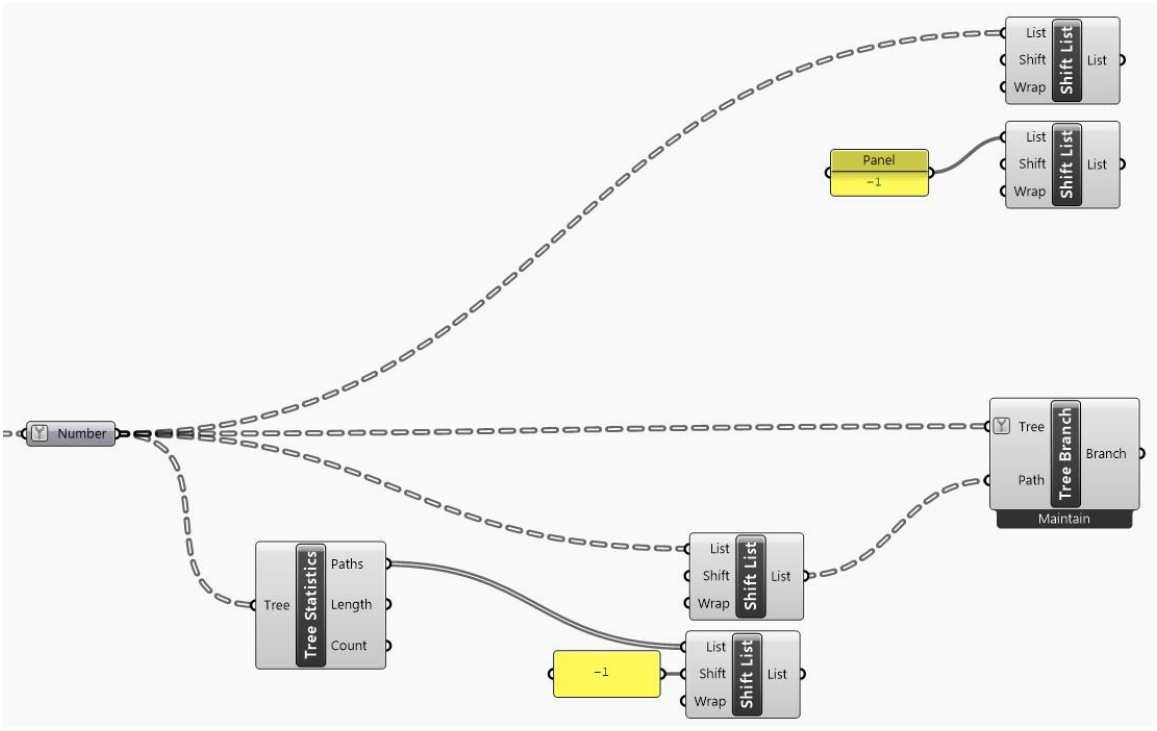


Fig. 11 Calculo 1 y 2 vecino y obtención de rutas. Fuente Elaboración Propia

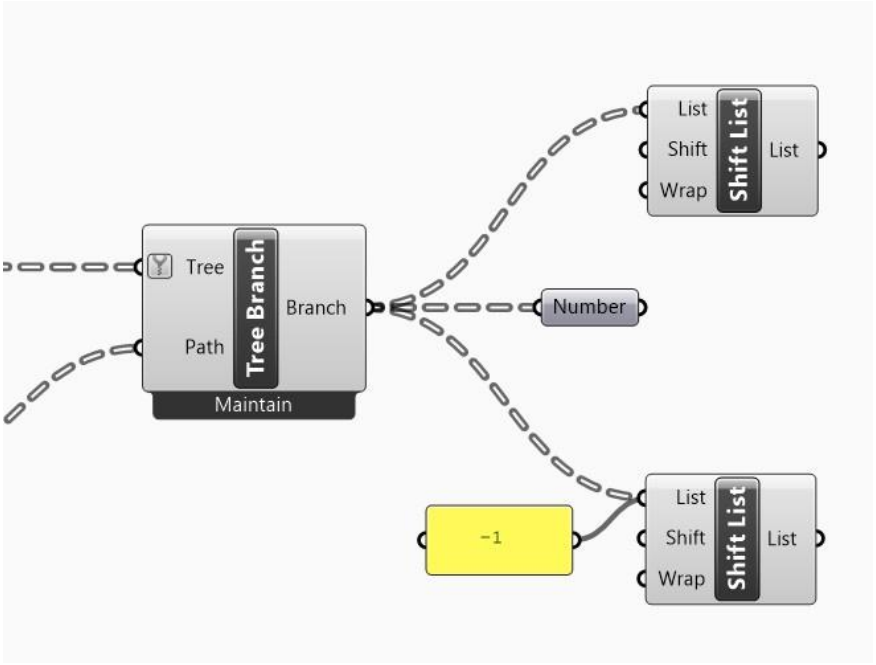


Fig. 12 Calculo 3,4 Y 5 vecinos y obtención de rutas

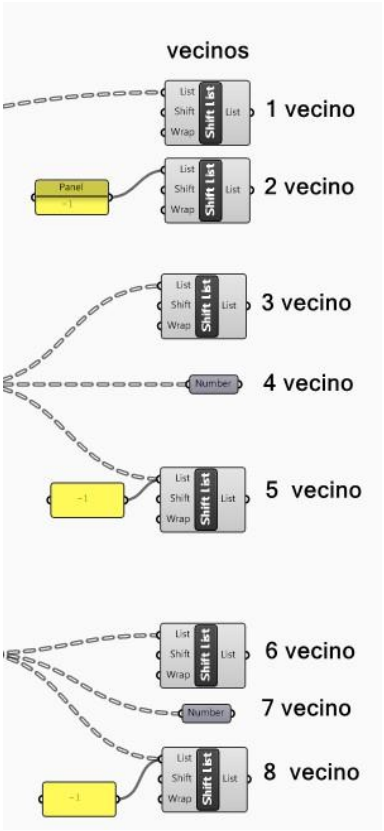


Fig. 13 Obtención de vecinos. Fuente: Elaboración Propia

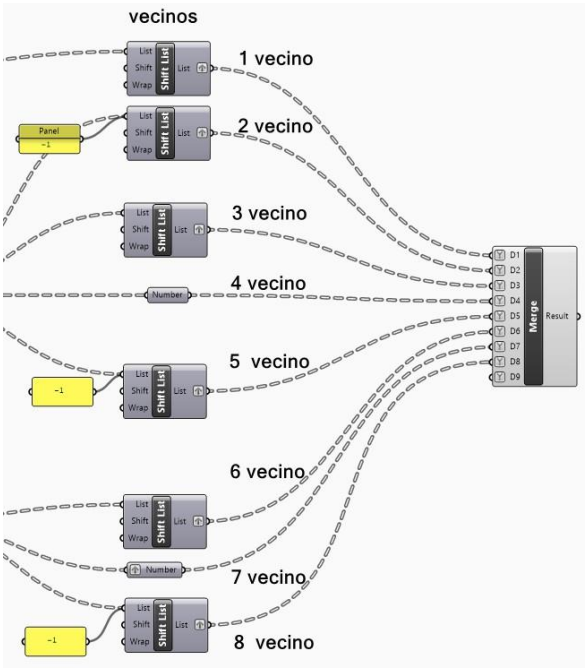


Fig. 14 Fusionar en una sola lista. Fuente: Elaboración Propia

Para determinar cuántos vecinos vivos hay:

Utiliza el componente Mass Addition para sumar los vecinos vivos.

Conecta el resultado de Mass Addition a Trim Tree.

Trim Tree consolidará las ramas más externas del árbol de datos en una sola rama.

Esto te permitirá obtener el total de vecinos vivos de manera consolidada. (Fig.15)

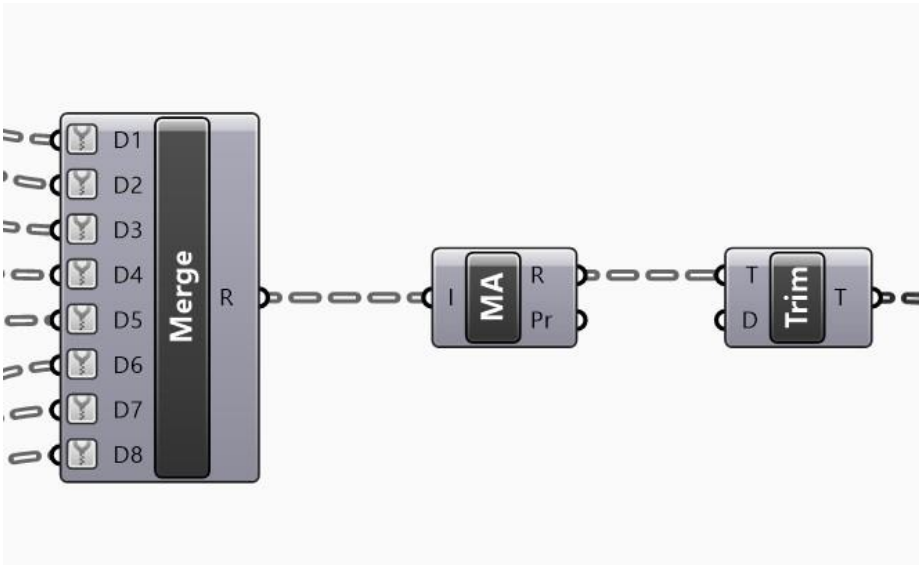


Fig. 15 Determinar el total de vecinos vivos. Fuente: Elaboración Propia

Grupo D: Determinar los puntos vivos y muertos mediante las reglas del juego Conway.

Vamos a comenzar estableciendo las reglas del "Juego de la Vida". En este juego, las celdas de la cuadrícula representan las células vivas o muertas. Las siguientes reglas determinan el estado de una célula en la próxima generación:

Subpoblación: Una célula con menos de 2 vecinos vivos muere en la próxima generación debido a la falta de suficientes células vecinas para sobrevivir.

El componente Trim Tree se conectará con Smaller Than con un parámetro numérico 2. Para determinar la subpoblación. (Fig.16)

Superpoblación: Una célula con más de 3 vecinos vivos muere en la próxima generación debido a la superpoblación, lo que lleva a una competencia excesiva por los recursos.

El componente Trim Tree se conectará con Larger Than con un valor numérico de 3.y conectaremos al componente Gate or . (Fig.17)

Sobrevivencia: Una célula muerta con 2 o 3 vecinos vivos continúa viva para la próxima generación, ya que tiene una cantidad adecuada de vecinos para mantenerse viva. Y conectaremos al componente Gate And. (Fig.18)

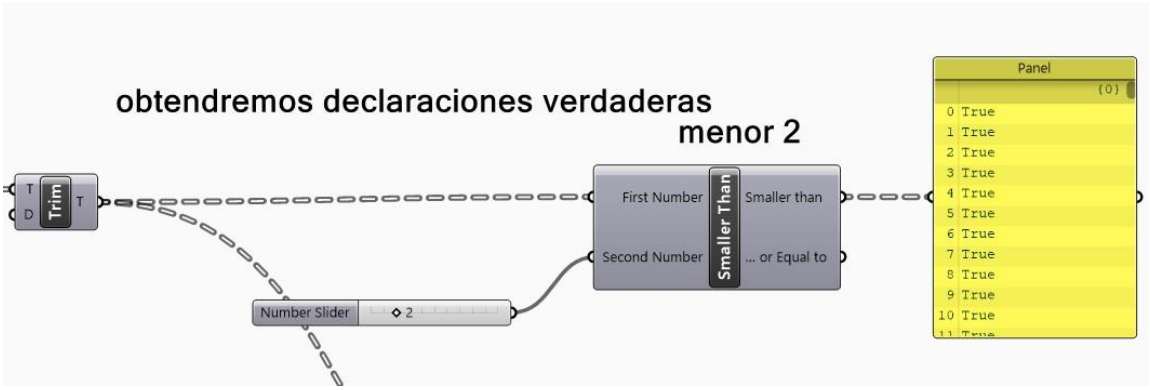


Fig. 16 vecinos vivos mueren por subpoblación. Fuente: Elaboración Propia

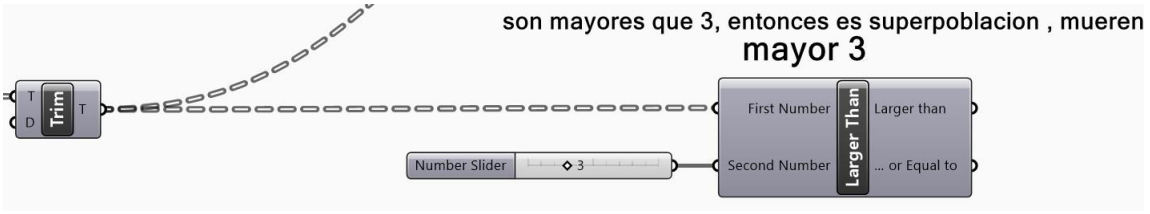


Fig. 17 vecinos vivos mueren por superpoblación. Fuente: Elaboración Propia

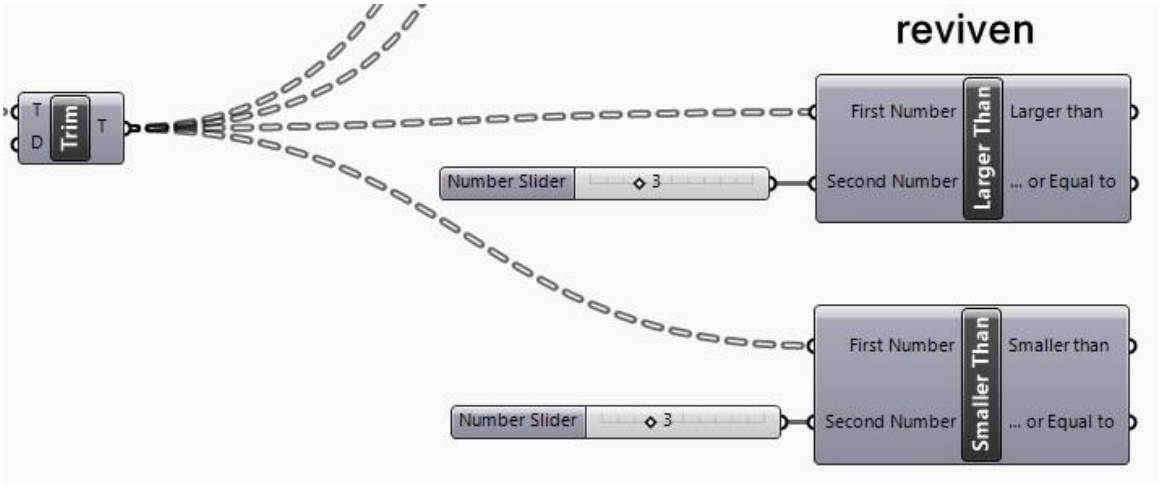


Fig. 18 vecinos vivos reviven. Fuente: Elaboración Propia

Aplicación Anemone

Estas reglas guiarán la evolución de la cuadrícula en cada generación del juego, determinando qué células vivirán y cuáles morirán.

Para implementar un bucle que repita varios pasos y seguir la lógica de nacimiento:

Plugin Anemone: (Fig. 19)

Utiliza el componente Loop Start y conéctalo al componente Loop End para definir el inicio y el final del bucle.

Añade un parámetro numérico y configúralo en 5 para definir el número de repeticiones del bucle.

Usa un componente Button para activar el bucle.

Conecta los datos necesarios para el bucle a la entrada D0. Esto incluye conectar la cuadricula y el componente Construct Point. (Primer grupo de Diseño)

Este proceso permitirá ejecutar el bucle de repetición y seguir la lógica de nacimiento deseada.

Para agregar un nuevo parámetro al bucle usando Anemone:

Loop Start:

Añade un nuevo parámetro al componente Loop Start y denomínalo (0 or 0).

Loop End:

Agrega el mismo nuevo parámetro (0 or 0) al componente Loop End.

Conexiones:

Conecta la salida del nuevo parámetro (0 or 0) del Loop Start al primer componente Number del segundo grupo de diseño.

Luego, conecta la salida del nuevo parámetro (0 or 0) del **L**oop End al segundo componente Number del segundo grupo de diseño. (Fig.20)

Este procedimiento asegura que el nuevo parámetro se utilice correctamente dentro del bucle, conectando los componentes numéricos del segundo grupo de diseño.

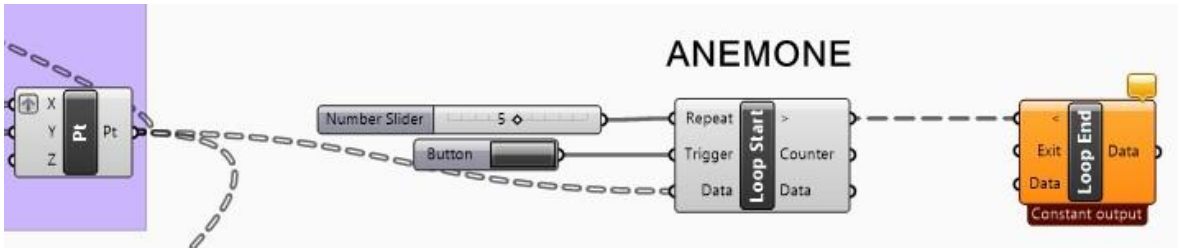


Fig. 19 Plugin Anemone. Fuente: Elaboración Propia

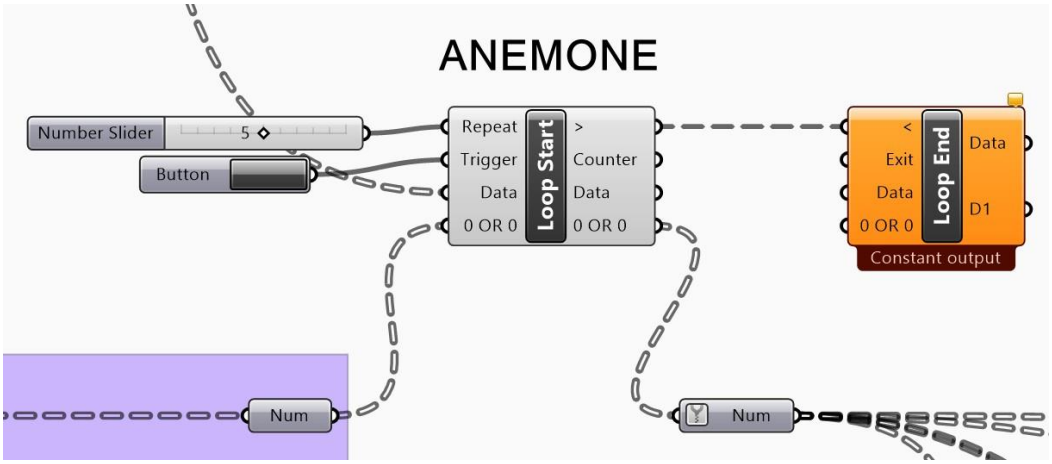


Fig. 20 Parámetro nuevo Anemone. Fuente: Elaboración Propia

Para utilizar el plugin Anemone y gestionar las listas, sigue estos pasos:

Parámetro (0 or 0):

La salida (0 or 0) del plugin Anemone se conecta a un componente Number.

Conexión entre Numbers:

Este componente Number se conecta a otro componente Number recién creado.

List Length:

Conecta este segundo Number al componente List Length para medir la longitud de la lista y obtener un número.

Repeat Data:

Conecta la salida de List Length al componente Repeat Data para generar una lista de ceros.

Pick n Choose:

Usa el componente Pick n Choose y configura sus entradas de la siguiente manera:

Conecta la entrada Gate a la salida (0 or 0).

Conecta Stream 0 al primer componente Number.

Conecta Stream 1 al componente Repeat Data.

Este flujo te permitirá gestionar los datos en el bucle, medir la longitud de las listas y utilizar el componente Pick n Choose para seleccionar entre diferentes flujos de datos. Pero para la parte muerta. (Fig. 21)

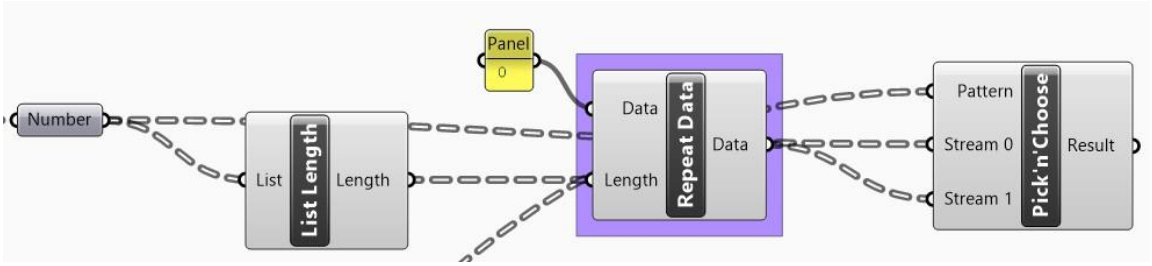


Fig. 21 Pick in Choose . Fuente: Elaboración Propia

Para manejar las posiciones iniciales y determinar las celdas que deben nacer o morir:

Gara And:

Usa el componente Gara And y conéctalo a la entrada P del componente Pick n Choose.

Pick n Choose (paso anterior):

Conecta la salida del Pick n Choose creado en el paso anterior a Stream 0 del nuevo Pick n Choose.

Repeat Data y List Length:

Conecta Repeat Data y List Length, con un parámetro numérico configurado en 1, a Stream 1 del nuevo Pick n Choose.

Conexiones detalladas:

List Length mide la longitud de la lista y obtiene un número.

Repeat Data genera una lista de ceros basada en la longitud.

Configura Repeat Data con un parámetro numérico de 1 para crear una lista de unos.

El nuevo Pick n Choose produce una lista donde la mayoría de los valores son cero (indicando que la mayoría de las celdas murieron), y los valores específicos son uno para las celdas que deben nacer (aquellas con exactamente 3 vecinos vivos).

Este proceso implementa la lógica para determinar la vida o muerte de cada celda en función de sus vecinos vivos, conforme a las reglas del juego. (Fig.22)

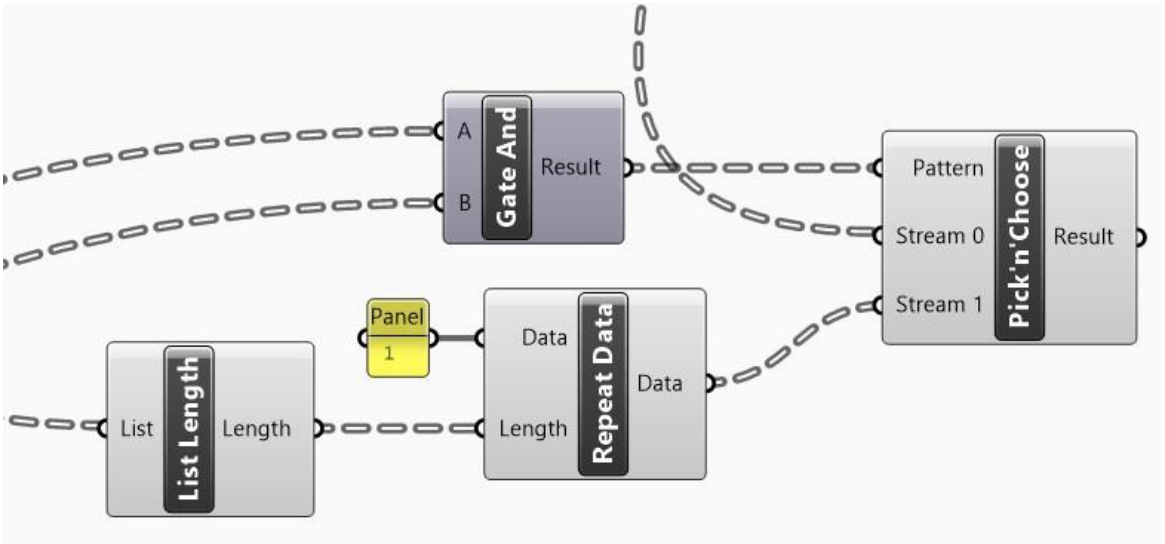


Fig. 22 Determinar celdas que nacen y mueren. Fuente: Elaboración Propia

Para visualizar los puntos que permanecen vivos después de la verificación, sigue estos pasos:

Cull Pattern:

Usa el componente Cull Pattern y conéctalo al Loop Start del bucle ya creado.

Los puntos verdes de la cuadrícula representan los valores que obtienes después de la verificación.

Pick n Choose:

Conecta la salida del Pick n Choose a la entrada P del componente Cull Pattern.

Resultado de Cull Pattern:

El Cull Pattern filtrará los puntos, resultando en una lista de puntos que permanecen vivos.

Cloud Display:

Usa el componente Cloud Display para una mejor visualización de los puntos vivos.

Resultado final:

La visualización final será una representación de los puntos vivos en la cuadrícula, usando Cloud Display para mostrar claramente el resultado.

Este proceso culmina en una visualización clara y efectiva de los puntos que permanecen vivos en la cuadrícula. (Fig.23) (Fig. 24)

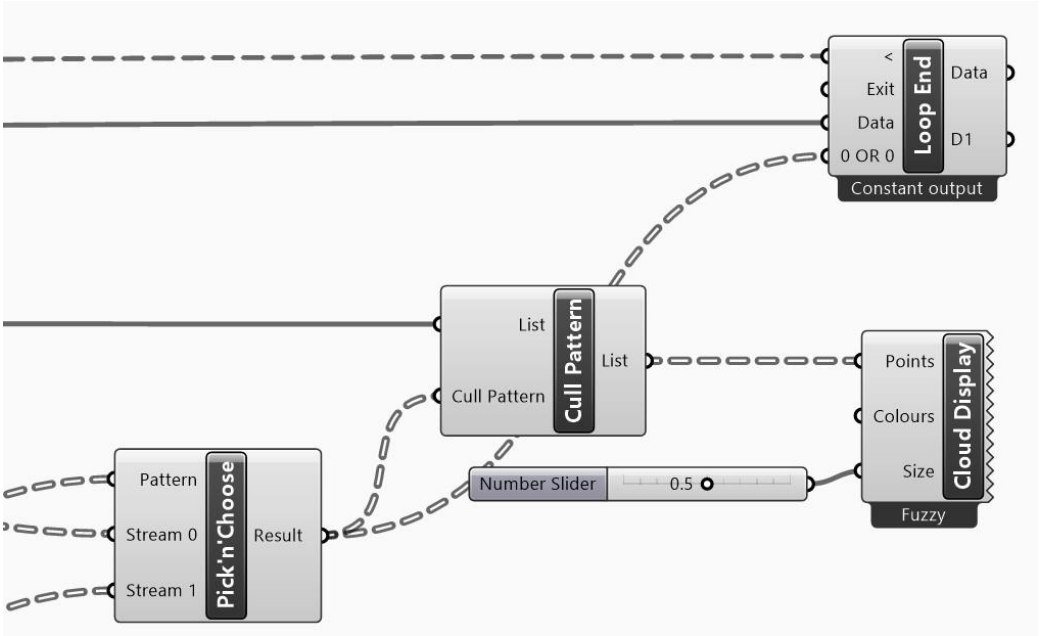


Fig.23 Visualización de puntos vivos. Fuente: Elaboración Propia

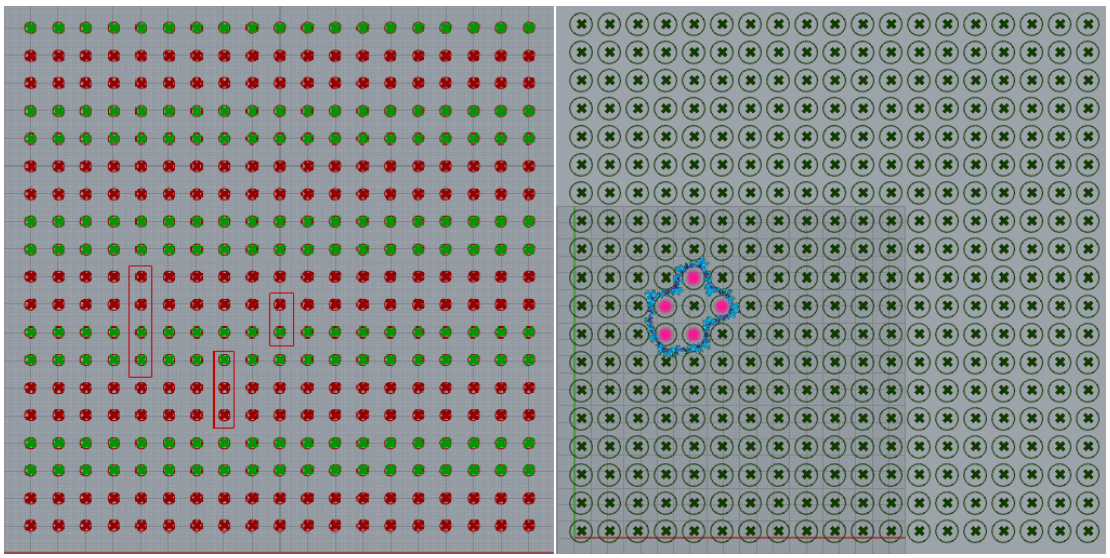


Fig. 24 Los Puntos vivos del sistema. Fuente: Elaboración Propia

4.4 Descripción de la Actividad

En esta actividad, los niños interactuarán con un tablero inspirado en el "Juego de la Vida" de Conway, donde se simulan comportamientos de células vivas y muertas en una cuadrícula. Esta versión incorpora elementos físicos y de agua para una experiencia más inmersiva.

Funcionamiento del Tablero

Puntos Vivos y Agua

En los puntos vivos del tablero, aparecerá agua.

El agua tendrá diferentes formas y alturas según el patrón de puntos encendidos en el tablero.

Interacción con los Niños

Los niños que se hayan colocado en los puntos vivos del tablero se mojarán.

Esos niños estarán fuera del juego temporalmente, permitiendo una rotación y participación de todos los niños.

Proceso del Juego

Inicio del Juego

El tablero se inicia con un patrón aleatorio de puntos vivos y muertos.

Los niños se colocan en los puntos vivos según el patrón inicial.

Evolución del Juego

El tablero actualiza su estado siguiendo las reglas del "Juego de la Vida":

Cualquier célula viva con dos o tres vecinos vivos sigue viva.

Cualquier célula viva con menos de dos o más de tres vecinos vivos muere.

Cualquier célula muerta con exactamente tres vecinos vivos se convierte en una célula viva.

Interacción Activa

Con cada actualización del tablero, los puntos vivos se mojan, afectando a los niños que se encuentran en ellos.

Los niños mojados salen del juego y se incorporan nuevos niños en los puntos vivos del nuevo patrón.

Objetivos Educativos y de Diversión

Visualización del Algoritmo: Los niños visualizan cómo el patrón inicial evoluciona según las reglas del "Juego de la Vida".

Participación: La interacción física con el tablero y el agua hace que la experiencia sea más divertida y memorable.

Aprendizaje Lúdico: Los niños aprenden conceptos de matemáticas y ciencia de manera práctica y lúdica, observando patrones y ciclos. (Fig.25) (Fig.26) (Fig,27)

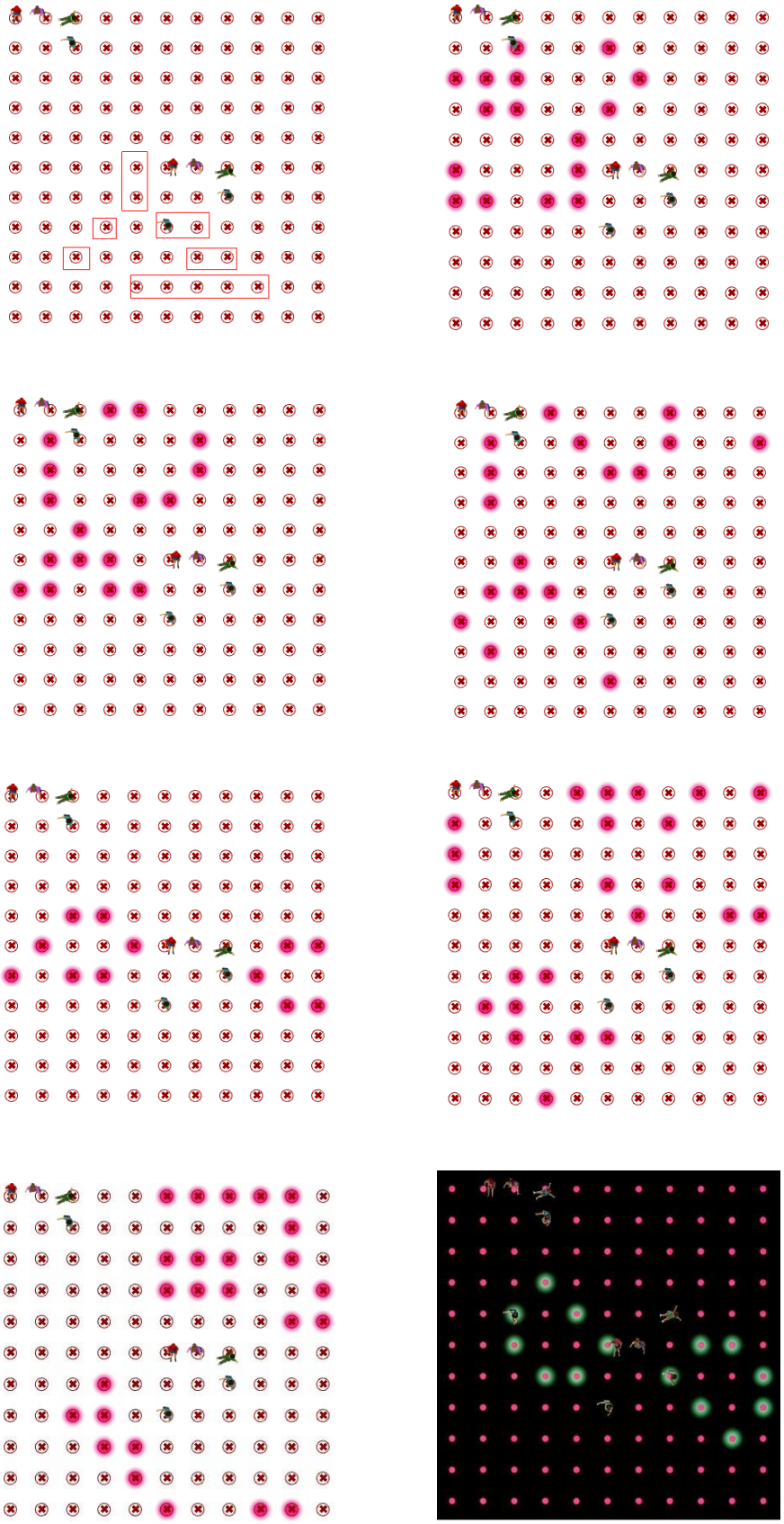


Fig. 25 Patrón inicial y desarrollo del juego conway . puntos vivos. Fuente: Elaboración Propia

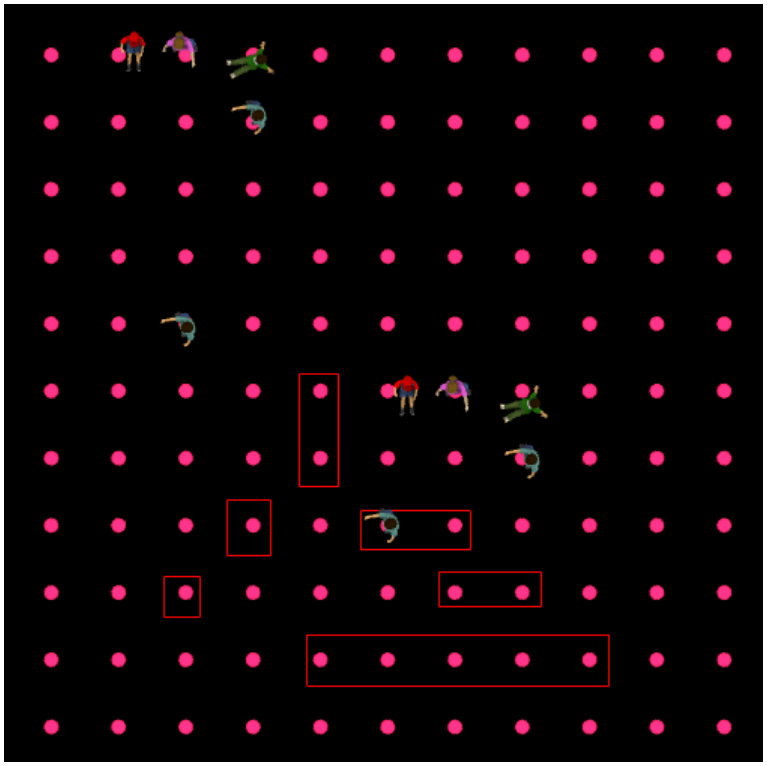


Fig. 26 Patrón inicial lo ejecutara los niños. Fuente: Elaboración Propia

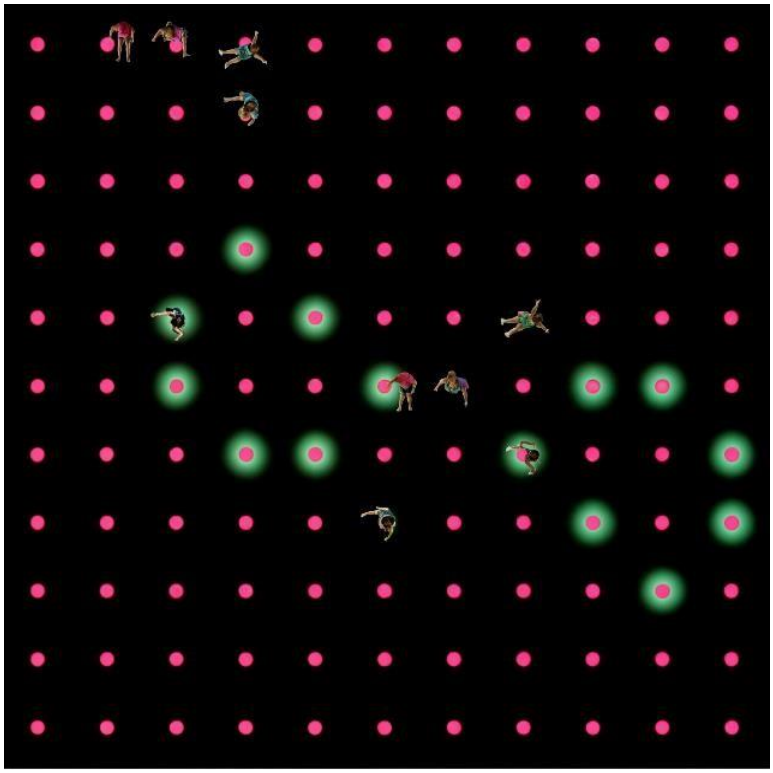


Fig.27 Puntos vivos. Fuente: Elaboración Propia

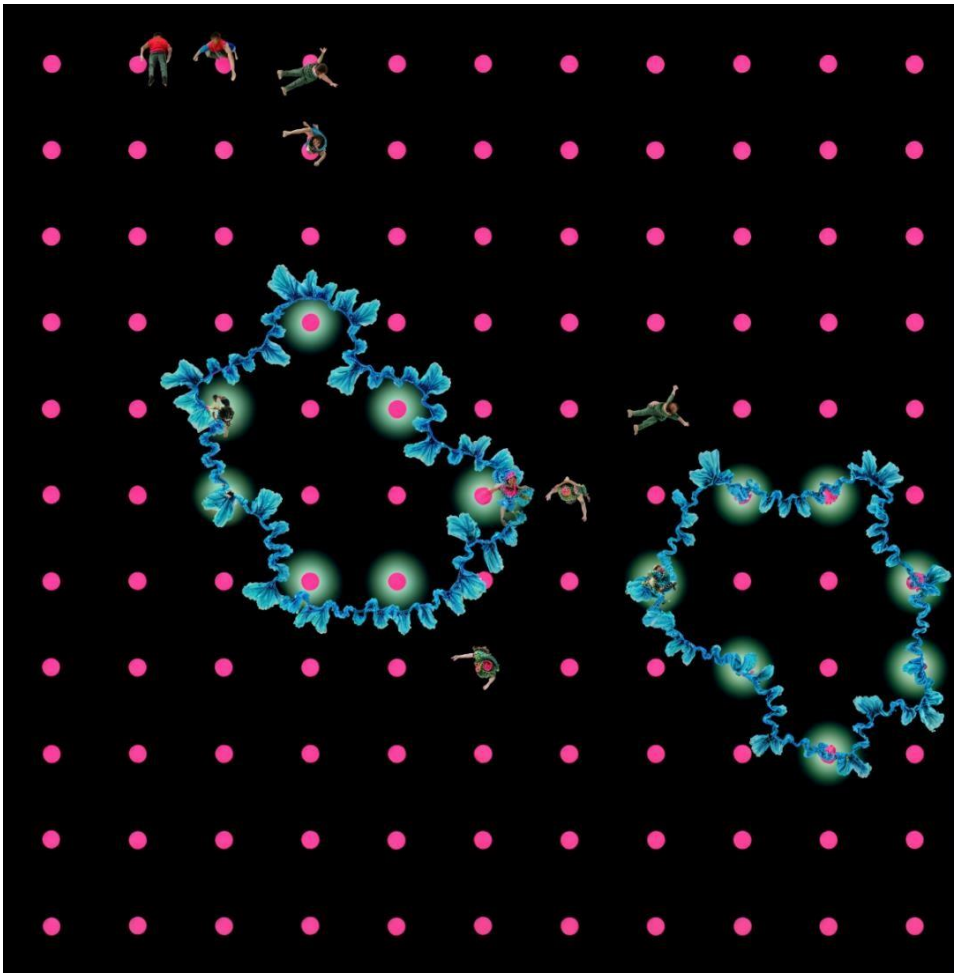


Fig. 28 juego en funcionamiento, niños colocados en los puntos vivos, perderán el juego, Fuente: elaboración Propia

Fig. 29 QR. Simulación del juego infantil. Fuente: Elaboración Propia

Fig. 30 Render del juego infantil en el Oriente Ecuatoriano . Fuente: Elaboración Propia

Fig. 31 Render del juego infantil en la Costa ecuatoriana. Fuente : Elaboración Propia

MATERIALIDAD

El juego diseñado es un tablero donde el niño dibujará su patrón. A partir de este paso, comenzará a correr un algoritmo diseñado por autómata celular. Este algoritmo encenderá unas luces, representando los puntos vivos. Si el niño se para en un punto y este se enciende, perderá automáticamente y la respuesta será un chorro de agua. Ahora, queremos una respuesta más sofisticada: cortinas de agua.
Estas cortinas de agua se diseñarán usando un algoritmo de crecimiento diferencial, similar al que vimos en el capítulo 3. Lo que variará es que los puntos que se encienden formarán una curva. A partir de esa curva, aplicaremos el diseño del algoritmo diferencial y obtendremos esas formas de fantasía inspiradas en el coral.

La materialidad del juego es el agua. Aún no se ha definido el material del tablero, ya que necesitaríamos consultar con expertos en electrónica, programación e hidráulica para obtener los lineamientos necesarios para construir el tablero y asegurar su funcionamiento seguro y eficiente, considerando que el chorro de agua es una parte integral del juego.

CAPITULO 5

CONCLUSIONES

Estudiar algoritmos en sistemas de crecimiento, como los que rigen la formación de estructuras biológicas (por ejemplo, corales, árboles, hongos), permite a los investigadores comprender de manera detallada los procesos naturales subyacentes. Estos algoritmos describen cómo organismos vivos crecen y se desarrollan en respuesta a su entorno, proporcionando un modelo matemático y computacional para fenómenos biológicos complejos.

El desarrollo y la conclusión de esta metodología, aplicada al caso de estudio del coral, amplían nuestro entendimiento del modelado paramétrico gráfico, mostrando la relación entre la ingeniería de software y la conexión de varias disciplinas. Este argumento se apoya en las relaciones que surgen de los cambios, la investigación, los métodos y prácticas de diseño compartidos. Estas conexiones posicionan a la ingeniería de software como un precedente importante para arquitectos y diseñadores, demostrando cómo el modelado paramétrico gráfico puede integrarse en la práctica diaria de un arquitecto profesional y cómo cada persona involucrada en las disciplinas de diseño percibe el modelo paramétrico.

5 CONCLUSIONES

5.1 Analogía entre Juegos y Normas (Algoritmos)

Los juegos y las normas (o algoritmos) comparten muchas similitudes, ya que ambos estructuran un conjunto de reglas y procedimientos que guían el comportamiento y las acciones dentro de un sistema. A continuación, se presenta una analogía detallada entre juegos y normas (algoritmos):

Juegos

Los juegos presentan reglas definidas que se determina como se juega y que acciones se permiten o son prohibidas. Por ejemplo, el ajedrez, cada pieza tiene un conjunto especifico de movimientos. Casi siempre los juegos presentan un objetivo claro que los jugadores intenta alcanzar. También los jugadores interactúan entre si y con el entorno del juego, tomando decisiones basadas en reglas. Un ejemplo claro son las cartas, los jugadores deberán decidir que cartas jugar en cada turno. Los juegos presentan acciones, pero estas acciones son llevadas a cabo mediante turnos.

Normas (Algoritmos)

Los algoritmos consisten en un conjunto preciso de instrucciones que especifican como deben realizarse las tareas. Estos están diseñados para alcanzar un objetivo o resolver un problema específico. Los algoritmos interactúan con datos de entrada y producen resultados basados en esas interacciones. También siguen una secuencia lógica de pasos ordenados para completar una tarea.

Como conclusión en esta investigación, tanto como los juegos como las normas (algoritmos) se basa en un conjunto estructurado de reglas e instrucciones que guían las acciones hacia un objetivo claro. Ambos sistemas requieren la interacción con elementos externos (jugadores o datos) y siguen una secuencia definida de pasos. Además, los resultados pueden variar dependiendo de las decisiones tomadas y las condiciones especificadas de cada caso. Esta analogía subraya la naturaleza sistemática y ordenada de los juegos y los algoritmos, destacando similitudes en términos de estructura, objetivos, interacción y secuencia de acciones.

5.2 La importancia del PROCESO en los juegos como en algoritmos

El proceso en los juegos y los algoritmos es fundamental porque determina cómo se llega a un resultado final, ya sea una victoria en un juego o la solución de un problema en un algoritmo. La importancia del proceso se manifiesta en varios aspectos clave:

1. **Optimización y Eficiencia**

Juegos

En los juegos, el proceso de tomar decisiones estratégicas es esencial para optimizar las posibilidades de ganar. Los jugadores deben planificar sus movimientos y prever las posibles respuestas de sus oponentes. Este proceso no solo incluye la ejecución de acciones, sino también la adaptación a situaciones cambiantes, lo que puede involucrar una estrategia dinámica y flexible.

Algoritmos

En algoritmos, el proceso se refiere a la secuencia de pasos lógicos diseñados para resolver un problema de la manera más eficiente posible. Un algoritmo bien diseñado debe minimizar el tiempo y los recursos necesarios para obtener un resultado. La eficiencia del proceso algorítmico es crucial en aplicaciones donde el tiempo de ejecución y el uso de recursos son limitados.

2. **Correctitud y Precisión**

Juegos

Un proceso preciso y bien estructurado en los juegos asegura que las reglas se siguen adecuadamente y que los resultados son justos. La correctitud en la aplicación de las reglas y estrategias garantiza que el juego sea equilibrado y que los jugadores tengan una experiencia justa y competitiva.

Algoritmos

Para los algoritmos, la correctitud es vital para asegurar que el algoritmo resuelve el problema correctamente cada vez. Esto implica que el proceso debe estar bien definido y probado para asegurar que, independientemente de las variaciones en los datos de entrada, el algoritmo produce los resultados esperados sin errores.

3. **Aprendizaje y Mejora Continua**

Juegos

El proceso en los juegos también es importante para el aprendizaje y la mejora continua. A través de la repetición y la reflexión sobre las partidas jugadas, los jugadores pueden identificar sus errores, aprender nuevas estrategias y mejorar su rendimiento. Este ciclo de retroalimentación es crucial para el desarrollo de habilidades a largo plazo.

Algoritmos

En el contexto de algoritmos, el proceso de desarrollo incluye la iteración y la mejora continua. Los desarrolladores y científicos de datos deben analizar el rendimiento de sus algoritmos, identificar áreas de mejora y refinar los pasos del proceso para aumentar la eficiencia y la exactitud. Este ciclo de iteración y optimización es esencial para el avance en campos como la inteligencia artificial y el aprendizaje automático.

.

En conclusión, tanto en los juegos como en los algoritmos, el proceso es crucial para asegurar eficiencia, correctitud, aprendizaje continuo, adaptabilidad y satisfacción del usuario. Entender y optimizar el proceso permite obtener mejores resultados y una mayor experiencia tanto en el juego como en la resolución de problemas algorítmicos.

Resultados inesperados

destacando cómo la implementación de algoritmos en el diseño de juegos puede generar resultados inesperados y sorprendentes. Los hallazgos de este estudio revelan que los algoritmos no solo proporcionan estructura y reglas claras dentro de los juegos, sino que también pueden introducir elementos de imprevisibilidad que enriquecen la experiencia de los niños.

Los algoritmos, al ser aplicados en el contexto de los juegos infantiles, permiten la creación de escenarios dinámicos y adaptativos que responden a las acciones de los jugadores de manera no lineal. Esta capacidad de adaptación y evolución constante de los juegos potencia el desarrollo cognitivo, creativo y social de los niños, ya que los enfrenta a situaciones nuevas que requieren pensamiento crítico y resolución de problemas en tiempo real.

Uno de los resultados más significativos de esta investigación es el reconocimiento de que los algoritmos pueden producir comportamientos emergentes dentro de los juegos, donde patrones complejos y novedosos surgen de la interacción de reglas simples. Estos comportamientos emergentes contribuyen a que cada sesión de juego sea única, manteniendo el interés y la motivación de los niños.

Además, se ha constatado que la integración de algoritmos en los juegos infantiles puede ser una herramienta poderosa para personalizar la experiencia de aprendizaje y juego, adaptándose a las necesidades y habilidades individuales de cada niño. Esta personalización facilita un aprendizaje más efectivo y una experiencia de juego más gratifica

LIMITACIONES Y DESAFIOS ENCONTRADOS EN LA METODOLOGIA

Antes de empezar a codificar, asegúrate de entender completamente el problema. Escribe una descripción detallada y los requisitos del problema. En este caso, se estudió el coral Acropora, y paralelamente se comenzó a investigar qué plugins podrían ser útiles, como Anemone. Es importante entender cómo funciona el plugin y cuáles son los requisitos para su uso. En la actualidad, acceder a la información es más fácil gracias a Internet; un clic es suficiente para obtener datos. Sin embargo, la verdadera dificultad reside en comprender esa información. A través de prueba y error, se puede mejorar el entendimiento de cómo funciona un plugin y qué es capaz de hacer.

Una gran sorpresa me llevé al usar un componente en Grasshopper que arrojaba un error. Aunque el programa indicaba cuál era el problema, no lograba entenderlo. Al poner ese mensaje en ChatGPT, este me lo traducía en palabras simples, lo que mejoraba mi comprensión de lo que estaba ocurriendo. Así avanzaba más rápido. Mi gran aliado fue ChatGPT (inteligencia artificial). Así fue en todos los diseños de los algoritmos. Además, fui encontrando blogs y personas de diferentes profesiones que se dedican al diseño algorítmico; de ellos también aprendí. Más que aprender, fueron mi inspiración.
Conocí más programas y me quedé con más curiosidad de aprender. Por ejemplo, en algo tan básico como cambiar el color del canvas en Grasshopper, tuve que hacer un algoritmo en programación. Aunque antes evitaba la programación porque la veía muy complicada, ahora me resulta muy interesante y estoy aprendiendo.

Como una conclusión personal, me doy cuenta de que no importa la edad que tenga, se puede aprender donde quieras y con lo que tengas; solo se necesita tenacidad. La tecnología será mi aliada al momento de diseñar. Lo que aprendí hoy posiblemente no será el conocimiento del mañana; todos debemos aprender, desaprender y volver a aprender.

Índice de Figuras

CAPITULO 2

CAPITULO 3

CAPITULO 4

Bibliografía

Arnheim, R. (1971). *Entropia , ensayo sobre el desorden y orden.* publicado University of California Press.

Carpo, M. (2012). Darwinismo digital: colaboracion masiva, busqueda de formas y disolucion autoria . 18.

Emily Kennedy, D. (2002). Un camino hacia la innovacion sostenible.

Fisch, M. (2017). La naturaleza del biomimetismo. *Sage Publications, Inc*, 30.

Hanna, S. (2017). Artesania Oxman .

Liebert, M. A. (2016). Cultivado,impreso aumentado biologicamnete . *Escuela de Arquitectura y Planificacion de Massachusetts*, 19.

Mortati, M. (2017). Nuevos conocimientos de diseño.

Osorio, U. R. (21 de julio de 2023). *Ecologia Verde*. Obtenido de https://www.ecologiaverde.com/tipos-de-corales-4481.html

Simon, F. l. (2000). Que saben, predicen y transmiten los animales y plantas.

system, n. (22 de 06 de 2015). *nervous system*. Obtenido de nervous system: https://n-e-r-v-o-u-s.com/blog/?p=6721

Vivia Otalvaro Guzman, B. I. (2012). Sostenibilidad y Mimesis, Analogia del Diseño.

Arnheim, R. (1971). *Entropia , ensayo sobre el desorden y orden.* publicado University of California Press.

Carpo, M. (2012). Darwinismo digital: colaboracion masiva, busqueda de formas y disolucion autoria . 18.

Emily Kennedy, D. (2002). Un camino hacia la innovacion sostenible.

Fisch, M. (2017). La naturaleza del biomimetismo. *Sage Publications, Inc*, 30.

Hanna, S. (2017). Artesania Oxman .

Liebert, M. A. (2016). Cultivado,impreso aumentado biologicamnete . *Escuela de Arquitectura y Planificacion de Massachusetts*, 19.

Mortati, M. (2017). Nuevos conocimientos de diseño.

Osorio, U. R. (21 de julio de 2023). *Ecologia Verde*. Obtenido de https://www.ecologiaverde.com/tipos-de-corales-4481.html

Simon, F. l. (2000). Que saben, predicen y transmiten los animales y plantas.

system, n. (22 de 06 de 2015). *nervous system*. Obtenido de nervous system: https://n-e-r-v-o-u-s.com/blog/?p=6721

Vivia Otalvaro Guzman, B. I. (2012). Sostenibilidad y Mimesis, Analogia del Diseño.

E. Haeckel, *Art Forms in Nature*, Dover Publications Inc.; Revised edition (2 Jan 2000).

D.W. Thompson, *On Growth and Form*, Cambridge University Press; New Ed edition (31 July 1992).