Premio Investigación Interuniversitaria Multidisciplinaria – 2015

Universidad de Lima – PUCP

Algoritmos y gramáticas generativas para el estudio de los patrones del diseño precolombino

Dr. Umberto Luigi Roncoroni Osio. Facultad de Comunicación, Universidad de Lima.

Dr. Veronica Natalie Crousse de Vallongue Rastelli, Facultad de Arte, PUCP.

Milán-Lima Enero 2016

Índice

Presentación

Cap. 1. Patrones formales en la naturaleza y en el diseño precolombino

1.1 Relación entre naturaleza, forma, diseño y gramáticas generativas
1.2 Fichas comparativas
1.3 Dibujos algorítmicos a mano alzada

Cap. 2. Programación con procesos de sustitución recursiva: símbolos programables para Sistemas-L.

2.1 Software
Introducción
Ventana de diseño
Ventana gramática
Utilidades

Cap. 3. Programación con Sistemas_L

Introducción
1. Símbolos para el control de flujo
2. Símbolos automáticos
3. Símbolos para los procesos aleatorios
4. Fórmulas y datos
5. Subsistemas (*nested* L-Systems)
6. *Image rules*
7. Símbolos especiales
8. *Símbolo recursivo "s"*

Cap.4. Programación con Sistemas_L y diseño precolombino. Moray.

Introducción

Anexo 1. Diseño de reglas

Anexo 2. Paper ArtNodes

Bibliografía general

Presentación

Este documento es el informe final del proyecto de investigación Patrones formales en la naturaleza y en el diseño precolombino *Patrones formales en la naturaleza y en el diseño precolombino*. El objetivo de la investigación es el uso de las gramáticas generativas para el estudio de los patrones de diseño precolombinos. Pero esto se puede ver en modo simétrico: es el estudio del diseño precolombino que sirve para el desarrollo de las gramáticas generativas.

Metodología

Primera parte artística, segunda parte computer science. Todo enlazado por la reflexión filosófica, sobretodo la parte informática. Resaltar el carácter interdisciplinario del proyecto y de los resultados.

Resultados del proyecto

Finalizando el proyecto se ha conseguido:

1. Materiales y documentos para artículos científicos, poster o participación a congresos. Un artículo ya ha sido enviado a la revista ArtNodes (ver anexo) de la Universidad Abierta de Cataluña (en fase de evaluación por pares).
2. Materiales y documentos para el congreso y la ponencia está listo y se enviará para el Congreso de GenerativeArt del Politecnico de Milán en Diciembre 2016. Este material está documentado en el cap 1y 2 del presente documento y en el CDROM anexo.
3. Materiales y documentos para un congreso y exposición. Dicho evento está programado para el 2017 en el Centro Cultural de Lima de la Telefónica. Además de la exposición, se ha programado la participación de grupos de investigación de del Grupo Hipertrópico de la Universidad de Antioquia de Medellín Colombia. y se ha planteado la publicación de las actas en forma de libro o catálogo, por el cual se utilizaran los fondos del presupuesto del proyecto. Posiblemente este evento se repetirá en la Universidad de Antioquia. Parte de este material (falta

completar algunos dibujos e impresión en grande formato) está documentado en el cap. 1 y 2 del presente documento y en el CDROM anexo.

4. *Software* aplicativo. Se ha desarrollado una aplicación para el desarrollo interactivo de gramáticas generativas basadas en los Sistemas-L. Este material está en parte documentado en este informe; el código fuente y el ejecutable se encuentran en el CDROM anexo y en la página web http://www.digitalpoiesis.org

Informe final

El informe consta de 2 partes. En el cap. 1 se documenta la investigación visual sobre las formas naturales y el arte precolombino, estudio que finaliza en una base de datos de documentos fotográficos hechos por los autores en sitios arqueológicos, reservas naturales y museos, y fotografías al microscopio de varios especímenes naturales y artificiales.

En el capítulo 2 se reporta el proceso y los documentos del desarrollo de las gramáticas generativas y las características de la resultante aplicación informática. Se documentan aquí los algoritmos, las interfaces, las pruebas y los ejemplos.

En el anexo se encuentra el *paper La virtualidad aumentada: procesos emergentes, arte y medios digitales.*

Cap. 1. Documentación visual

1.1 la relación entre naturaleza, forma, diseño y gramáticas generativas

El tópico central del problema de la forma generativa se refiere a la sustentación, comprensión y formalización de los procesos naturales en relación al arte y a la computación. El término generativo indica un elemento esencial del concepto de forma, que es el potencial de estos procesos en cuanto herramientas creativas que pueden ser utilizadas en modo original en contextos diferentes y con una pluralidad de aproximaciones metodológicas.

El poder de la herramienta informática –las gramáticas generativas, *shape grammars* y Sistemas-L- está precisamente en la flexibilidad operativa, aprovechando de la interactividad y de la capacidad del lenguaje digital de formalizar y computar los procesos naturales y culturales (lo precolombino, en este caso).

En este apartado lo que se intenta sustentar los principios formales subyacentes a las formas naturales, al arte precolombino y a los algoritmos generativos. El trabajo de investigación se ha organizado en estas etapas: documentación de las formas naturales, comparación con el arte precolombino, análisis de las formas por el dibujo "procedural" a mano alzada, implementación en gramáticas generativas, desarrollo de modelos 2D y 3D.

Como se explica la relación simbólica y el valor místico de las formas? En las fichas comparativas se investiga precisamente este problema. Tomando en cuenta Einstein: "*ningún camino lógico lleva a estas leyes originales: solo la intuición, apoyada en la experiencia, puede conducirnos a ellas*". (1975, p. 39).

- Lo que es original es global (lo que da origen se expande, es generativo)
- Forma como sistema coherente de hipótesis y procesos (se aplica a diferentes contextos)
- Analogía (similitud de funciones)
- Homología (similitud de estructuras)

En este apartado se explica los elementos y los principios que posibilitan una relación estructural entre estos diferentes contextos formales. Es decir, aquello que permite plantear un hilo conductor, una sintaxis común capaz de ser formalizada.

Forma y arquetipos

La relación entre formas, naturaleza, mitos y arquetipos es el primer problema que es preciso investigar. El arquetipo puede explicar lo que hay en común entre dominios tan diversos. Ahora, el concepto de arquetipo puede tener diferentes significados. El místico puede concebir a los arquetipos como dioses o diosas codificados en el inconsciente colectivo, a quienes no podemos olvidar sin costo ni riesgo. Los arquetipos son algo así como las diferentes facetas de ese Dios, accesibles a la capacidad de la psiquis de imaginar realidades sagradas. Los científicos pueden concebir los arquetipos como paradigmas o metáforas que ejercen control, los patrones invisibles de la mente que controlan cómo experimentamos el mundo. Los artistas pueden concebir a los arquetipos como formas simbólicas recurrentes que se encuentran en todas las culturas: espirales, ramificaciones, teselaciones, etcétera.

Para el presente proyecto, la cosa interesante es que el arquetipo, en términos informáticos, puede ser traducido en reglas, la regla es el arquetipo formal (un proceso arquetípico) en cuanto genera una familia de formas que de este arquetipo conserva obviamente las características estructurales. Además, esto lleva a reflexionar, en sentido generativo, sobre varios aspectos de la forma y sus interpretaciones: la relación forma función, el estructuralismo, la *Gestalt*.

Las formas de la naturaleza.

Los principios de las formas naturales han sido investigadas por Wagensberg (2003), Stevens (1986), D'Arcy Thomson (1961) y en sentido computacional por Flake (1998). La relación se basa en principios funcionales y topológicos. La espiral tiene característica de longitud y uniformidad, la ramificación relaciona con el centro y rellena el espacio, el meandro ocupa el espacio, la explosión conecta con el centro… cada una tiene ventaja y desventaja que dependen del espacio en lo que ocurre un cierto fenómeno, las características de la materia y del contexto. Cada solución que adopta la

naturaleza responde a optimizar estos parámetros. Igual sucede con las formas del arte (la arquitectura precolombina), es una explicación funciona.

Igual sucede con la programación. Los algoritmos pueden seguir formas lineales o no lineales, directas o indirectas, cortas o largas. Pero el aspecto topológico funcional no es el único que nos permite entender las relaciones entre las formas.

Lo precolombino y la cosmovisión

Los precolombinos crean sus procesos en base a una cosmovisión. Las formas precolombinas describen la expresión de las leyes y de los principios del cosmos. La intervención en el territorio responde a esta cosmovisión y a las formas naturales.

Pero esto nos relaciones con los arquetipos y las formas naturales y se introduce el concepto de ritmo. Se con Di Napoli (2011): "*diciamo le forme, transeunti ma ricorrenti, "diagrammi di forze", ne vediamo le pulsazioni, le confluenze vettoriali. Nessuna forma nasce dal nulla, ma si-forma nel tempo, è un processo che inizia in un punto (seme-ritmo), termina nella sua de-formazione, e ricomincia – ritmo come irraggiamento, eco, affinità, contrasto, crescita (e decrescita)*".[1]

Cómo han señalado Stevens, Wagensberg, Thomson, Flake y otros más, los diagramas de fuerzas se basan en principios de adaptación, morfogénesis, evolución, relaciones con un mismo contexto.

Las fichas comparativas abren a la intuición los principios subyacentes a la naturaleza y al arte. Aquí hay que señalar lo importante de lo precolombino (primitivo, neolítico, original) en cuanto exento de contaminaciones estéticas conscientes, es decir, libres de establecer conexiones arquetípicas con las formas naturales.

Desde el punto de vista científico, que a pesar del privilegio de la intuición, es útil considerar (por sus reflejos concretos en la *computer science*), el sentimiento humano de la forma se puede explicar en base al estructuralismo o a la *Gestalt*.

[1] Decimos que las formas efímeras pero recurrentes son diagramas de fuerzas de los que vemos las pulsaciones, y la convergencia de sus vectores. Ninguna forma nace de la nada, sino que se hace, se deshace, y se trasforma. Ritmo como radiación, resonancia, afinidad, crecimiento y decrecimiento.

Estructuralismo

Respecto a la constancia perceptiva, los estructuralistas habían supuesto una correspondencia puntual perfecta, que relacionaba una sensación con un estímulo particular (hipótesis de la constancia) y, por consiguiente, un cambio en el estímulo daba lugar a un cambio sensorial-experiencial. Lo percibido (experiencial) resultaba inferido a partir del núcleo de sensaciones correspondientes a la imagen retiniana, tras repetidas experiencias en las que tales sensaciones se asocian a otras, derivándose un aprendizaje. La interacción inconsciente entre elementos y asociaciones de sensaciones, las cuales poseen un contexto elaborado que le otorga significado, hacen que la percepción nos parezca constante, idéntica a pesar de las variaciones sensoriales.

Gestalt

En cambio, para los Gestaltistas, la constancia es primaria, se da en la experiencia inmediata del perceptor y, por consiguiente, no debe ser algo secundario a explicar, sino que son los elementos o partes constituyentes los que son puras abstracciones teóricas, en cuanto no se hallan presentes en la experiencia fenomenológica. En otros términos, para los estructuralistas los datos primarios son los elementos y lo secundario, obtenido por aprendizaje asociativo, es la constancia del tamaño, forma, color, etc.; mientras que, para la psicología de la Gestalt la experiencia fenomenológica de la constancia es el dato primario organizado y estructurado, los elementos son derivaciones secundarias segregadas por abstracción. El principio básico de la organización perceptual es que el todo es más que la suma de las partes, es decir, que las propiedades de la totalidad no resultan de los elementos constituyentes, sino que emergen de las relaciones espacio-temporales del todo.

Dibujo a mano alzada

El dibujo a mano alzada recoge esta complejidad espontánea y emergente.

En el automatismo del trazo emerge, intuitivamente, el ritmo, el arquetipo, al modo del dibujo automático de los surrealistas[2].

Pero el dibujo también puede ser un instrumento analítico, pues en un segundo momento la espontaneidad se conjuga con la regla, descubre una regla. "L'arte non ripete le cose visibili, ma rende visibile" è la *Confessione creatrice* di Paul Klee (1976). E' la *forma immateriale* di un oggetto che l'arte 'imita', non la cosa percepita dai sensi. Dante: "*Chi pinge figura, si non può esser lei, non la può porre*" (*Convivio*, III, 53). Ogni immagine discende da forme formanti (*ritmi*) che il cuore dell'uomo riflette contemplandole: l'esperienza estetica indiana è contigua all'esperienza estatica – all'accordo (*samstava*) fra il cielo e il cuore, nell'indivisibile punto d'incontro fra il conoscente e il conosciuto. Valmiki, l'autore del *Ramayana*, con la ferma attenzione dello yoga, *visualizza* l'intero poema prima di dettarlo (Di Napoli, 2011).

El lenguaje

Este apartado se basa en algunos conceptos de la lingüística de Chomsky. No pretende ser exhaustivo ni científicamente riguroso, sino apropiarse de algunos conceptos en sentido práctico para el desarrollo de las *shape grammars*; su eficacia práctica hace que no dependa del rigor académico. Por otro lado, se pretende abrir algunas líneas de investigación en la relación entre *computer science* y lingüística.

Existe el hecho que los seres humanos tienen propiedades lingüísticas innatas. El innatismo es la facultad lingüística como patrimonio genético de la especie humana. (LAD - *Language Acquisition Device*) encuentra pruebas en varias propiedades de la adquisición del lenguaje en los niños.

La sintaxis es el conjunto de principios que regulan las posibilidades de combinación de las palabras.

La gramaticalidad es la buena combinación de palabras de acuerdo a la intuición del hablante nativo. Existen al respecto intuiciones del hablante nativo sobre la interpretación de las proposiciones. También el hablante nativo puede dar juicios inconscientes sobre la gramaticalidad de las proposiciones. Es un sistema interiorizado de reglas y principios.

[2] Véase el cap. 1 y el artículo *La virtualidad aumentada: procesos emergentes, arte y medios digitales* anexo a este documento.

Es necesario diferenciar la gramaticalidad de la aceptabilidad. Esto significa que hay proposiciones sintácticamente correctas pero sin significado. Es decir, que expresan situaciones que no tienen relación con ninguna situación comunicativa predeterminada. La aceptabilidad se refiere al utilización efectiva del lenguaje en situaciones de comunicación concreta. Entonces pueden darse situaciones sin sintaxis correcta pero comprensibles.

El lenguaje humano tiene ciertas características que lo hacen diferente de otras formas de comunicación[3].

- Recursividad. Proposiciones de longitud potencialmente infinita utilizando proposiciones embebidas una en otra.
- Dependencia de una estructura. Existe una estructura abstracta que valida todas las relaciones sintácticas.

Una teoría de la gramática debe ofrecer los siguientes criterios (basados en el programa minimalista de Chomsky en los años 90):

- Universalidad (principios que subyacen a todos los lenguajes)
- Unicidad (no confundirse con otros tipos de comunicación)
- Comprensibles/Enseñables (se deben poder aprender)
- Simples

Naturalmente los procesos lingüísticos computables, generativos y los sistemas de sustitución simbólica recursiva como los Sistemas-L difieren del lenguaje natural. Algunos aspectos pueden ser implementados en el sistema simbólico con poco aprendizaje, otros no.

La intuición no existe en un sistema de cómputo. Bajo este punto de vista, para facilitar el programa minimalista, es muy importante el diseño de la interfaz de la aplicación. Esto es parte importante de la presente investigación, como se documenta en el cap. 2.

Shape Grammars y Sistemas-L

Para el desarrollo de gramáticas generativas computables no es necesario decidir entre mito y ciencia o entre estructuralismo y *Gestalt*. Aunque ninguna teoría puede

[3] Al respecto es muy interesante comparar el lenguaje natural con el lenguaje computable.

explicar exhaustivamente el fenómeno de la forma, todas dan indicaciones que se pueden utilizar en los procesos de diseño, y que enlazan el eje naturaleza-precolombino-diseño-reglas. En este sentido la última etapa del informe es sobre la aplicación de las gramáticas generativas al proceso de Moray.

Ahora según Thom (1980), el fragmentase de las olas, las rajaduras de un viejo muro, la espuma y las burbujas de una vaso de cerveza representan formas de bifurcación (fractales) y generan angustia y malestar. Hay ciertas formas de la complejidad que supuestamente ponen en jaque el ritmo, el arquetipo, la regla.

Son las formas fractales, autosimilares; hoy queda claro que también esta clase de formas tiene su orden, un orden recursivo, dinámico, caótico. Un orden que da angustia solo porque sus procesos parecen ininteligibles, debido a su complejidad dinámica. Pero la comprensión de la complejidad se puede hacer con las gramáticas generativas. La comprensión del proceso, no de la forma, es algo que la computación bajo ciertas condiciones puede lograr. En seguida se muestran los principios de esta relación.

<u>El ritmo, las reglas, el arquetipo y la repetición</u>

Comenzamos diciendo que en una forma generada algorítmicamente hacemos la distinción entre los principios algorítmicos y los parámetros. La complejidad nace de regla simples (la espiral, el meandro, los andenes, las teselaciones de Voronoi) que reiteradas varias veces, producen complejidad. Es una complejidad en términos numéricos, cuantitativos, el sublime matemático, como decía Kant. Se logran de acuerdo a dos procesos:

1. Un bucle se presta para generar una espiral, un proceso recursivo una ramificación, etc. Igualmente se dan características de velocidad y claridad.
2. Por otro lado, la ramificación, las bifurcaciones se obtienen mediante llamadas a funciones o a subrutinas, pueden ser recursivas, utilizando saltos condicionales (hacer una tabla) y también utilizando programación con hilos (*multithread*).

<u>La complejidad dinámica y los parámetros</u>

Estos procesos simples producen cantidad, pero son procesos que pueden ser parametrizados, entonces tenemos cantidad pero con características siempre distintas, porque los parámetros del proceso pueden ser cada vez diferentes. Inclusive estos parámetros pueden variar en tiempo real, dinámicamente durante el mismo proceso.

La recursividad

Los procesos se suman, tenemos el proceso algorítmico y algoritmos para variar los parámetros. El mismo ritmo puede sustentar ambos procesos. Esto lleva al concepto de recursividad, que es clave para la complejidad. El proceso recursivo se retroalimenta con sí mismo. Los fractales son claros ejemplos de esta complejidad recursiva.

Las *shape grammars* y los Sistemas-L implementan todas las características de los sistemas algorítmicos recursivos y paramétricos. Las reglas implementan el algoritmo. Las reglas pueden ser parametrizadas. Los procesos pueden tener varios grados de recursividad.

La regla deja su marca en la forma emergente (el ritmo, el arquetipo, la *Gestalt*). Pero el proceso se puede estructurar, es parametrizado. Las gramáticas generativas se pueden programar, y las posibilidades de la programación simbólica y de los parámetros crean las estructuras de aprendizaje, la programación es experimentación, desarrollo por pruebas y errores.

Naturalmente el proceso computacional es determinista, mientras el ritmo, la *Gestalt*, es impredecible, es creativo, también el estructuralismo lo es. Entonces el ritmo, la contemplación creativa de la forma (*aletheia*) se implementa no tanto en la gramática en sí, en el algoritmo y en los parámetros, sino en el lenguaje subyacente, el que permite el diseño de estas reglas, en el lenguaje que permite que estos procesos hablen en términos generativos.

Esta es la tarea del programador y del diseñador de interfaces, la tarea ultima que viene después de todo, y que al mismo tiempo, desde el punto de vista del usuario, viene antes de todo. A modo de resumen, presentamos una síntesis de los procesos formales generativos y computables en la metodología de la presente investigación.

El concepto de forma y la metodología de investigación

- Las formas naturales. Revelan el ritmo, los principios de la Gestalt, analogías y homologías
- Arte precolombino: relación con la cosmovisión, relación con la naturaleza a través de la intuición y de la *Gestalt*, el arquetipo, la relación con el espacio, el tiempo y la función (Stevens).

- Dibujos: escritura automática (surrealismo), principios de las formas naturales, arquetipos y reglas
- *Shape grammars*: en el proceso de sustitución se implementa el ritmo natural, en las reglas el arquetipo, la *Gestalt.* Homología y analogía entre formas naturales, formas del arte y algoritmos.

Bibliografía

Flake, Gary W. (1998). *The computational beauty of nature*. Cambridge: The MIT Press.

Thomson D'Arcy W. (). *On growth and form*. Cambridge: Cambridge University Press.

Di Napoli, Giuseppe. (2011). *I principi della forma*. *Natura, percezione e arte*. Torino: Einaudi.

Wagenesberg, Jorge. (2003). *Ideas sobre la complejidad del mundo*. Barcelona: Tusquets.

Stevens, Peter. (1986). *Pautas y patrones de la naturaleza*. Barcelona: Salvat.

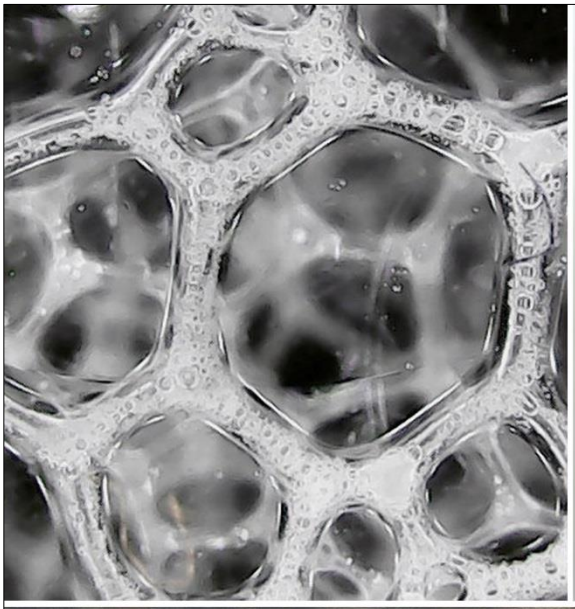
Klee, Paul. (1976). *Teoría della forma y della configurazione*. Milano: Feltrinelli.

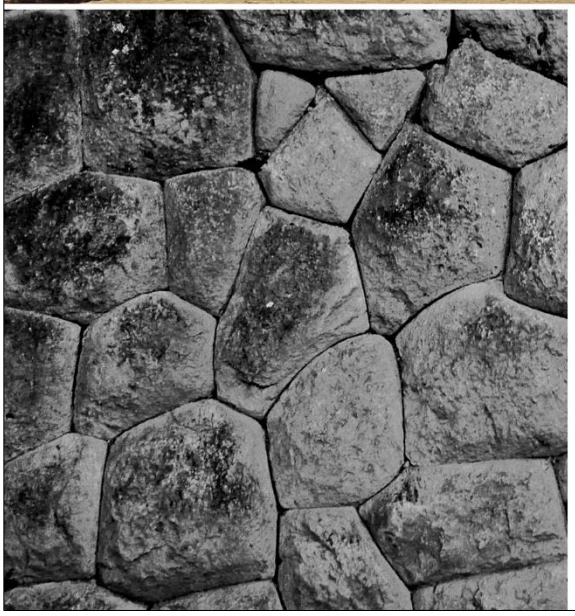
Thom, René. (1980). *Stabilitá strutturale e morfogenesi. Saggio di una teoria generale dei modelli*. Torino: Einaudi.

1.2 Fichas comparativas

En seguida se presentan las fichas comparativas que incluyen las formas naturales y el diseño precolombino.

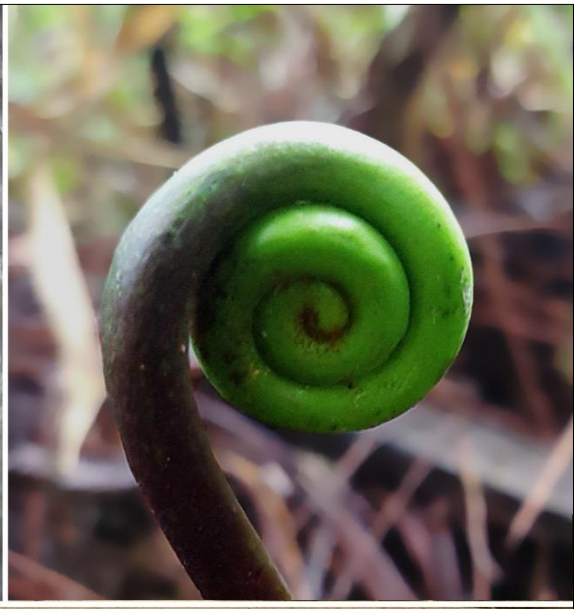
La función de estas fichas (14) es comparar las imágenes de la naturaleza y del arte precolombino. La comparación revela, a través de la lectura artística y de la intuición que interpretan las similitudes formales y estructurales, cuáles pueden ser los procesos morfogeneticos y sus principios, que se aplicarán a los dibujos algorítmicos y a las reglas.





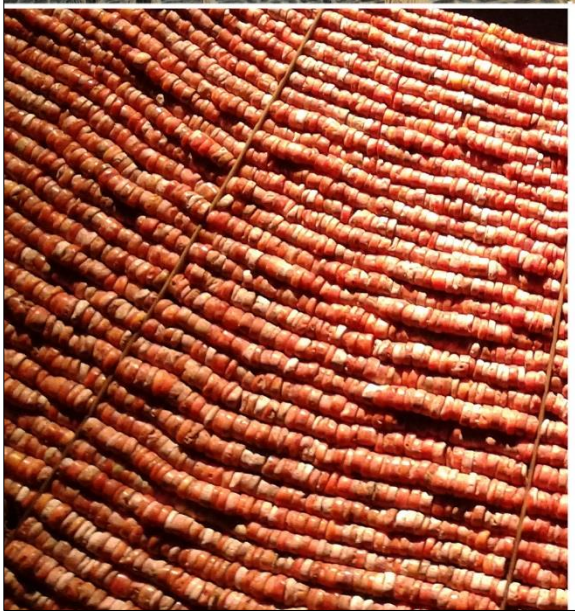
Patrones poligonales

Burbujas al microscopio y choclo.
Corrales en San Pedro de Cajas, Junín.
Muro inca en Chinchero, Cuzco.



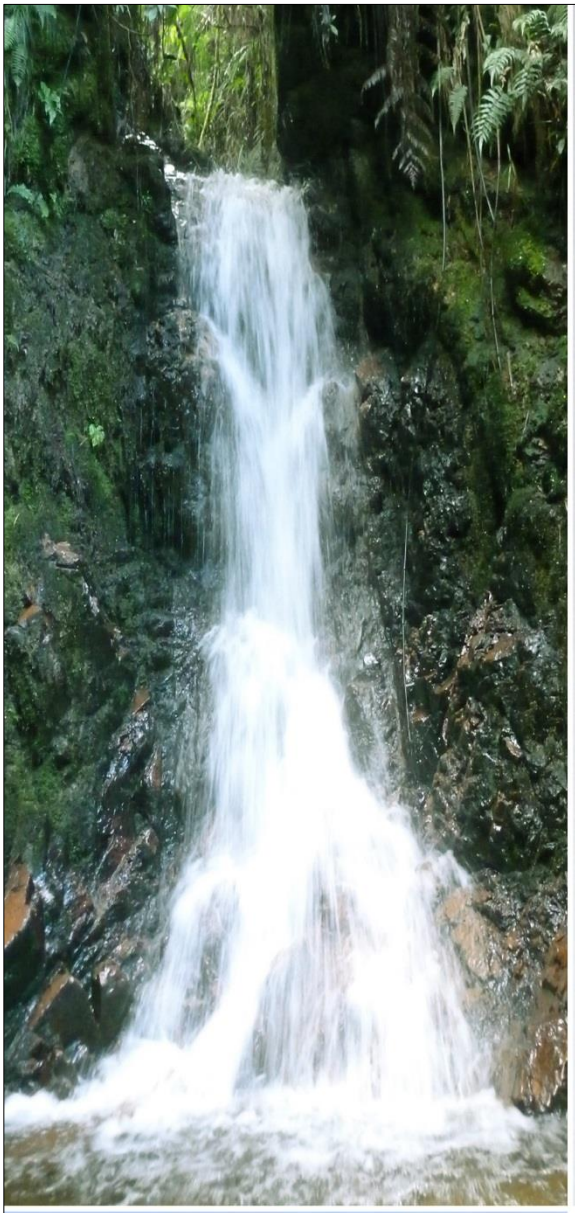
Estructuras espirales

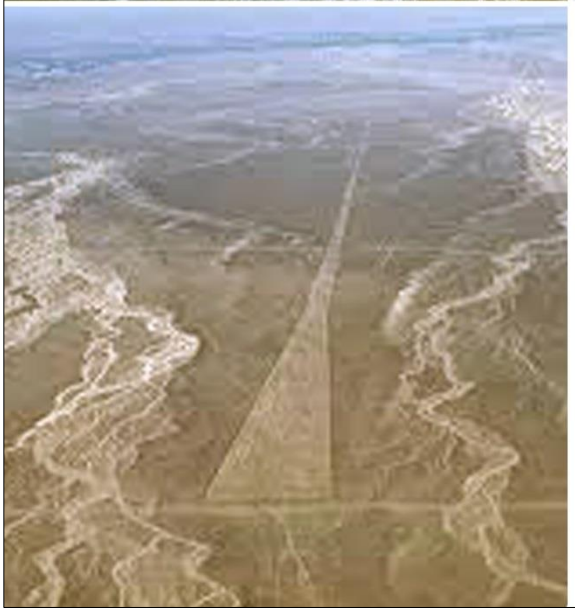
Huella digital.
Brote de fronda de helecho.
Acueducto de Cantayoc.
Tejido precolombino.



Ordenamientos estratificados

Estratificación rocosa en la costa de Lima.
Estratiticación de adobes, Huaca Pucllana.
Collar de cuentas de spondilus.



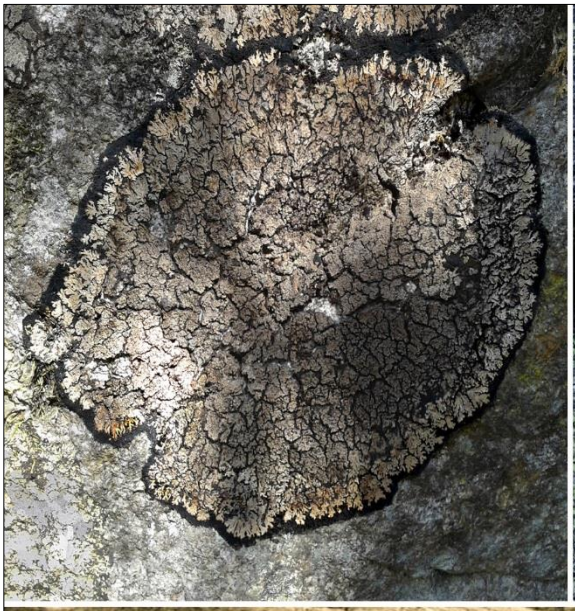


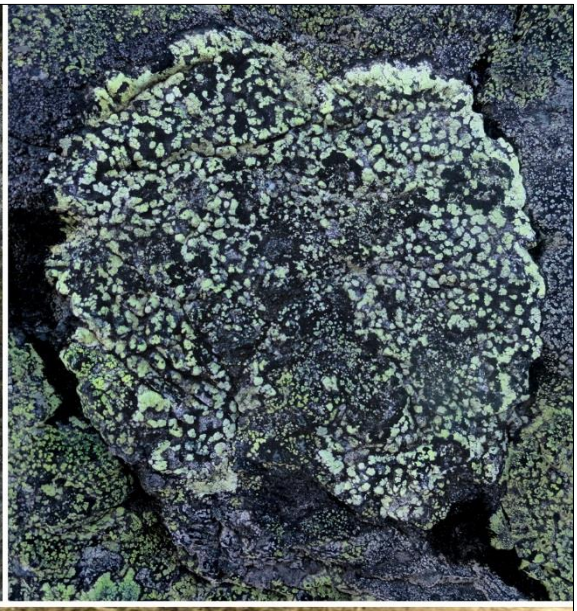
Estructuras cuneiformes

Cascada en Oxapampa.
Tupu precolombino.
Lineas en pampas de Nazca.

Estructuras de ramificación alternada

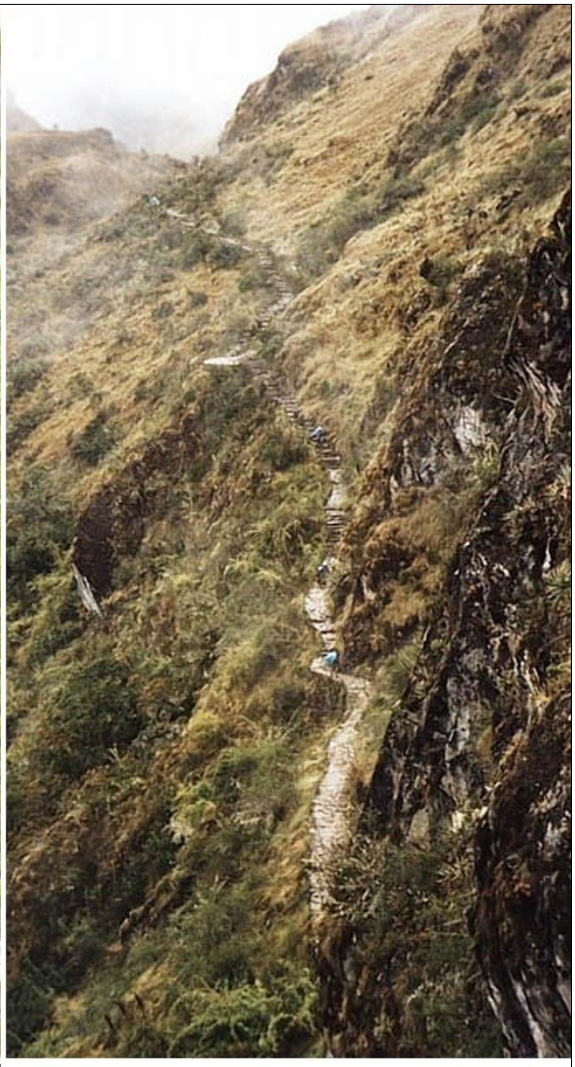
Hoja de helecho.
Sistema de regadío en Cochas Bajo.
Tejido precolombino Huari.

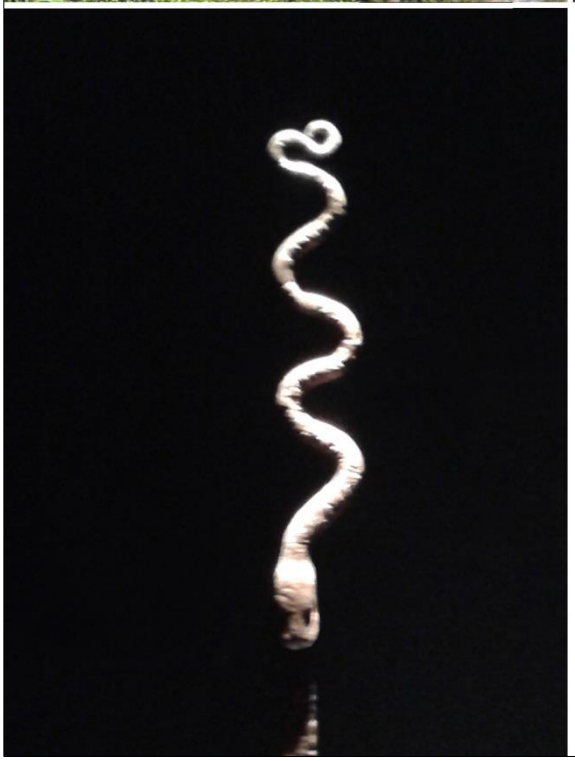




Crecimiento circulartificados

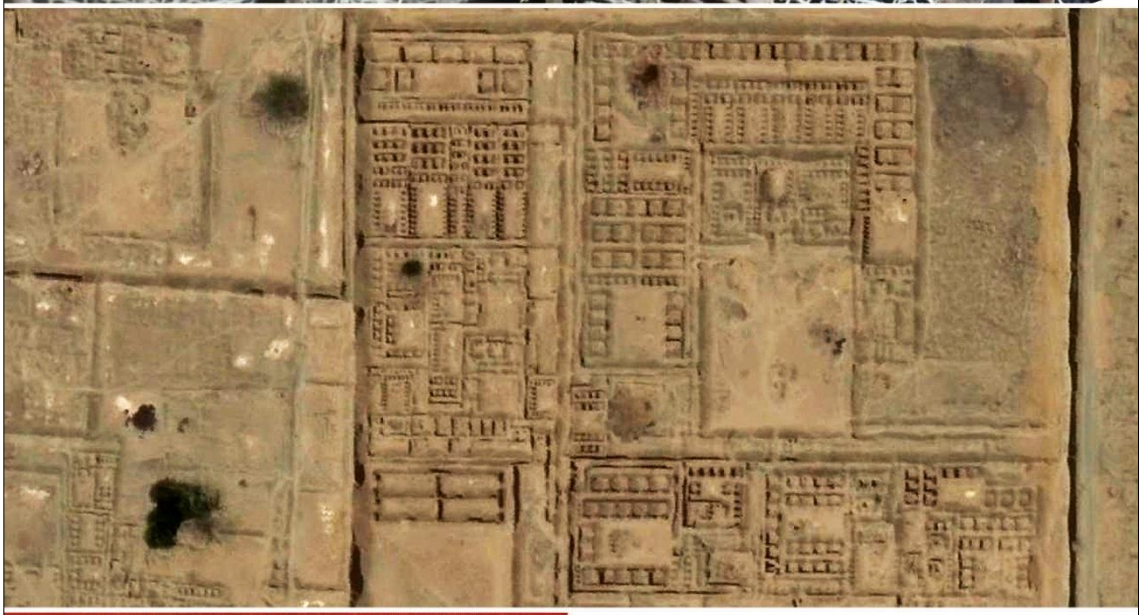
Líquenes en roca.sa en la costa de Lima.
Complejo de Chankillo en Casma.ucllana.
Chakana en tejido Huari.dilus.

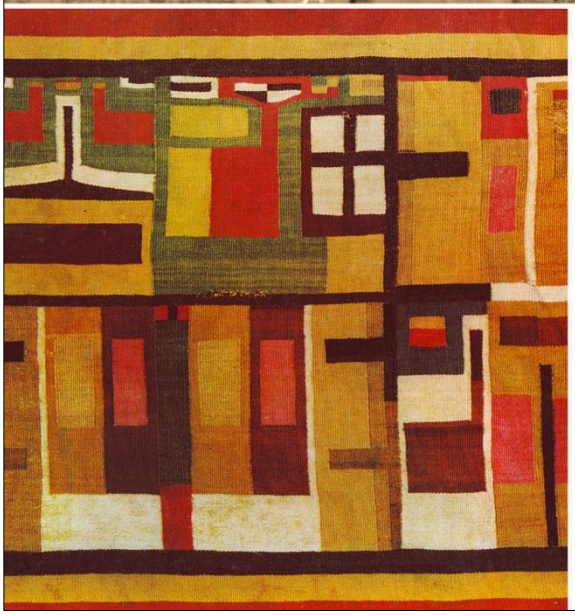




Estructura oscilante, meandros

Serpiente en Oxapampa.
Camino Inca:
Tupu precolombino con
terminación de serpiente.





Ordenamientos partitivos.

Estructura de hoja.
Chan Chan.
Textil Huari.

Patrón de crecimiento por anillos concéntricos

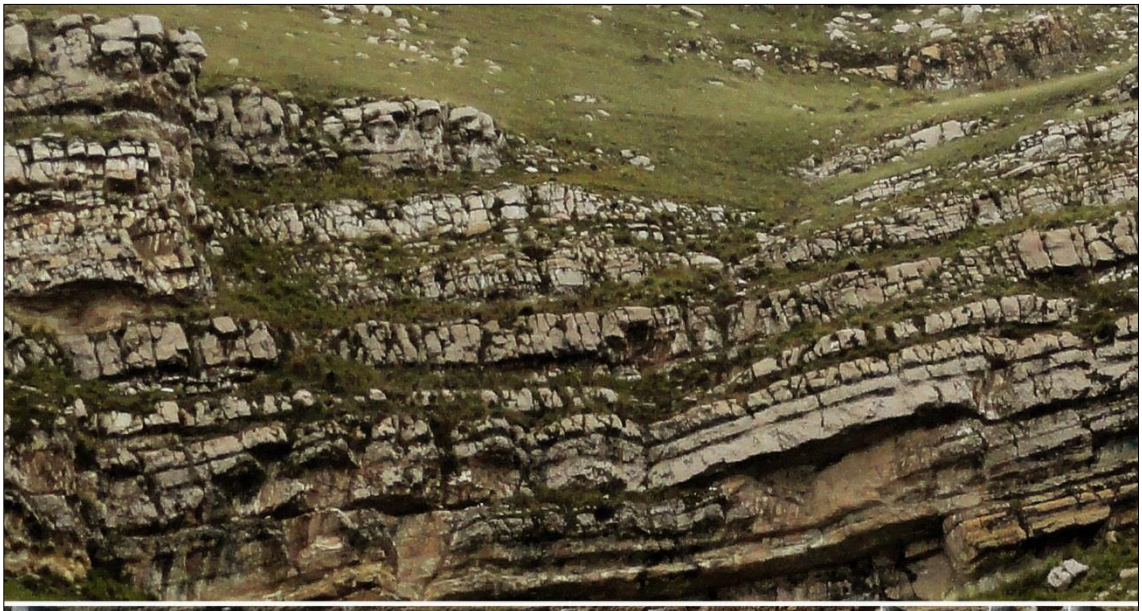
Ondas producidas por caida de gota en agua.
Anillos de crecimiento en un tronco.
Complejo de andenería Inca de Moray, Cuzco.
Decoración de círculos en botella Cupisnique.

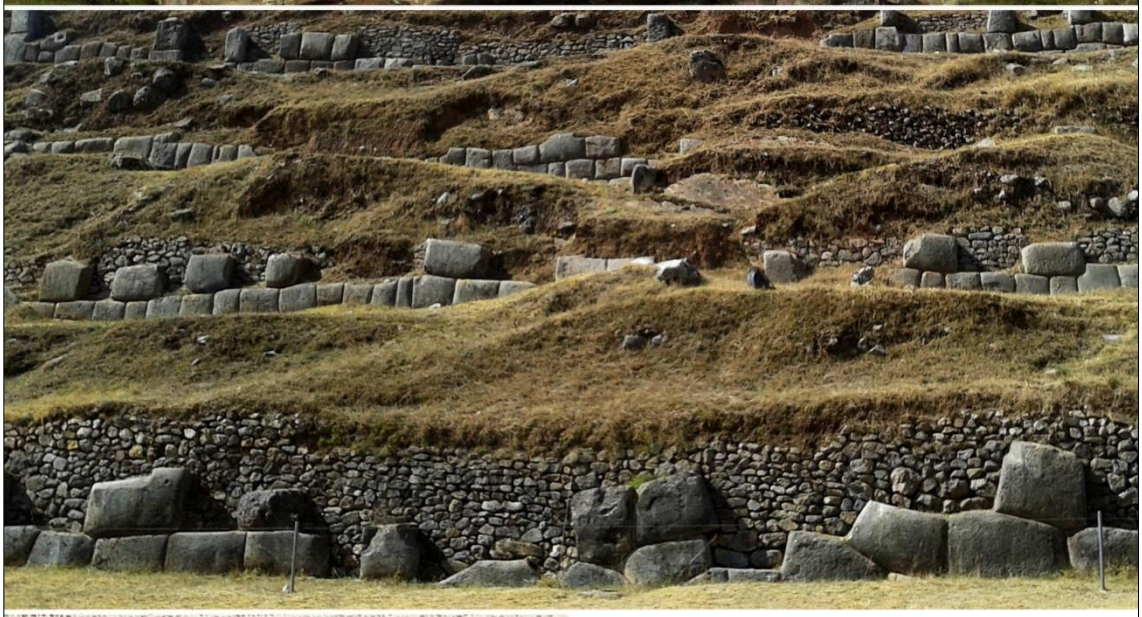
Sistemas de espirales

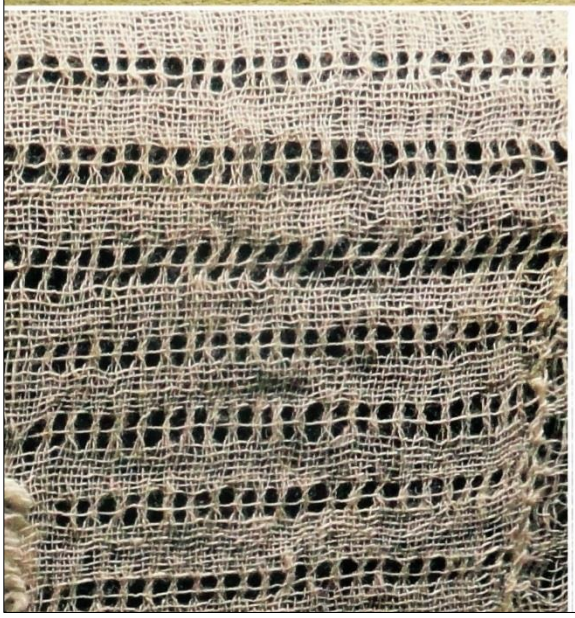
Arietitas.fósiles.
Acueducto de Cantayoc, Ica.
Tupu precolombino.

Estratificación y partición

Corteza de árbol.
Estratiticación de adobes, Huaca Pucllana.
Tejido precolombino de algodón.

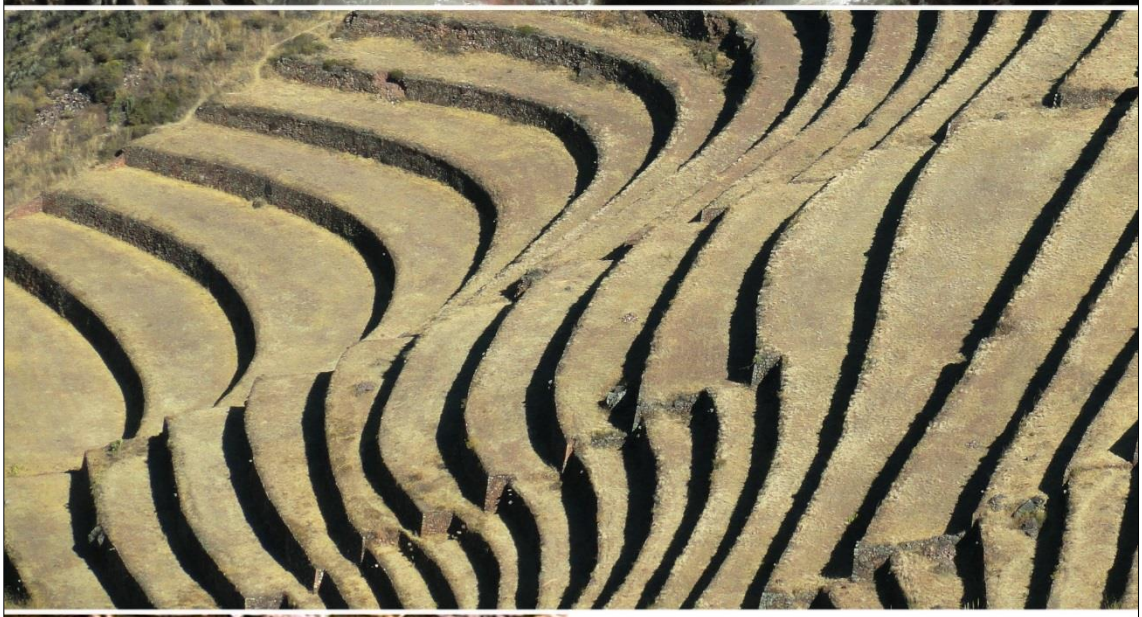






Estratificación alternada

Capas geológicas en Casaracra, Junín.
Alternancia entre muros y terreno, Sacsayhuamán, Cuzco.
Tejido precolombino de unku Chimú.



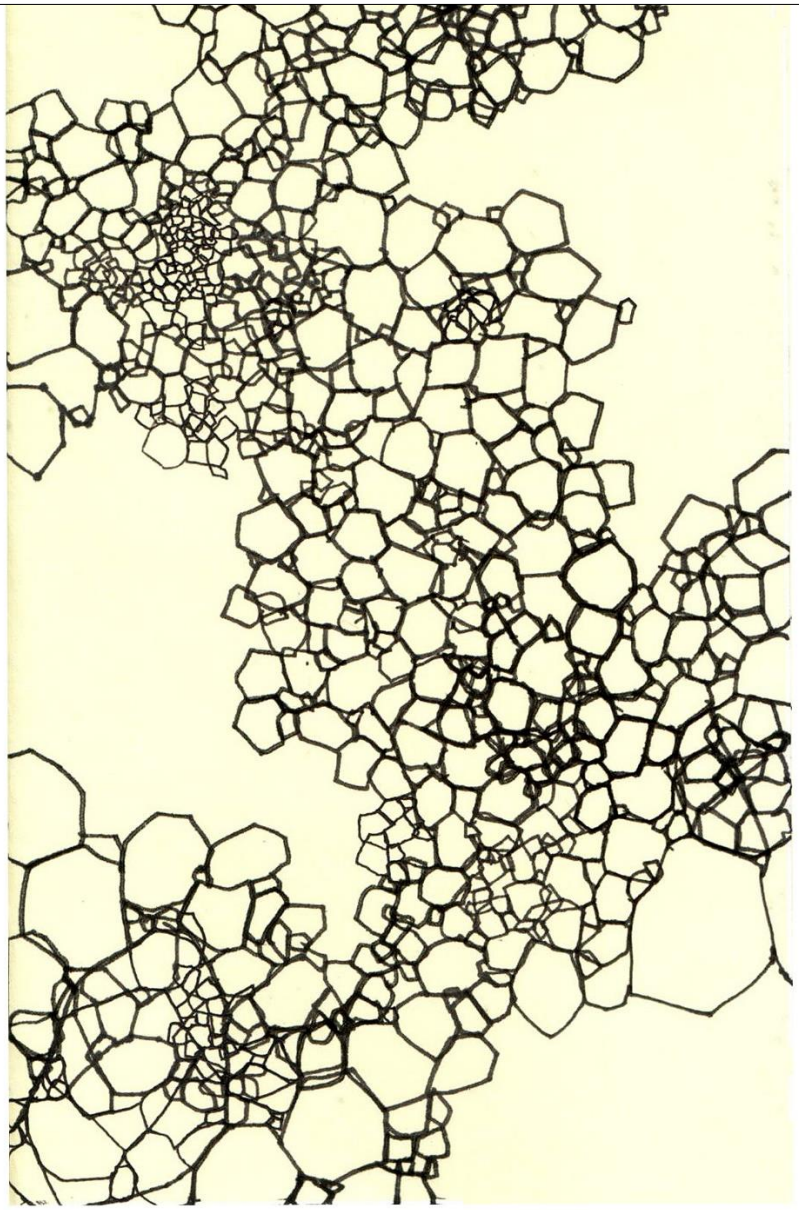
Sistema de ondulaciones paralelas

Ostra fósil..
Andenería de Pisac, Cuzco.
Cabellera ondulante de.cabeza decapitada, Templo de Sechín, Casma.

Estructuras romboidales con tallo

Hoja.
Tupu precolombino.
Tejido precolombino.

1.3 Dibujos algorítmicos

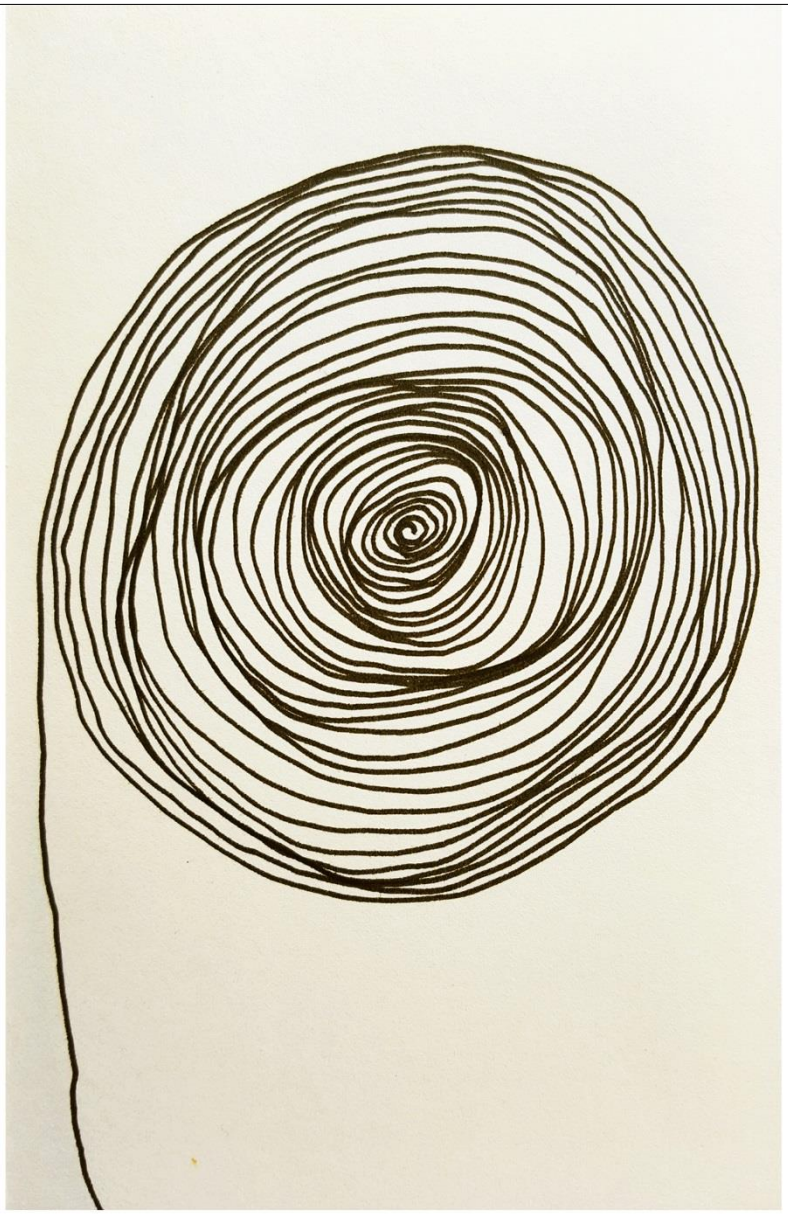


Patrones poligonales

Burbujas al microscopio y choclo.
Corrales en San Pedro de Cajas, Junín.
Muro inca en Chinchero, Cuzco.

Patrones poligonales

Dibujo algorítmico a mano alzada.
Proceso:
Se inicia dibujando una forma poligonal, uno de sus lados será compartido con la siguiente forma poligonal que se dibuje, y así sucesivamente. De este modo, se va llenando compactamente el espacio, cada nuevo polígono se ubica y adapta su forma y medidas a los vacíos que dejan los polígonos circundantes. Este proceso se repite y los polígonos van ocupando el espacio en todas direcciones sin que el trazo se despegue del papel. El patrón y dirección de crecimiento del sistema está determinado por la interacción entre el punto de cierre del polígono que se acaba de dibujar y la disposición y ubicación de sus vecinos.



Estructuras espirales

Dibujo algorítmico a mano alzada.

Proceso:

Parte una línea del borde del papel y a un cierto punto se empieza a plegar estableciendo un recorrido centrípeto que la lleva a enrollarse en sí misma. Las líneas intentan estar lo menos distantes entre ellas, pero el dibujo a mano alzada crea irregularidades que se mantienen o acentúan con la repetición de las líneas, configurando una espiral no perfectamente circular.

Ordenamientos estratificados

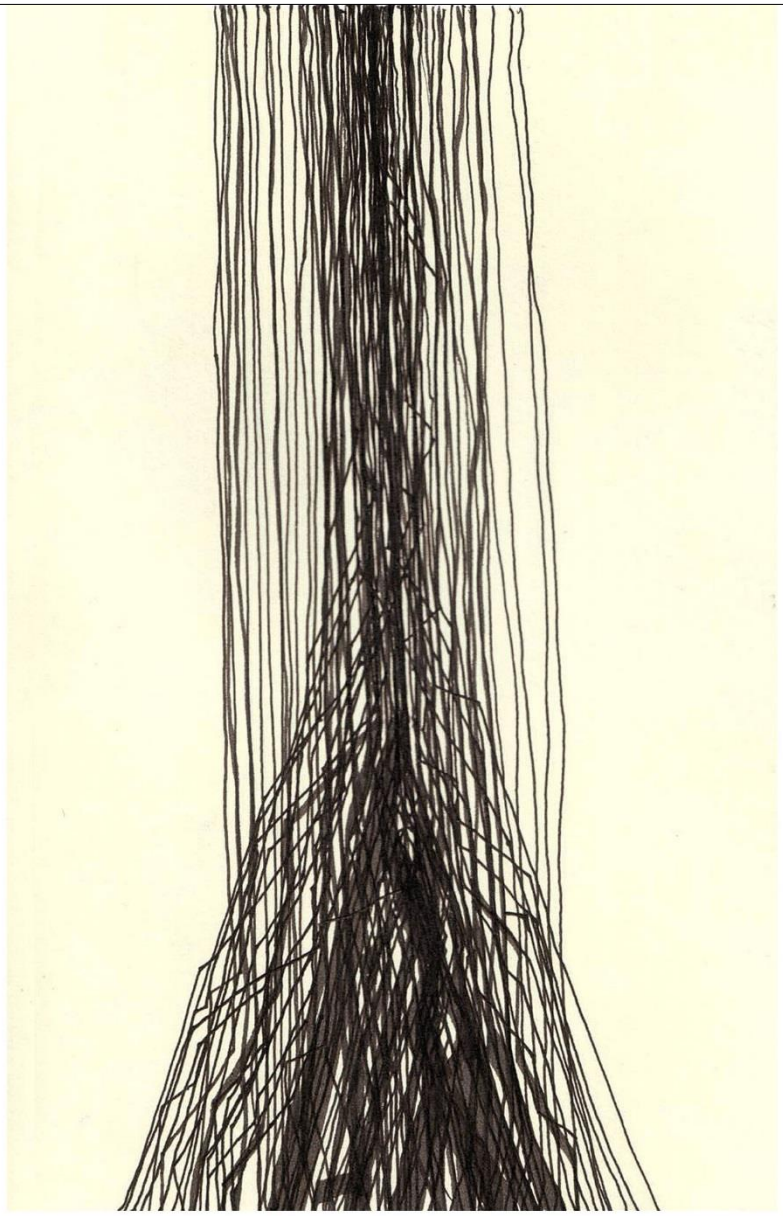
Estratificación roosa en la costa de Lima.
Estratificación de adobes, Huaca Pucllana.
Collar de cuentas de spondilus.

Estratificaciones

Dibujo algorítmico a mano alzada.

Proceso:

Líneas horizontales irregulares se trazan de derecha a izquierda y desde la parte baja del papel.
Cuando llegan a la parte izquierda, regresan trazando otra línea horizontal irregular.
Se forma así la estratificación horizontal, y desde las irregularidades de la misma, parten líneas verticales y diagonales que forman la estructura estratificada vertical.



Estructuras cuneiformes

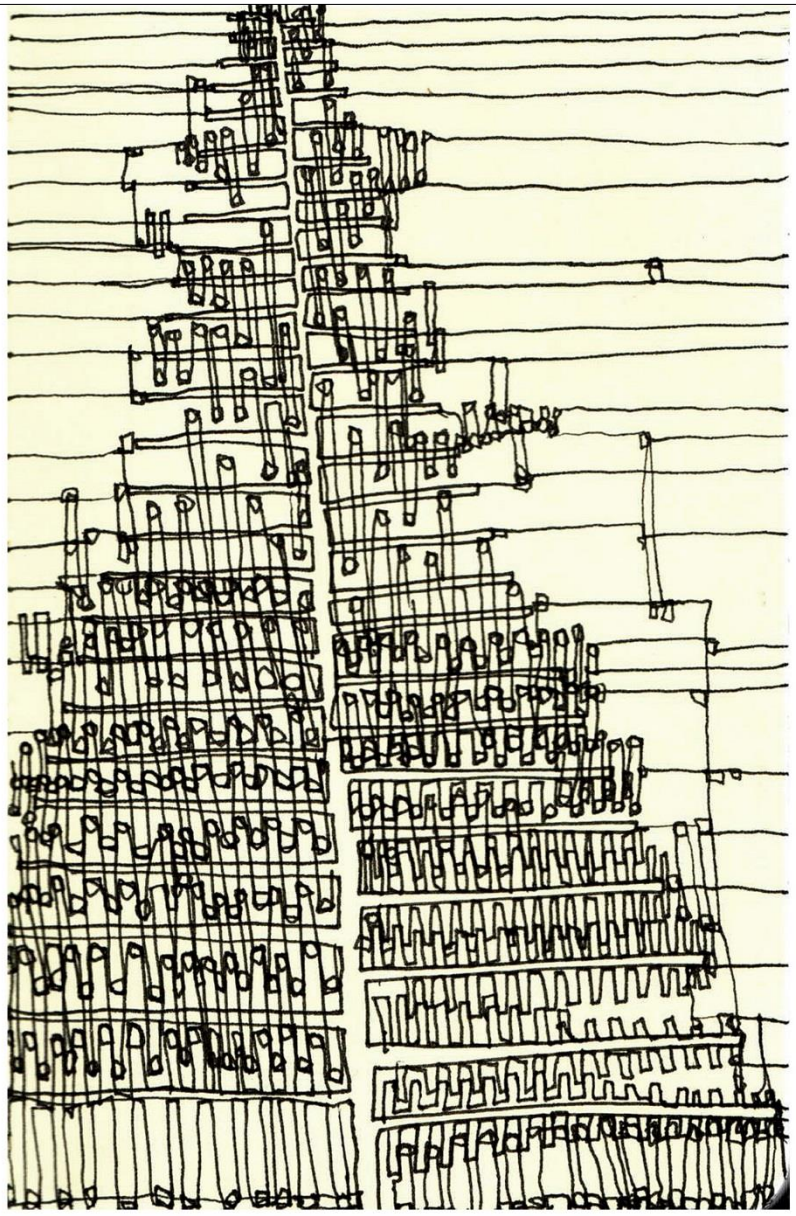
Cascada en Oxapampa.
Tupu precolombino.
Líneas en pampas de Nazca.

Estructuras cuneiformes

Dibujo algorítmico a mano alzada.

Proceso:

Se establece un flujo descendente a través de líneas verticales. Cuando el flujo ha tomado un determinado ancho, se incide en la parte central del mismo agregando más líneas descendentes que luego de haber recorrido el primer tramo verticalmente cambian de dirección, abriéndose con un ángulo de aproximadamente 45°. La acumulación de estas líneas diagonales forma la estructura cuneiforme.



Estructuras de ramificación alternada

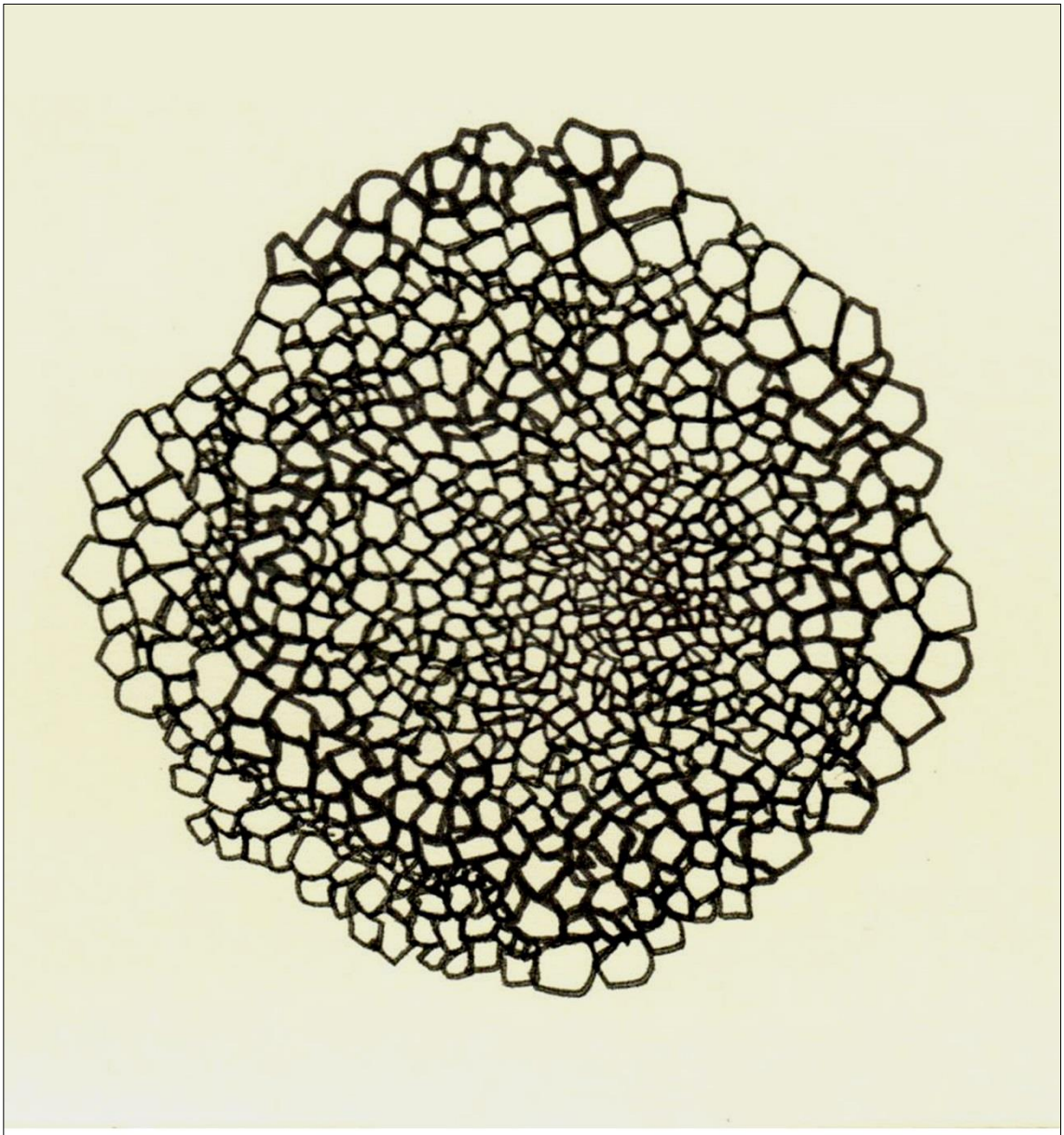
Hoja de helecho.
Sistema de regadío en Cochas Bajo.
Tejido precolombino Huari.

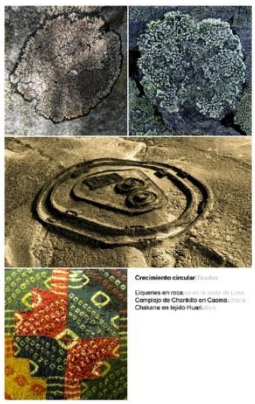
Estructuras de ramificación alternada

Dibujo algorítmico a mano alzada.

Proceso:

Se establece la estructura central vertical y sus ramificaciones laterales mediante líneas perpendiculares a la misma que se van alternando a izquierda y derecha. Luego se van creando relaciones entre las ramificaciones mediante líneas verticales que se van entrelazando a través de tramos cortos y giros en “U”.



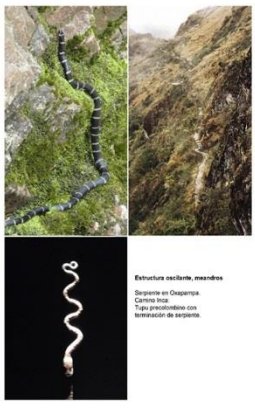


Crecimiento circular centrífugo

Dibujo algorítmico a mano alzada.

Proceso:

Se inicia en el punto central del papel dibujando una forma poligonal de 4, 5 o 6 lados. La arista donde se cierra el primer polígono es el punto de partida del siguiente, que queda ubicado inmediatamente adyacente al primero y sin dejar espacio.
Este proceso se repite una y otra vez, siendo la condicionante ocupar el espacio del centro hacia la periferia sin que el trazo se despegue del papel.



Estructura oscilante, meandros

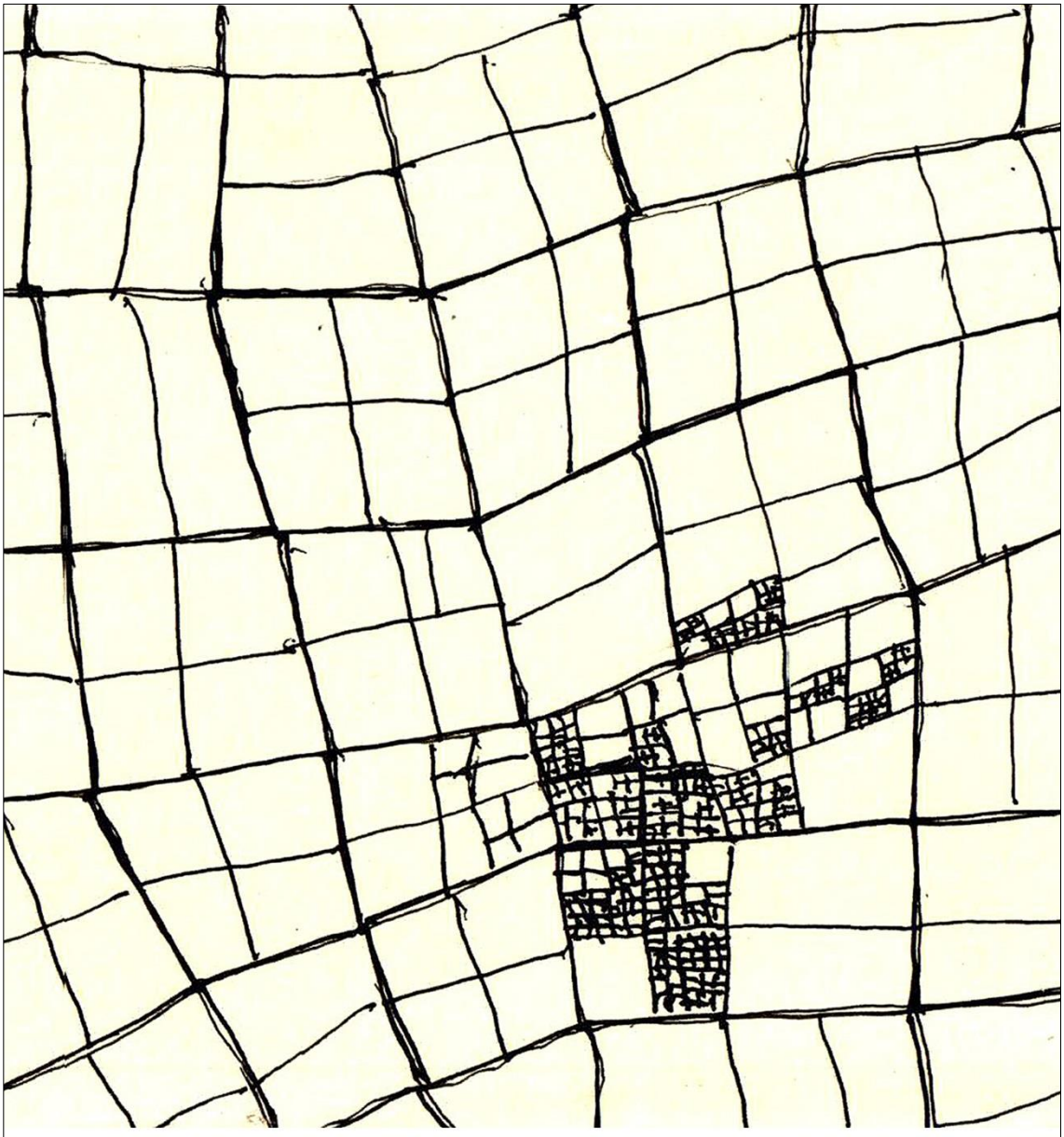
Serpiente en Oxapampa
Camino Inca
Tupu precolombino con
terminación de serpiente.

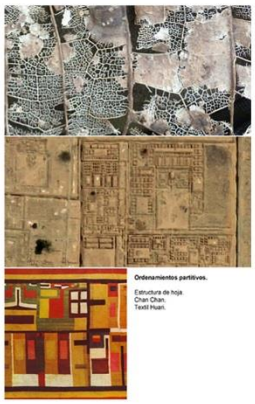
Estructura oscilante, meandros

Dibujo algorítmico a mano alzada.

Proceso:

Desde la parte baja del papel, líneas oscilantes se suceden replicando las curvas de su línea vecina. Éstas se van desplazando sobre zonas rugosas formadas por grupos de líneas que contienen a estas oscilaciones. En la parte alta se forman delgados meandros que parten desde la irregularidad (a modo de escalón) de una línea recta. Líneas paralelas a ésta replican la irregularidad que multiplicándose, se va desplazando y constituyendo formas sinuosas.



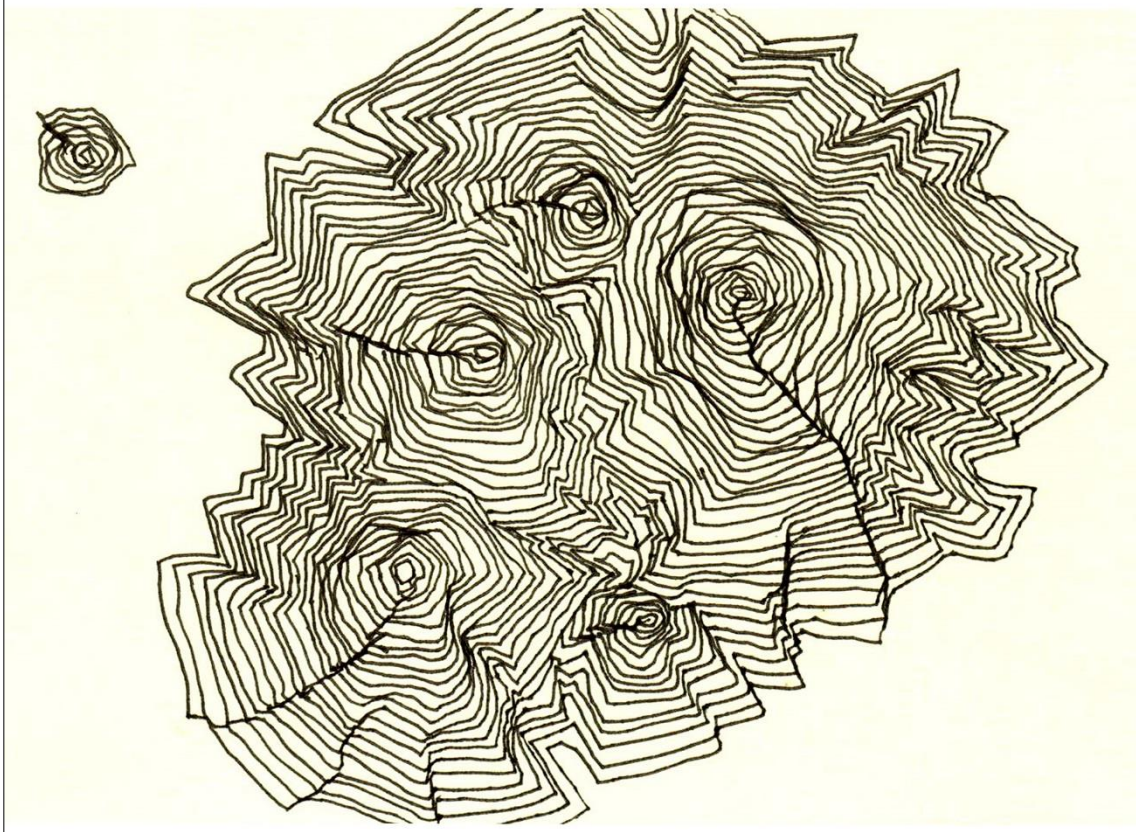


Ordenamientos partitivos.
Estructura de hoja.
Chan Chan.
Textil Huari.

Ordenamientos partitivos

Dibujo algorítmico a mano alzada.
Proceso:

Se dibujan grandes polígonos irregulares de 4 lados sin dejar espacio entre ellos. Una vez establecidas estas “parcelas”, se las va subdividiendo en partes cada vez más pequeñas, en mitades, luego en cuartos, en octavos y así sucesivamente hasta llegar a la mínima subdivisión posible y visible.



Patrón de crecimiento por anillos concéntricos

Dibujo algorítmico a mano alzada.

Proceso:

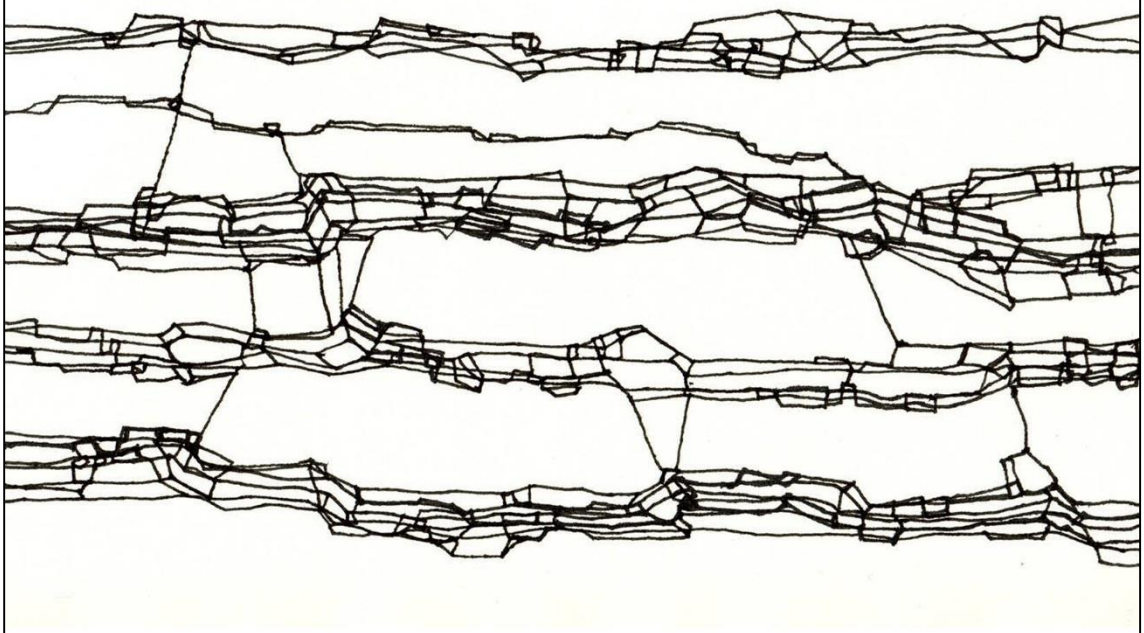
Se parte de varios pequeños círculos dispuestos en distintos puntos del papel. Cada círculo va creciendo agregando anillos exteriores sobre una línea guía que se prolonga paulatinamente para dar el punto de partida del siguiente anillo. Los círculos iniciales se transforman así en sistemas de anillos, cada uno con un ritmo de crecimiento distinto. En el proceso de crecimiento, los sistemas se tocan, y sus líneas entran en interacción compitiendo por el espacio del papel.

Estratificación y partición

Dibujo algorítmico a mano alzada.

Proceso:

La línea parte verticalmente desde el borde inferior del papel, a un cierto punto de su recorrido traza un pequeño cuadrado que le permite cambiar de dirección 180° y volver con una línea paralela a la anterior. Cuando llega a la parte baja traza otro cuadrado y vuelve hacia arriba. El proceso se repite hasta abarcar horizontalmente todo el papel y formar el primer estrato.
El siguiente se forma por una nueva línea que parte desde los cuadrados de cierre del anterior. Así se van "tejiendo" los estratos, acumulándose uno sobre otro, y la distinta densidad de línea de cada uno determina su mayor o menor partición vertical.



Estratificación alternada

Capas geológicas en Casaracra, Junín.
Alternancia entre muros y terreno.
Sacsayhuamán, Cusco.
Tejido precolombino de uriku Chimú.

Estratificación alternada

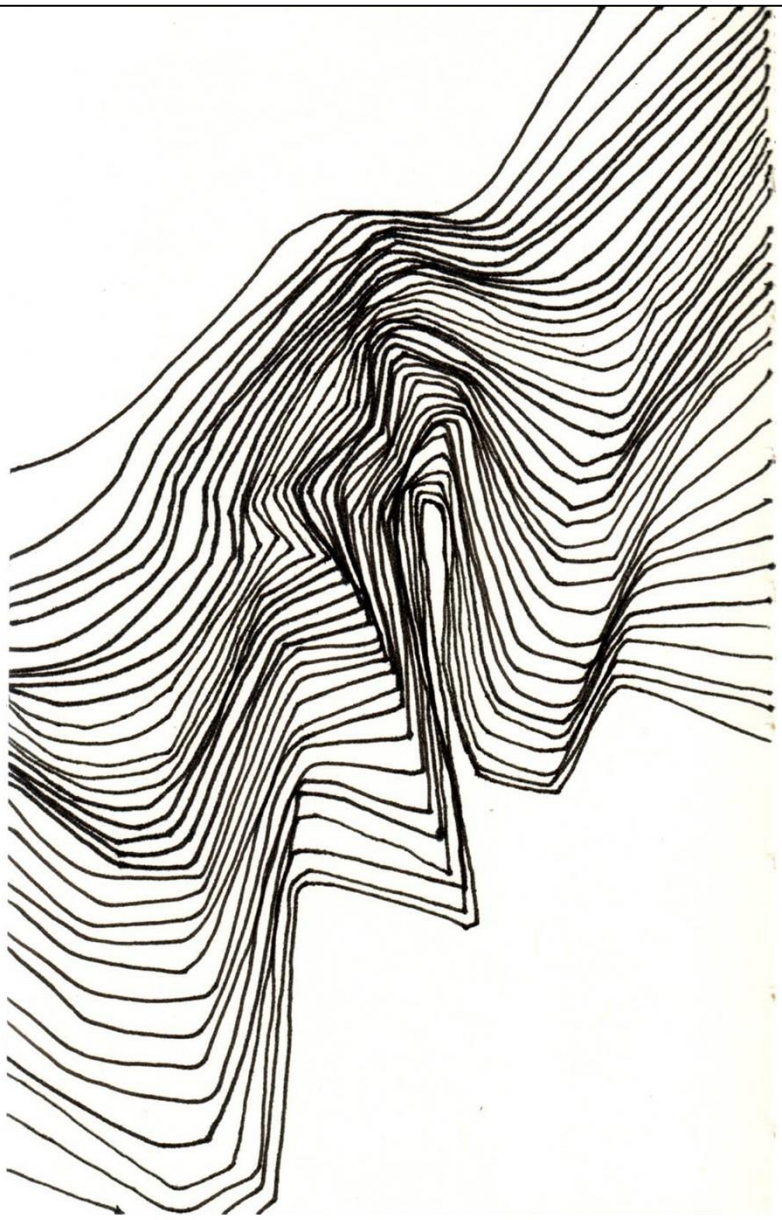
Dibujo algorítmico a mano alzada.

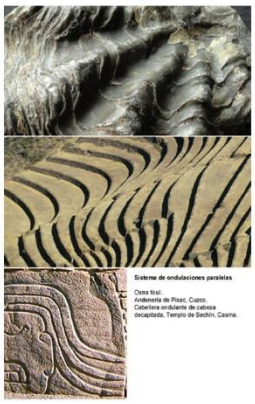
Proceso:

Se traza en la parte baja del papel un grupo de líneas horizontales, estableciendo un espacio en blanco entre el primer y el siguiente grupo de líneas. Esta alternancia entre grupos y espacio se repite hacia arriba varias veces para generar las estratificaciones.

La mano alzada impide una precisión en el trazo, lo que configura irregularidades y quiebres en las líneas.

Las aristas de estos quiebres son el punto de partida para trazar polígonos irregulares que fragmentan y complejizan los estratos, haciéndolos también crecer en grosor si rebasan su ancho. Las líneas diagonales de los polígonos pueden sobresalir del estrato lo suficiente para llegar al siguiente y crear conexiones entre ellos.



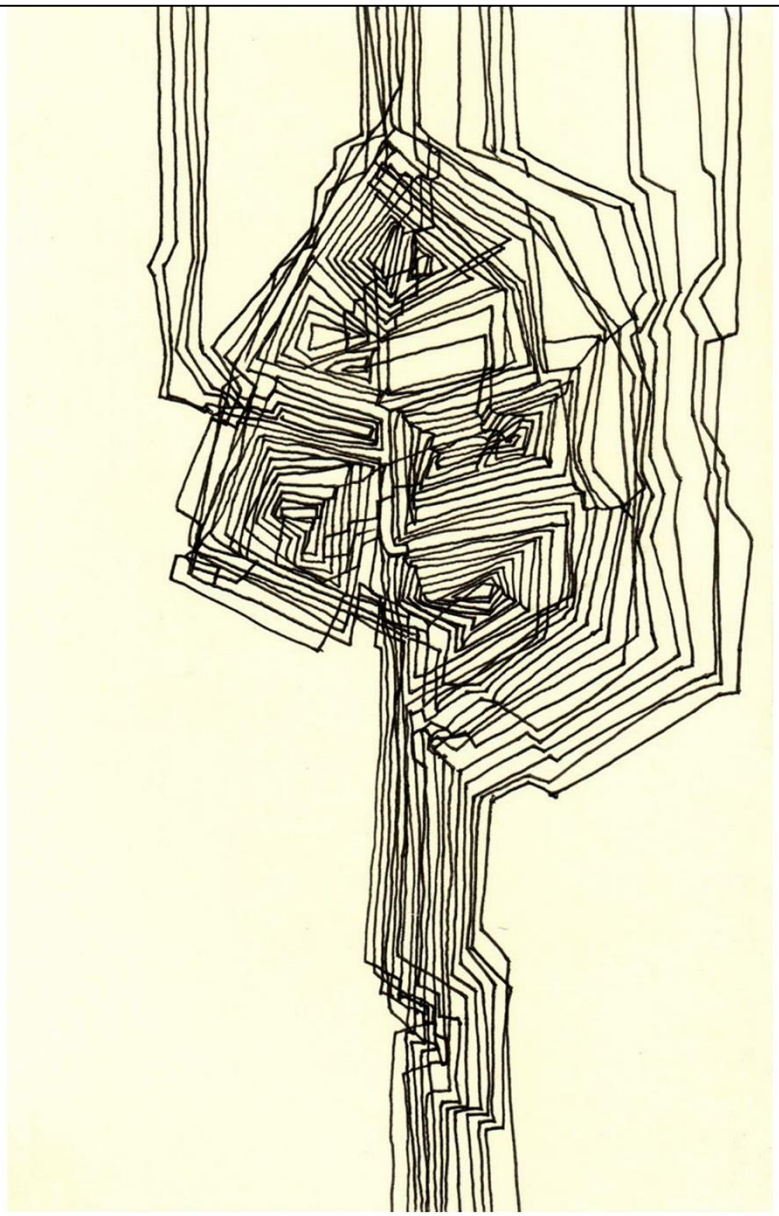

Sistema de ondulaciones paralelas
Ostra fósil.
Andenería de Pisac, Cuzco.
Cabellera ondulante de cabeza decapitada, Templo de Sechín, Casma.

Sistema de ondulaciones paralelas

Dibujo algorítmico a mano alzada.

Proceso:

Se inicia en la parte alta del papel trazando una línea con una ligera ondulación. Se continúa hacia abajo, dibujando líneas que intentan adaptarse a su entorno inmediato y ser paralelas a su vecina inmediata. Esto quiere decir que las irregularidades y variaciones de cada línea, introducidas por la casualidad y la imprecisión que aporta el dibujo a mano alzada, va creando ondulaciones totalmente distintas a las de la línea inicial.
Es un dibujo emergente, que mediante la repetición y sucesión de líneas, va creando un sistema complejo de ondulaciones.

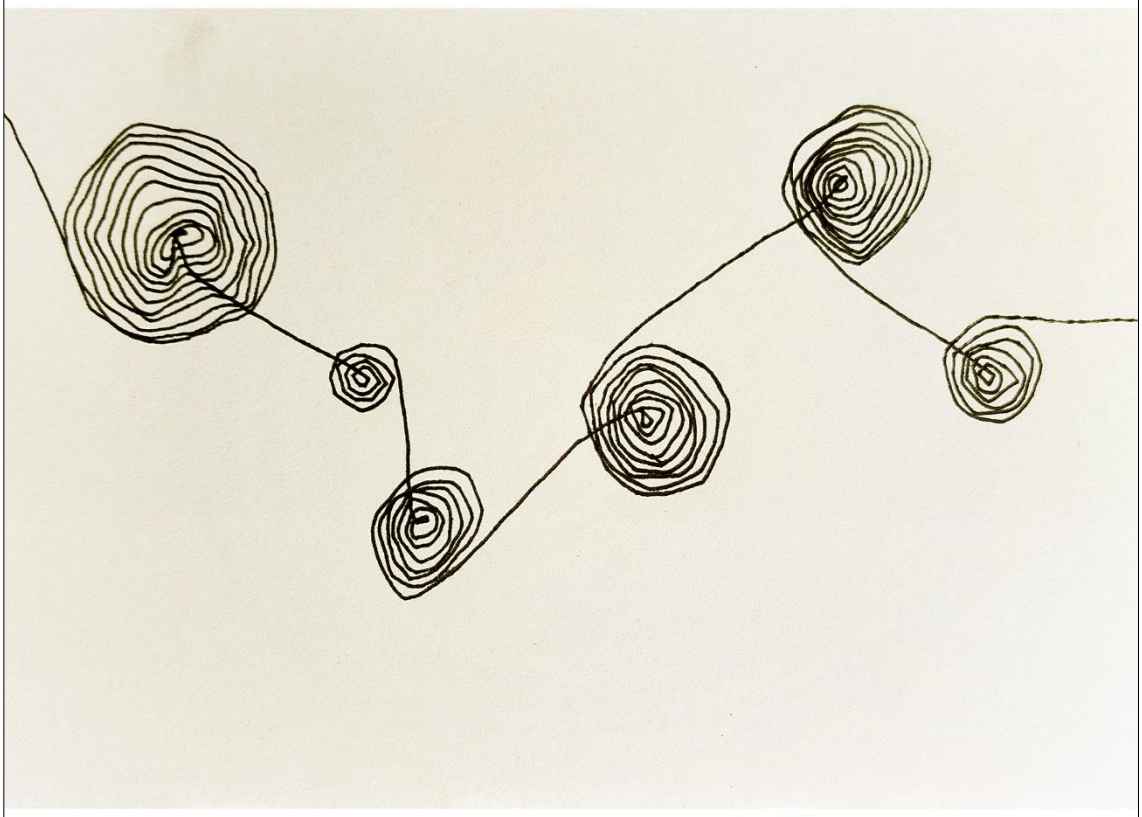


Estructuras romboidales con tallo

Dibujo algorítmico a mano alzada.

Proceso:

Cada línea hace su recorrido vertical ascendente sin despegarse del papel, desde su límite inferior hasta el superior. Se inicia en el centro del papel, la línea en su trayecto hace pequeños cambios de dirección que configuran irregularidades. Se continúa trazando líneas paralelas a la primera, tanto a su izquierda como a su derecha, y las irregularidades van creciendo exponencialmente a medida que las líneas se suceden. Así se configura el tallo. Las líneas que le suceden (aproximadamente a la mitad de su recorrido vertical) se desvían en diagonal hacia ambos lados, desplegándose lateralmente y volviendo al tallo para continuar su ascenso vertical varias veces antes de terminar su recorrido en la parte alta del papel. Repitiendo muchas veces la alternancia entre las lógicas de desarrollo vertical con las de despliegue lateral se va configurando la estructura romboidal a través de la acumulación de líneas paralelas.



Sistemas de espirales

Dibujo algorítmico a mano alzada.

Proceso:

Una línea empieza del borde lateral superior del papel. La línea comienza a curvarse enrollándose hacia el centro formando una espiral. Cuando no puede seguir su trayectoria hacia el centro, la línea se dispara en otra dirección y cuando está lo suficientemente lejos, empieza a generar otra espiral pero esta vez del centro hacia afuera. Este proceso se repite y las espirales se van sucediendo formando un sistema, donde cada espiral depende de la dirección de la línea y de la dirección en que ésta se enrolla.

Cap. 2. *Software*

Introducción

Para el desarrollo del proyecto se ha desarrollado un software para el diseño con las gramáticas generativas, basado en los Sistemas-L. Existen diferentes soluciones informáticas para el diseño con los Sistemas_L disponibles gratuitamente en internet, como Fractint para Windows. Sin embargo carecen de interfaz gráfica, de documentación y sus funciones no permiten el control interactivo del sistema.

El software que se ha desarrollado es una evolución de GDesign 2.0, precedentemente desarrollado por quien escribe. Para la investigación era necesario disponer de funciones para el diseño de patrones naturales y para el estudio de formas arquitectónicas, siendo el arte precolombino una síntesis que presenta al programador muchos retos en términos algorítmicos y de diseño de interfaz. El diseño precolombino presenta estructuras y procesos que ponen las gramáticas generativas al tope de sus posibilidades.

Por otro lado, los procesos de sustitución recursiva, como los IFS, las *shape grammars* y los Sistemas-L son implementados sin dejar mucho espacio a la interactividad con el usuario. Sin embargo estos procesos son conceptualmente simples y pueden ser comprendidos sin tener una cultura informática específica, lo que permite a los usuarios crear sus propios sistemas personalizados.

Para esto, tanto el diseño de la interfaz cuanto los algoritmos utilizados en el proceso de sustitución recursiva deben ser interactivos y trasparentes, es decir, el usuario debe siempre estar al tanto de lo que hace el sistema y de como lo hace. En seguida se mostraran las características del software, el diseño de la interfaz, las propiedades interactivas y como se ha logrado la trasparencia de los procesos.

La aplicación, denominada GDesign 4.0 se compone básicamente de 2 módulos principales, el diseño y la gramática, más unos módulos adicionales para la gestión de parámetros y opciones. La versión 4.0 incluye varios avances desarrollados para la investigación, aunque su interfaz no es completa como la versión 3.0.

Ventana de diseño

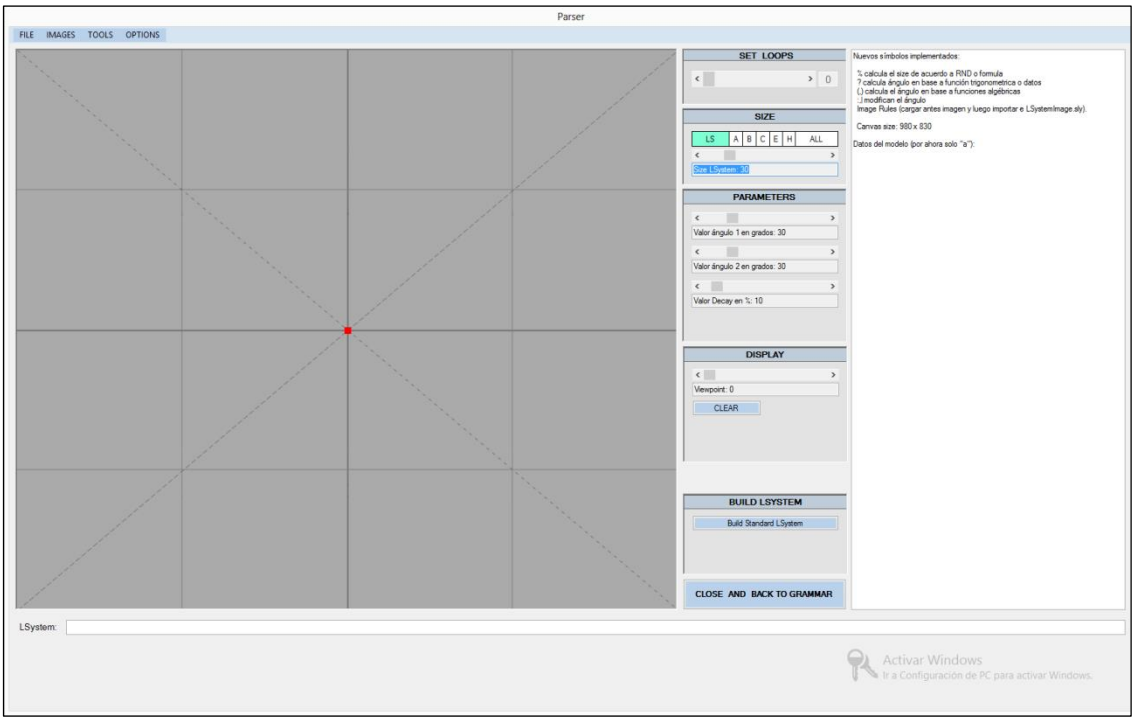


Fig. 1. Área de dibujo 3D

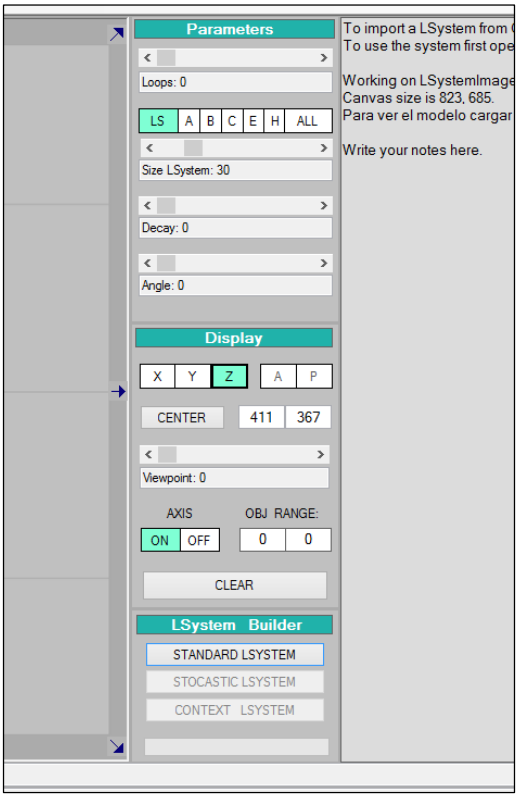


Fig. 2. GDesign 3.0

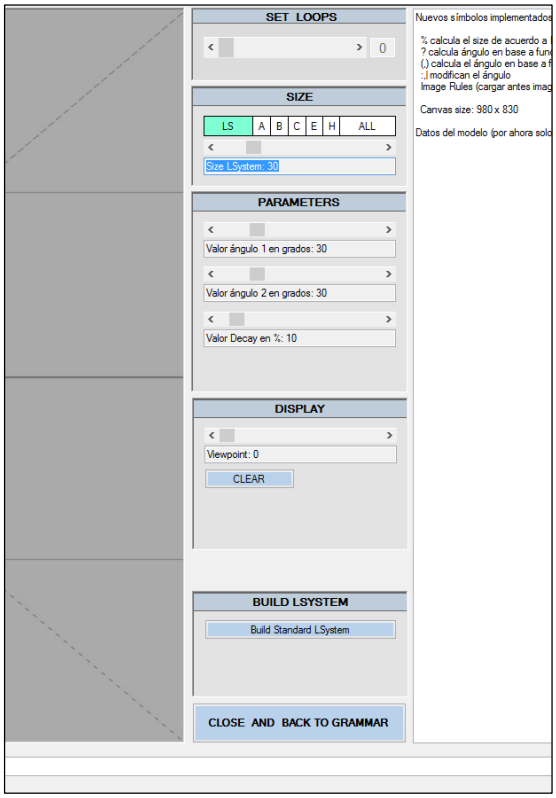


Fig. 3. Versión 4.0

En la ventana de diseño se genera y se visualiza el Sistema-L que ha sido programado en la ventana Gramática. El modelo es paramétrico, es decir se puede modificar interactivamente de acuerdo a una serie de variables.

Se pueden modificar los tamaños de los objetos, los ángulos de rotaciones, los porcentajes de trasformaciones de escala (Decay), los puntos de vista y el lente de la cámara. Es posible visualizar los ejes de los objetos y habilitar la vista de axonometría.

Los parámetros varían entre las versiones 3.0 y 4.0, sin embargo todas las opciones de la v. 3.0 (izquierda) pasarán a la 4.0 (derecha) pero no viceversa.

Ventana gramática

La ventana gramática es el verdadero corazón de la aplicación, es donde se programa el proceso de sustitución recursiva. Son disponibles 5 símbolos no terminales, varios símbolos especiales, funciones y subsistemas. Las reglas se expanden para dar acceso a las reglas estocásticas y de sistema.

Cada símbolo tiene la posibilidad de programar 8 pasos diferentes en orden secuencial de arriba (regla 1) hacia abajo (regla 8).

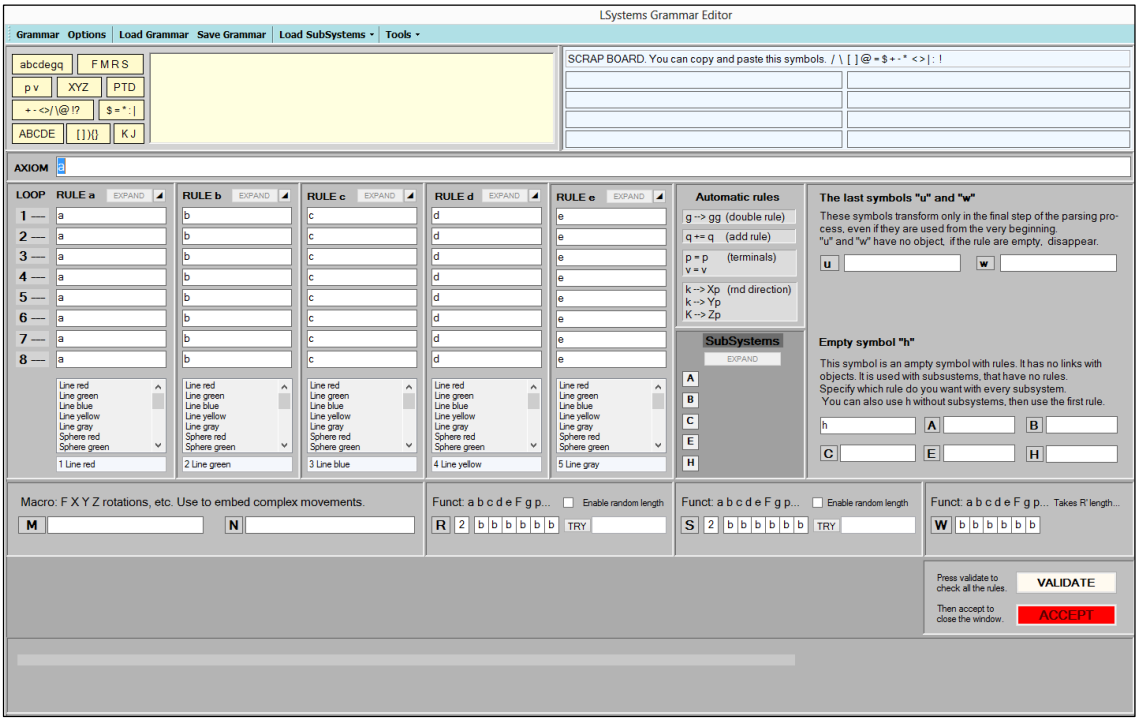


Fig. 4. GDesign 3.0. La ventana Gramática. Son disponibles todos los símbolos no terminales pero no hay todos los símbolos que se han agregado en la *release* 4.0.

El formulario provee varias utilidades para la edición de las reglas y la programación de los símbolos especiales en diferentes ventanas adicionales.

Un aspecto complicado de los procesos de sustitución recursiva es precisamente monitorear las reglas en las diferentes etapas de su desarrollo. Se ha tratado de diseñar la interfaz para que el proceso sea intuitivo e interactivo. Esto ha sido el fruto de prácticas continuas desarrollando los diseños precolombinos que presentan retos siempre diferentes a la programación.

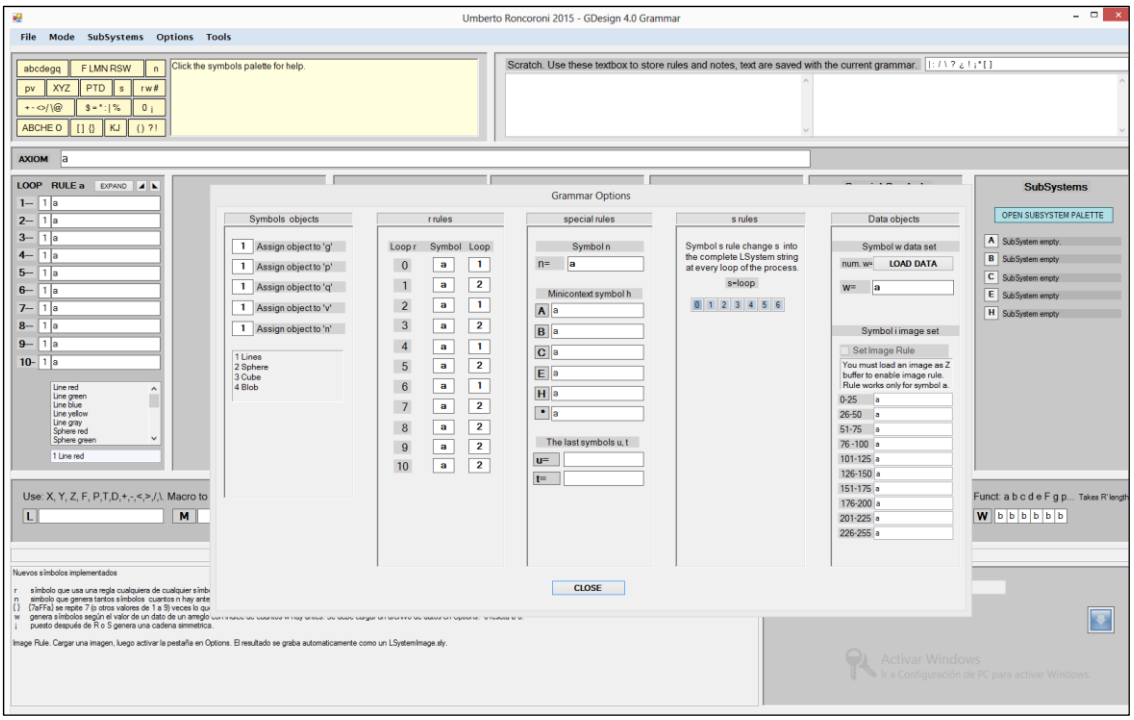


Fig. 5. GDesign 4.0. Ventana Opciones para los símbolos especiales.

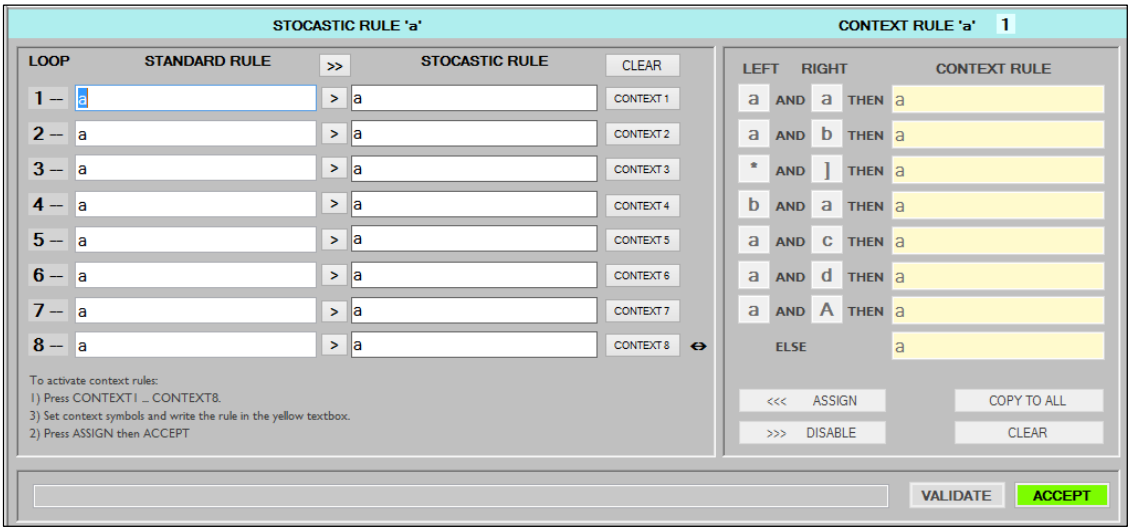


Fig. 6. GDesign 3.0. Ventana para reglas estocástica y de contexto.

Los subsistemas son Sistemas-L completos que se pueden utilizar dentro otro Sistema-L. Esto se hace también copiando un Sistema-L completo dentro una regla, pero con los subsistemas pueden modificarse de acuerdo al contexto (fig. 7) porque incluyen el símbolo "*" que se modifica diversamente de acuerdo al contexto; además tienen la posibilidad de un símbolo especial para la conexión entre otros símbolos.

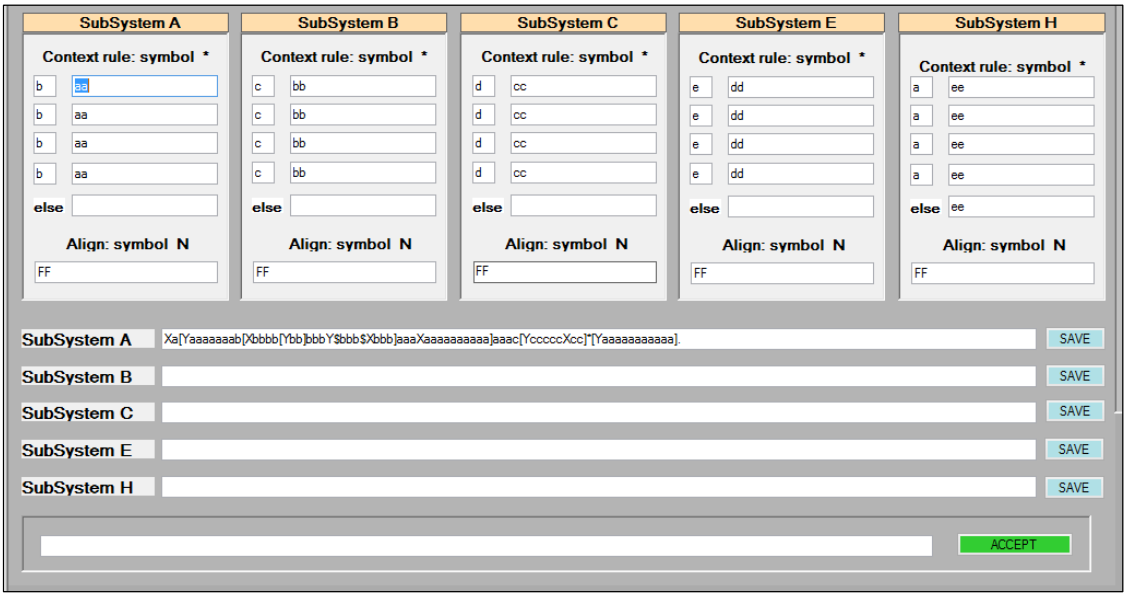


Fig. 7. Paleta de subsistemas en GDesign 3.0

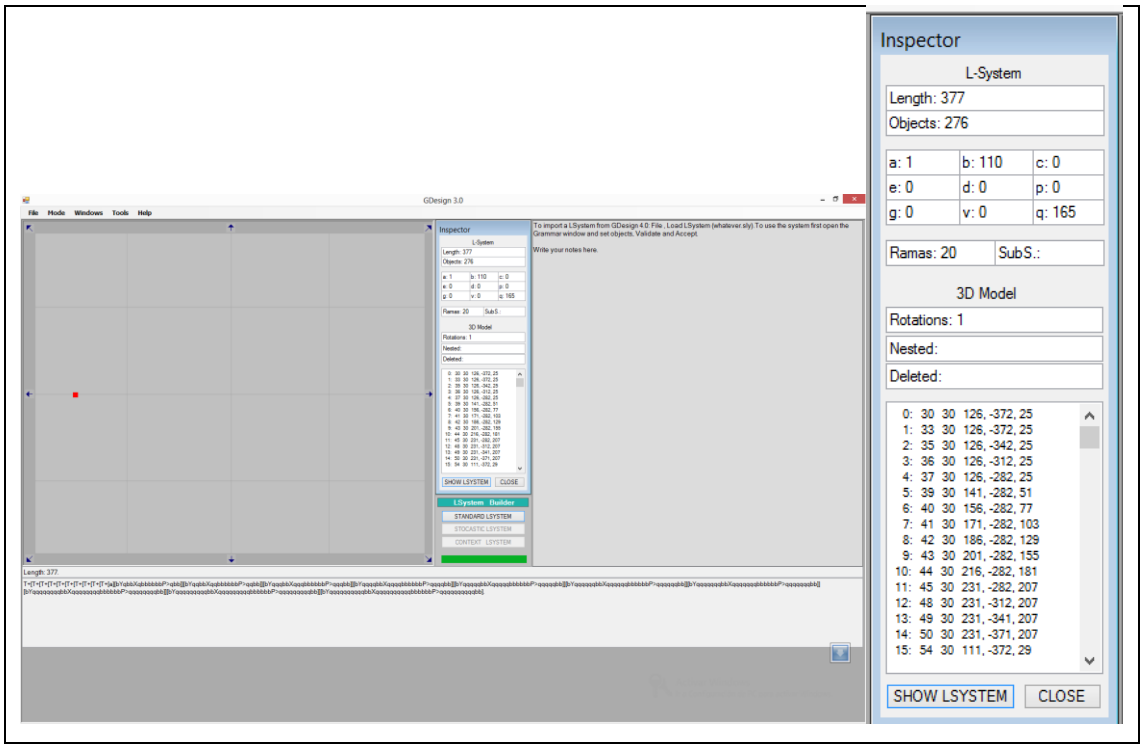


Fig. 8. Paleta para el análisis del modelo

Utilidades, *export* e *import,* postproducción

Auto Edit

La aplicación permite analizar el desarrollo del modelo 2D o 3D para controlar los objetos duplicados o incrustados, para evitar cruces o intersecciones (por ejemplo de ramas).

También se puede intervenir en la geometría, juntando vértices (weld) dentro un rango de vecindad fijado por un *threshold*.

Export e *import*

Los modelos pueden ser exportados a diferentes paquetes de diseño 3D, como Rhino (exportando el modelo como RhinoScript), 3DSMax (MaxScript), POVRay (Script). Además el modelo puede ser exportado en formato STL que es compatible con programas de diseño y de impresión 3D.

Postproducción

Utilidad externa a GDesign que permite la postproducción y la edición de los Sistemas-L. El archivo editado se importa en GDesign para ser generado o ulteriormente desarrollado.

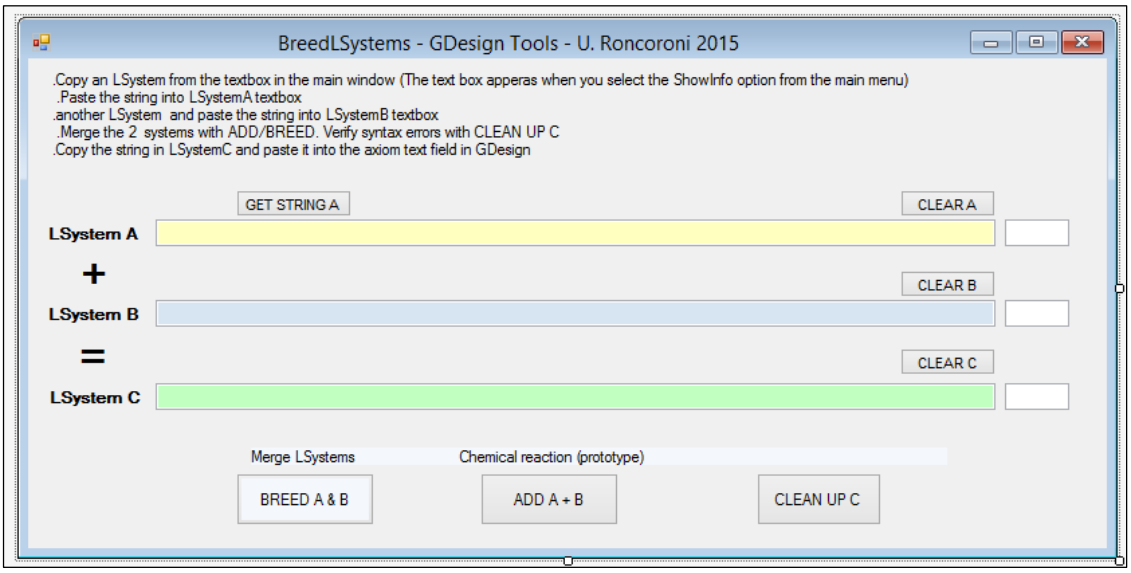


Fig. 9. Paleta para la hibridación de sistemas.

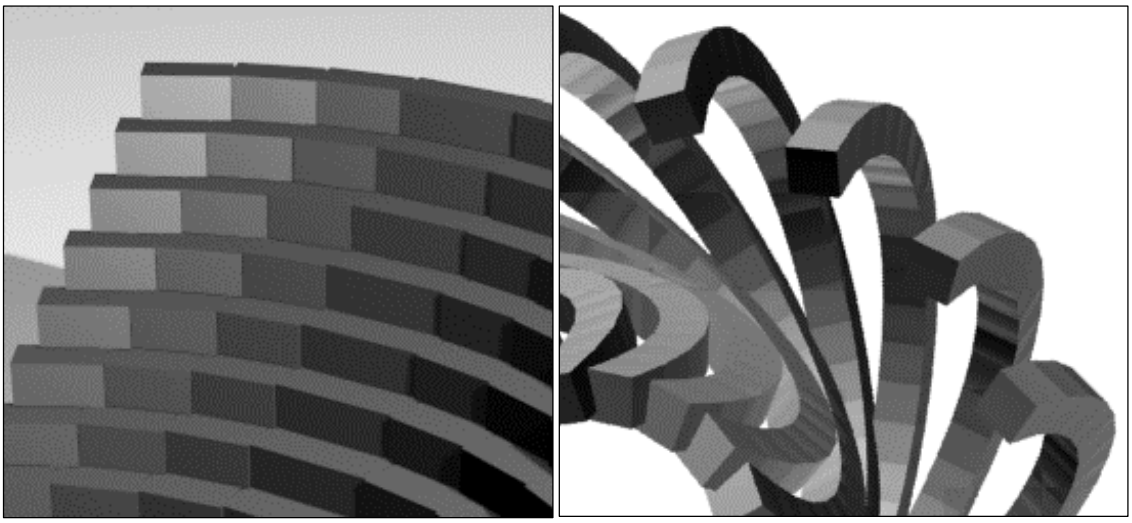


Fig. 10. A la izquierda ejemplo de vértices sin Weld (unión de vértices cercanos). A la derecha, los vértices son continuos por el weld.

Programación en postproducción

Permite un análisis del Sistema-L que no se puede hacer en tiempo de diseño, es decir en el proceso de sustitución recursiva.

- Mezcla de sistemas simbólicos
- Fusión de sistemas (química)
- Fusión de símbolos. Por ejemplo, “a” + “b” dan “c” en el sistema fusionado
- Inserto de sistemas en otros sistemas (botánica)
- Eliminación de elementos
- Expansión
- Contracción, reducción y compresión
- Contracción de un sistema (compresión). Por ejemplo “aaa” se contrae en “a”. “aaaaaa” en “aa”, etc.
- La cadena de un Sistema-L es como la cinta de una máquina de Turing, por lo tanto se podría procesar utilizando varios estados para reconocer ramas, rotaciones, etc.
- Programación de celular autómatas.

Cap. 3. Programación con Sistemas-L

Introducción

Los procesos simbólicos de sustitución recursiva como las *shape grammars* y los L-Systems son bastante conocidos y utilizados para generar formas complejas generalmente fractales. Un ejemplo son el copo de nieve de von Koch, o el triángulo de Sierpinsky, así como las generación de formas ramificadas (árboles, helechos, etc.) para lo cual han sido específicamente diseñados los L-Systems.

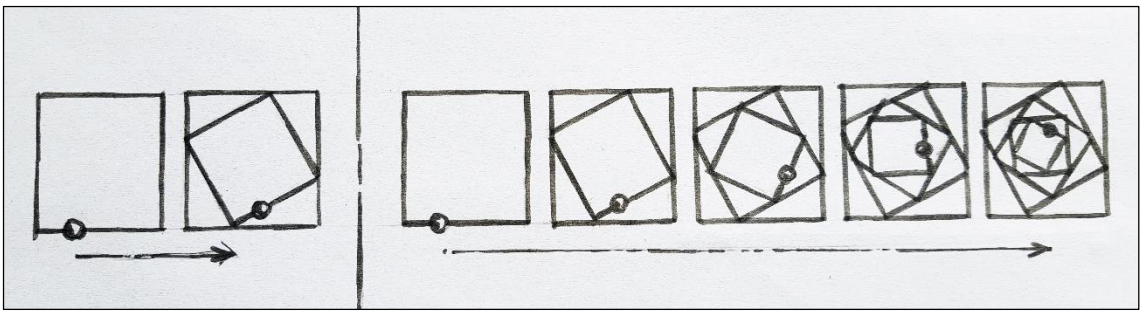


Fig. 1. Ejemplo de *shape gramar*

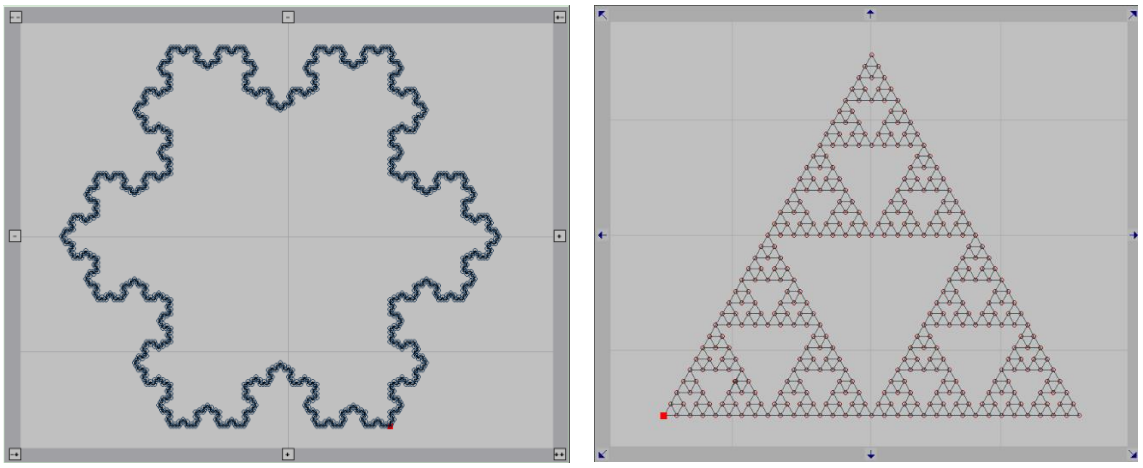


Fig. 2. Copo de nieve de von Koch y triángulo de Sierpinsky

Stiny y Gips han aplicado las gramáticas generativas para el estudio de patrones de diseño en la decoración, en las artes aplicadas y en la arquitectura. En efecto estos sistemas permiten la generación de formas complejas mediante representaciones simbólicas relativamente sencillas y computacionalmente relativamente económicas.

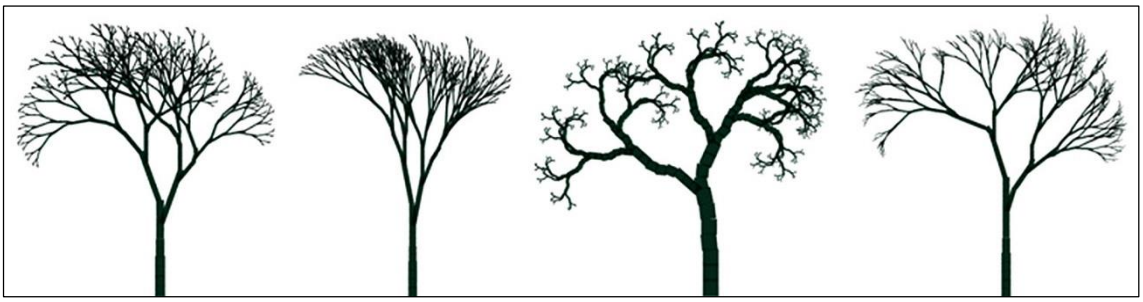


Fig. 3. Estructuras ramificadas 2D con Sistemas-L paramétricos y temporizados.

La representación simbólica de formas 2D y 3D permite un análisis peculiar de las formas complejas y también ofrece posibilidades originales para su edición o trasformación.

Es cierto que se han introducido variantes como las gramáticas de contexto, los sistemas temporizados y los sistemas paramétricos, que de alguna manera permiten un control del proceso recursivo por lo menos en términos cuantitativos. También se han conectado los procesos de sustitución recursiva con el entorno real mediante *physical computing*.

Sin embargo, los procesos de sustitución recursiva no son flexibles ni interactivos, y el control del proceso de sustitución es siempre determinado rígidamente sin posibilidad de alterarlo o modificarlo interactivamente o mediante los instrumentos de control de flujo normalmente utilizados en los lenguajes de programación tradicionales.

Objetivo del presente estudio es presentar un Sistema de sustitución recursiva basado en los L-Systems programables, mediante la introducción de nuevos símbolos y procesos dentro la sustitución tradicional, y por el otro el análisis de la interfaz para facilitar la programación al usuario. Pues si bien estos procesos son sencillos, no son predecibles pues su nivel de complejidad crece exponencialmente a cada etapa de la sustitución.

Por lo que se refiere a los símbolos, se hará una descripción de las siguientes técnicas de sustitución simbólica, que podríamos agrupar en las siguientes categorías:

- Símbolos para el control de flujo
- Símbolos automáticos
- Símbolos para los procesos aleatorios
- Subsistemas (*nested* L-Systems)

- Símbolos especiales
- Subsistemas

Por lo que se refiere a los ejemplos y a la interfaz, se hará un análisis de la aplicación GDesign 4.0 desarrollada por quien escribe en el marco del proyecto de investigación PIIM 2015.

De esta forma los procesos de sustitución recursiva y los L-Systems en modo especial se trasforman en un verdadero instrumento de programación cuyas peculiaridades permiten su aplicación en campos nuevos como el procesamiento de imágenes digitales y los efectos especiales.

Fig. 4. *Image processing* con objetos tridimensionales

3.1 Símbolos y técnicas para el control de flujo

El proceso de sustitución recursiva se desarrolla mediante una secuencia de sustituciones, en cada paso se genera una nueva cadena o serie de símbolos mediante un conjunto de reglas que pueden ser determinísticas, estocásticas o de contexto.

En esta sede no se hará una descripción de estos principios pues son generalmente conocidos.

El punto es que la secuencia de sustitución es fija y las implementaciones actuales no permiten personalizarlas, esto es, alterar o variar la secuencia de las reglas. Por lo tanto se presentan unas técnicas para superar estas limitaciones:

- Máscaras
- Implementación de bucles
- Saltos (*go to*)

Esto supone no solo la gestión de nuevos símbolos, sino un apropiado diseño de interfaz para poder gestionar el proceso en modo visual e interactivo.

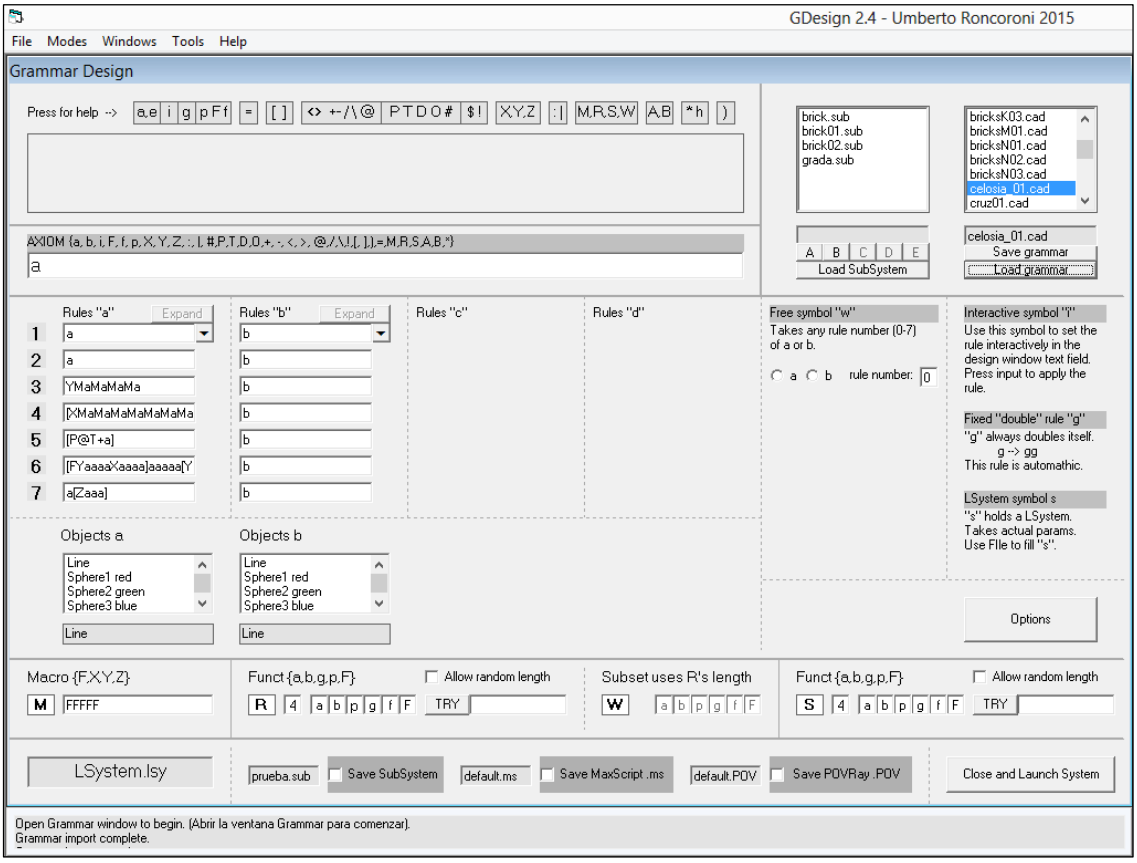


Fig. 5. Interfaz para el diseño de gramáticas en GDesign v. 2.4.

Cómo se puede notar en la fig. 5, las reglas de los símbolos son dispuestas en secuencia desde la 1 a la 7, lo que significa que la regla 1 se aplica en el primer proceso de sustitución, la regla 2 en el segundo, etc. Esto permite utilizar diferentes reglas en el proceso y establecer cuando utilizarlas. Ahora veremos cómo se puede afinar ulteriormente el control del proceso.

3.1.2 Máscaras. El símbolo “=”

El símbolo “=” sirve para bloquear o congelar un símbolo en el proceso en el cual está utilizado. Así en la cadena “aa=” y con la regla “a” → ”bb”, la sustitución dará como resultado “bba”, pues el segundo “a” está bloqueado por el “=”. Se pueden utilizar varios “=” seguidos, lo que permite bloquear un símbolo por más de un turno y con esto asignar diferentes reglas al símbolo y en momentos diferentes del proceso.

Esto permite realizar secuencias que con el normal proceso de sustitución serían imposibles.

Observes la secuencia mostrada en las figuras 6, 7, 8.

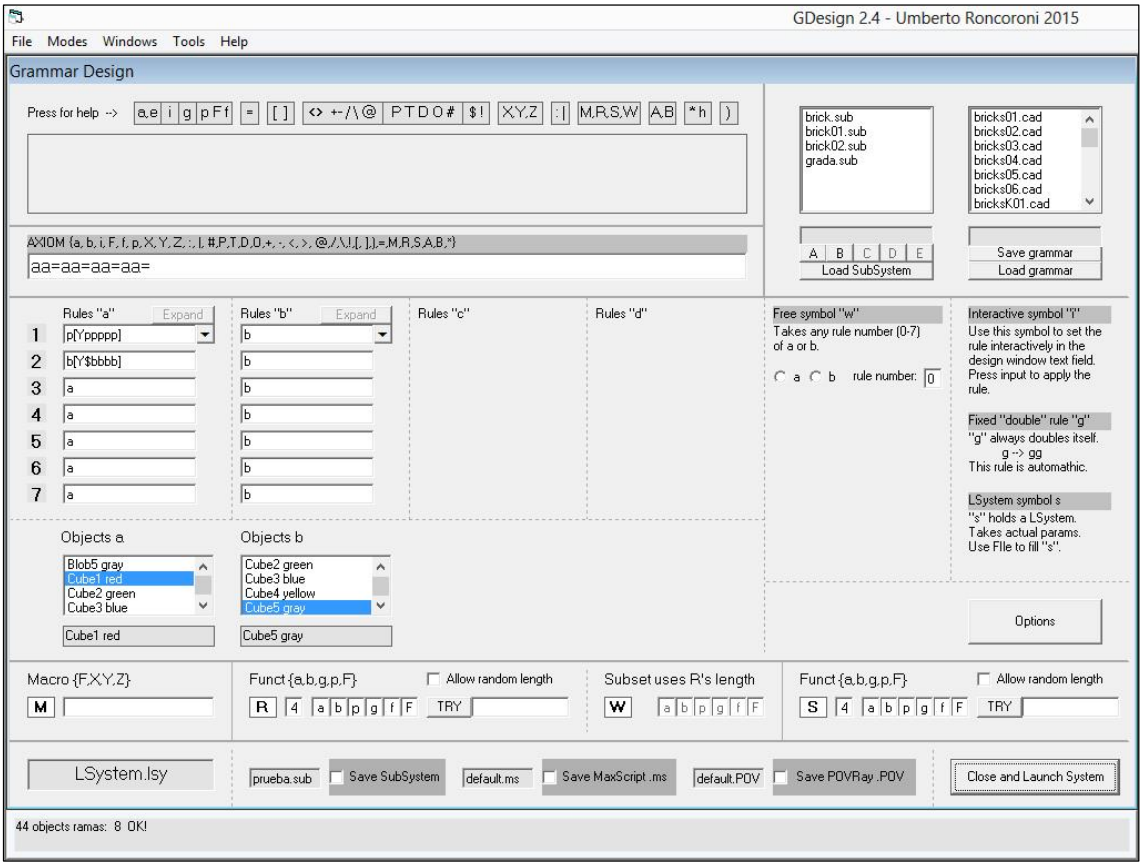


Fig. 5.

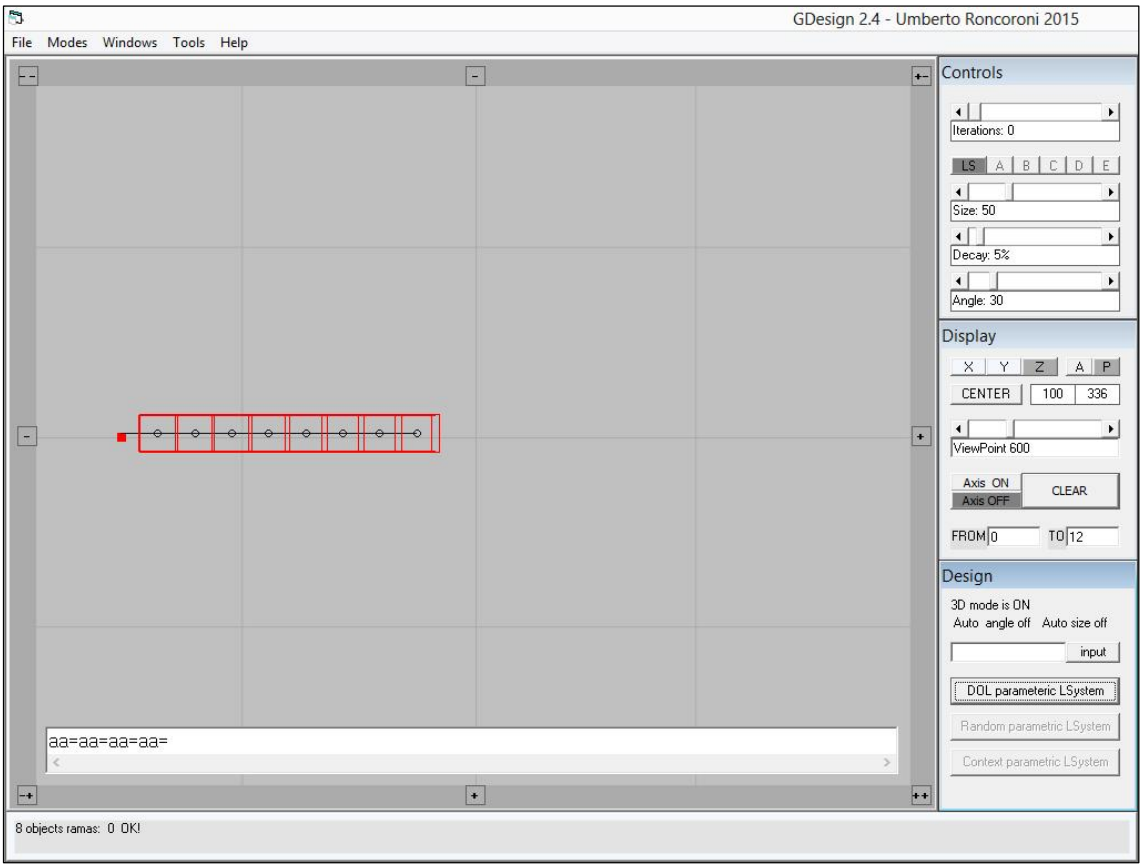


Fig. 6. Axioma.

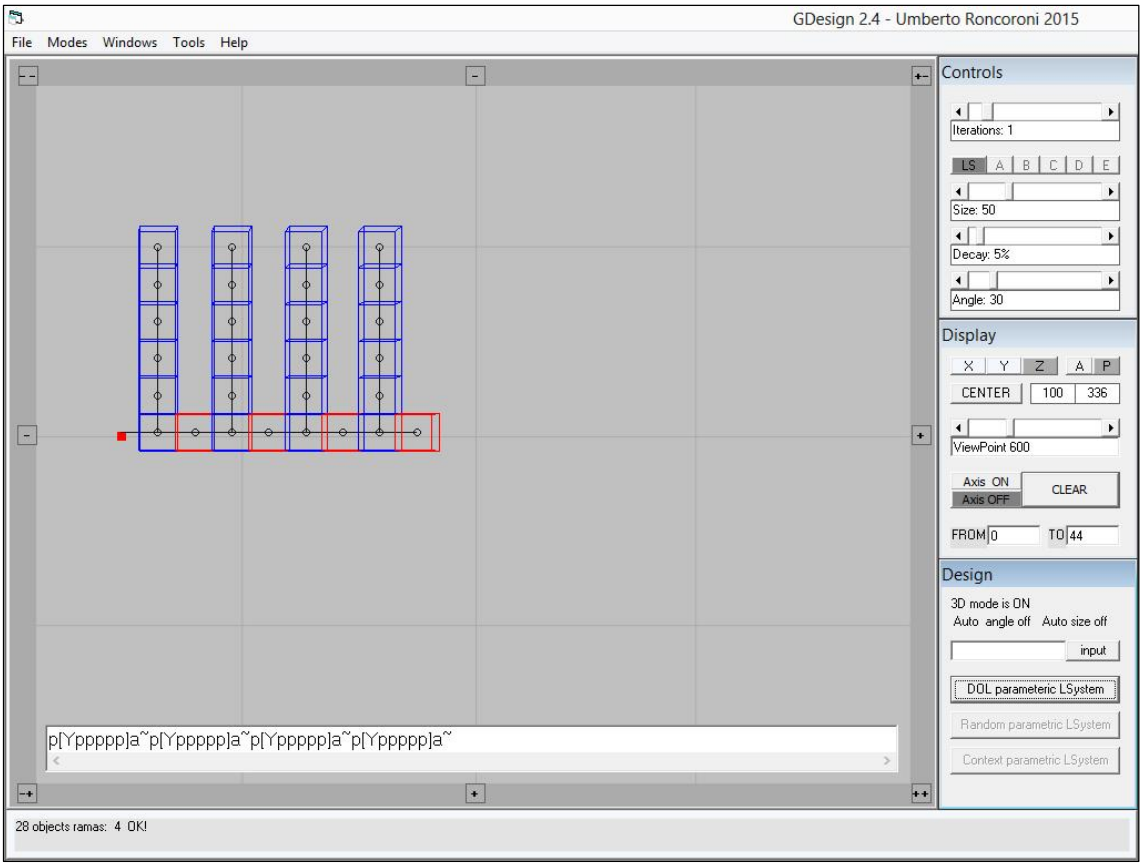


Fig. 7. Producción del símbolo “a” sin “=”.

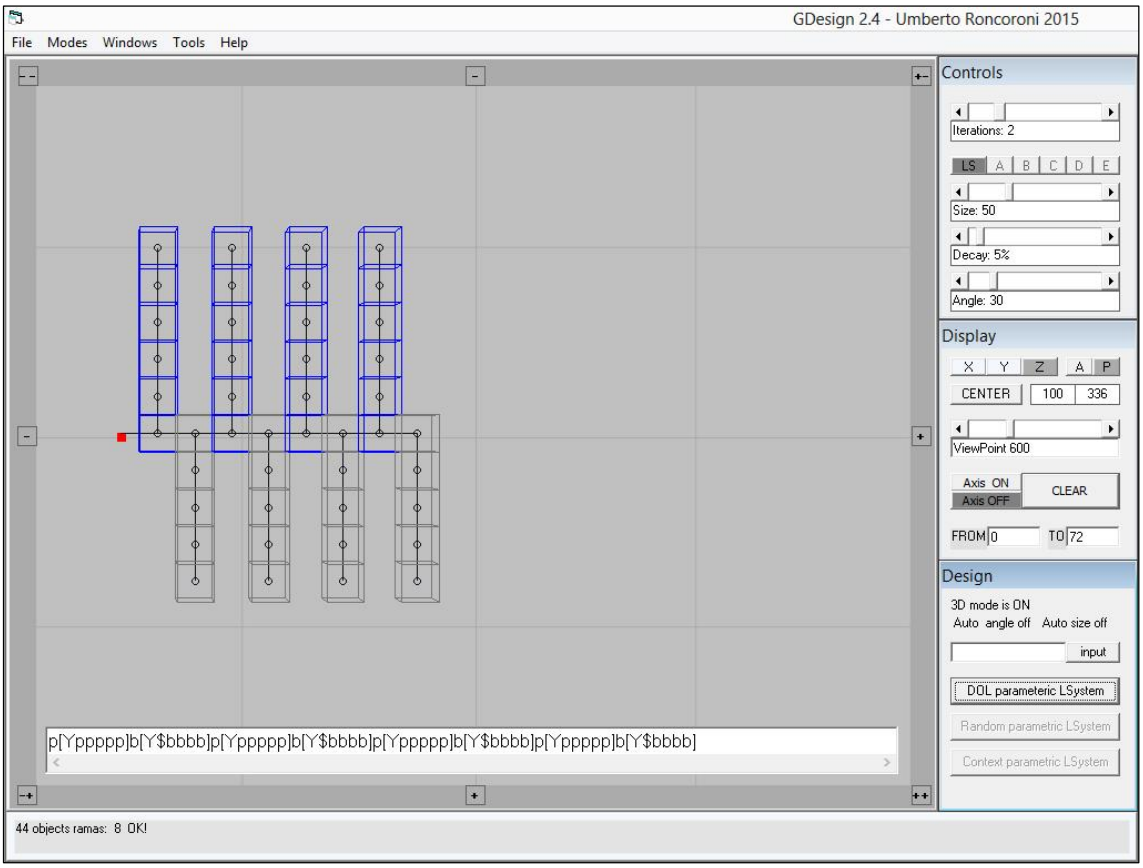


Fig. 8. Se sustituye el “a” bloqueado.

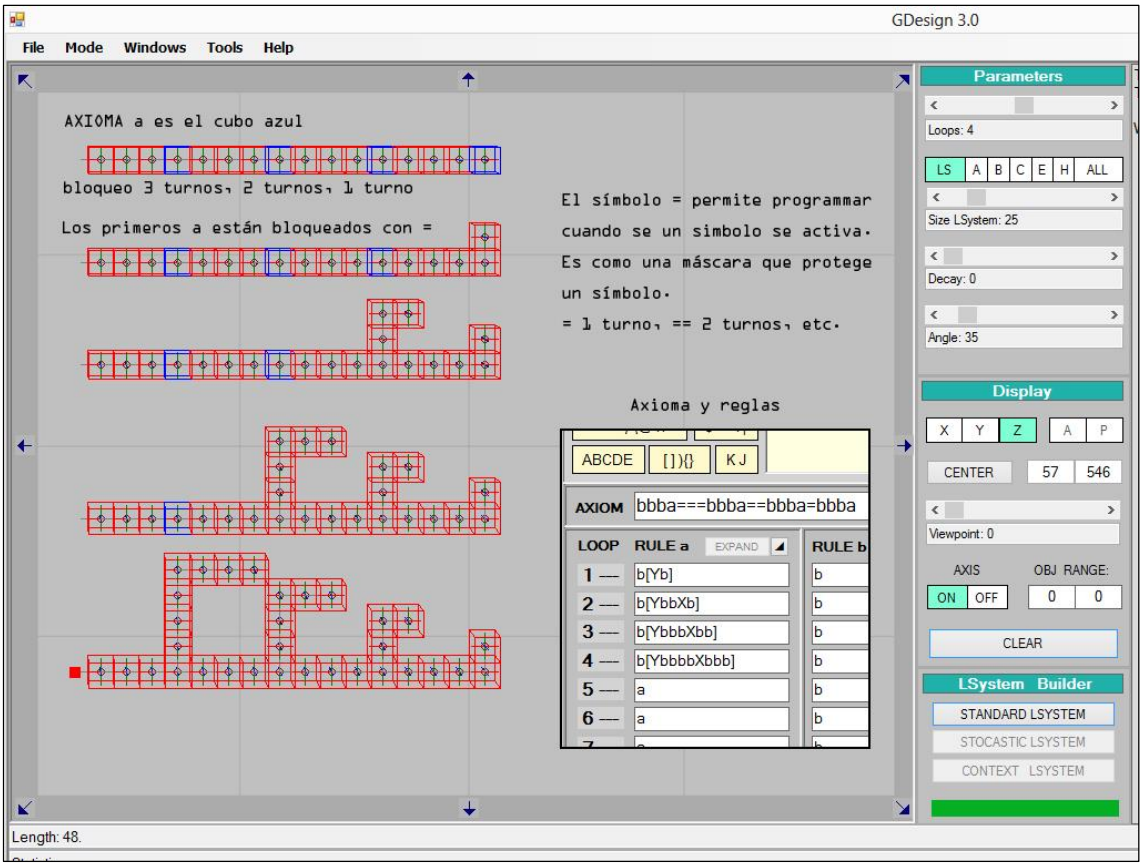


Fig. 9. Uso de múltiples “=”.

2.1.1 Saltos o *go to*. Símbolo “r”

El símbolo “r” permite utilizar cualquier regla de cualquier otro símbolo independientemente de la etapa del proceso de sustitución.

Este sistema permite solucionar varios problemas del control del proceso de sustitución.

En primer lugar constituye una especie de “go to” pues permite saltar de un símbolo a otro y de una etapa del proceso a otro. Esto es útil cuando por ejemplo se quiere replicar una regla aleatoria en modo de tener un proceso controlado.

“r” se puede utilizar en cualquier etapa del proceso, por ejemplo:

1. “r” → ”2a”
2. “r” → ”4b”
3. “r” → ”1a”
4. Etc.

Lo que significa que en la etapa 1 “r” se sustituye con la regla de la etapa 2 de “a”, en la etapa 2 con la regla de la etapa 4 de “b”, etc.

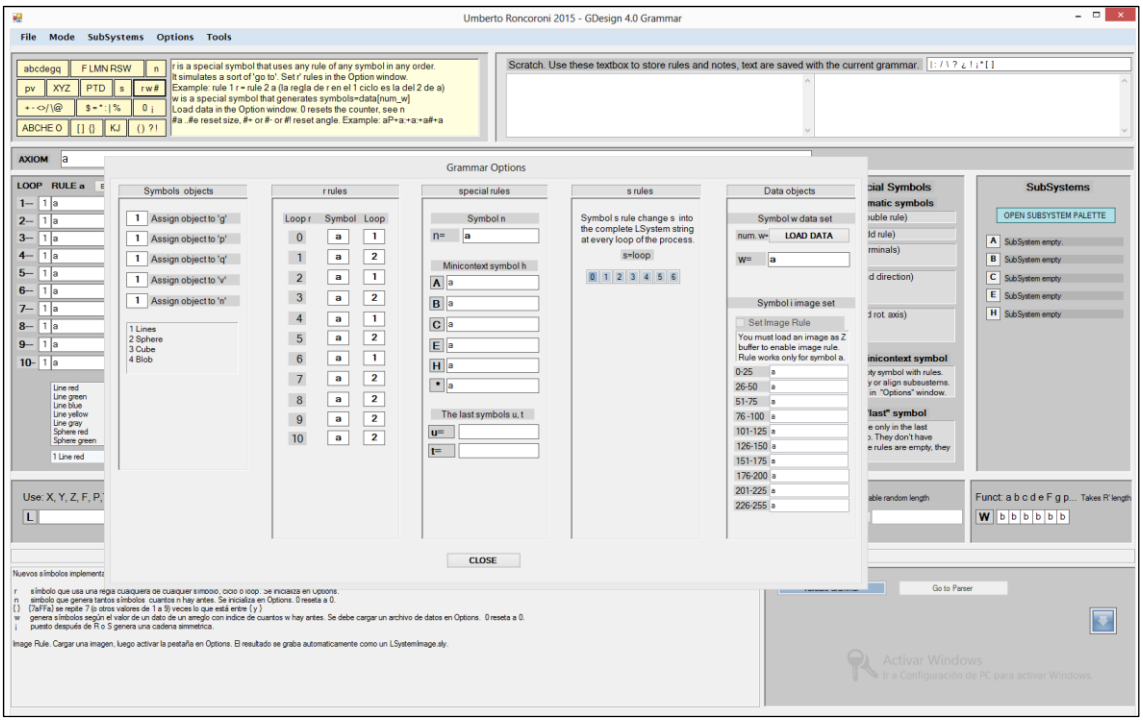


Fig. 10. Help y menú de reglas para el símbolo “r”.

2.1.4 Bucles

Permiten repetir n veces una regla o una cierta secuencia de símbolos.

Repetición recursiva de reglas

Es un parámetro que permite asigna el número de veces que se duplica una regla. Se asigna a nivel de interfaz. En fig. 11, el número a la izquierda de cada regla define cuanta veces se repite.

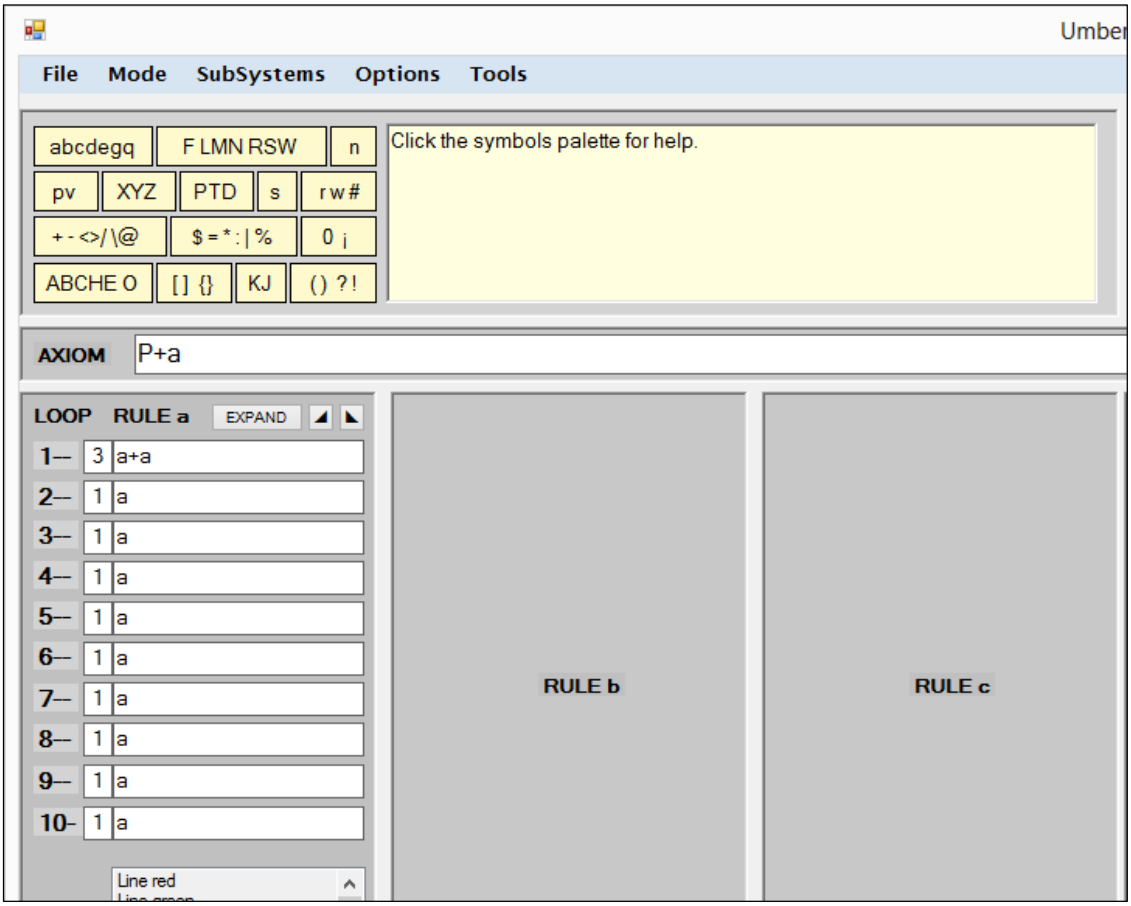


Fig. 11. La regla 1 se repite tres veces.

El bucle es recursivo, en efecto trasforma cada regla en un proceso recursivo para cada símbolo por lo cual se activa el bucle.

De esta forma se economiza en reglas y se pueden hacer procesos recursivos de una parte específica de un L-System.

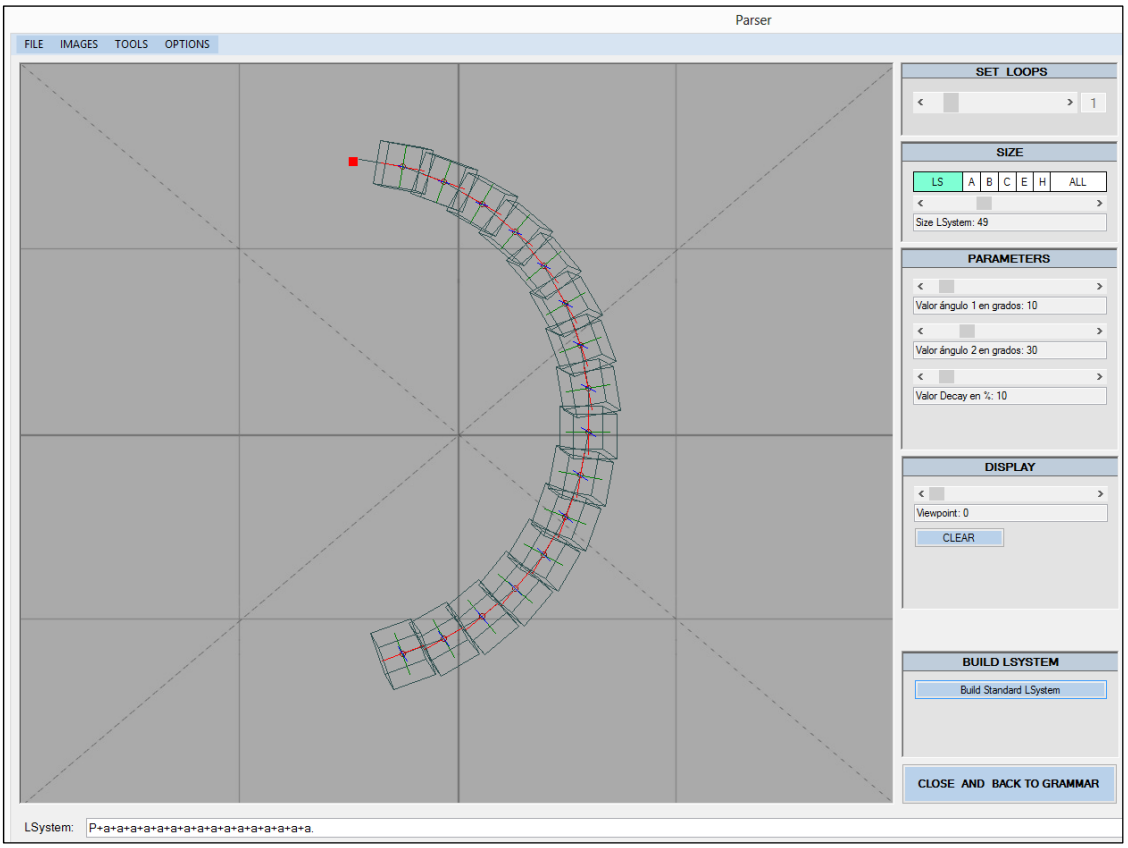


Fig. 12. El resultado de la etapa 1.

Esta técnica resulta cómoda para figuras autosimilares, pero puede llevar a resultados demasiado impredecibles. Es preciso utilizar en el bucle el símbolo "=" u símbolos terminales[4] para controlar la producción de símbolos en el bucle.

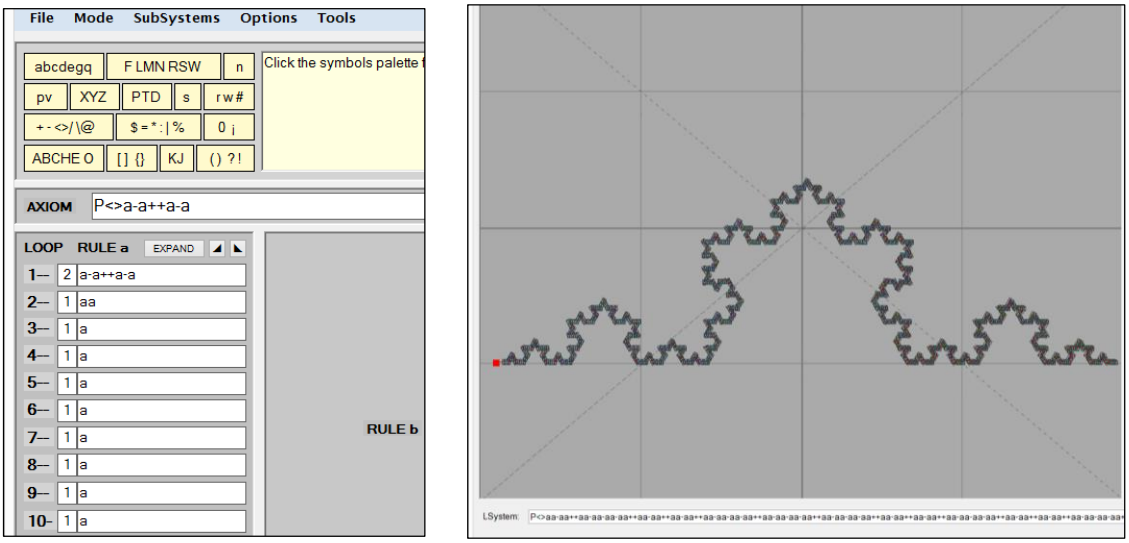


Fig. 13. Triángulo de Sierpinski con bucle en la regla 1

4

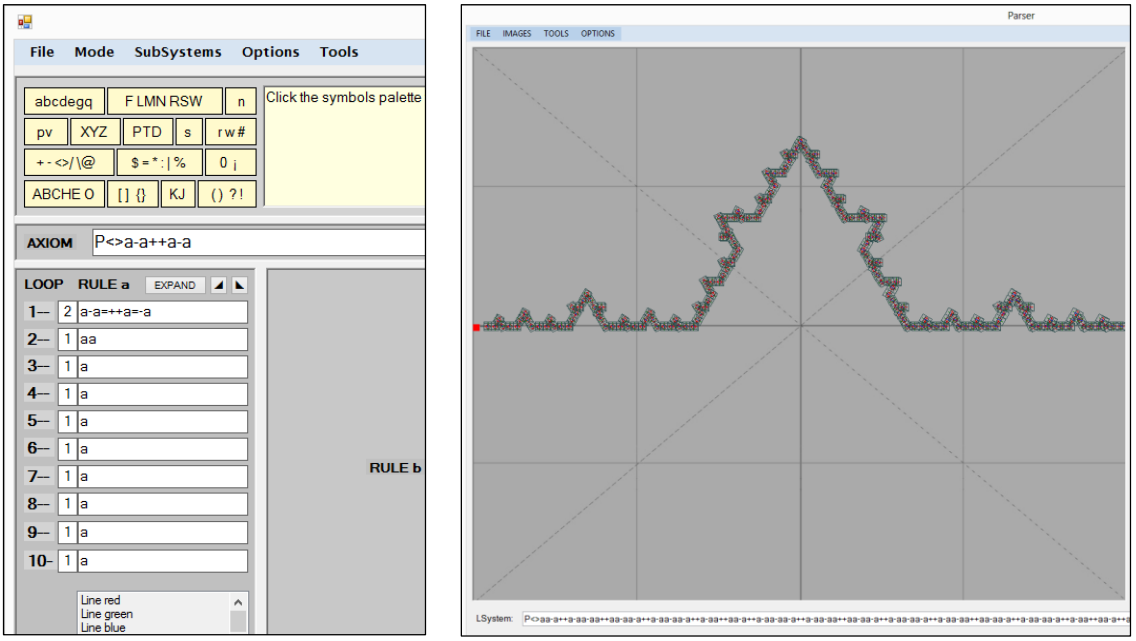


Fig. 14. Variación del triángulo de Sierpinsky con bucle en la regla 1 utilizando "="

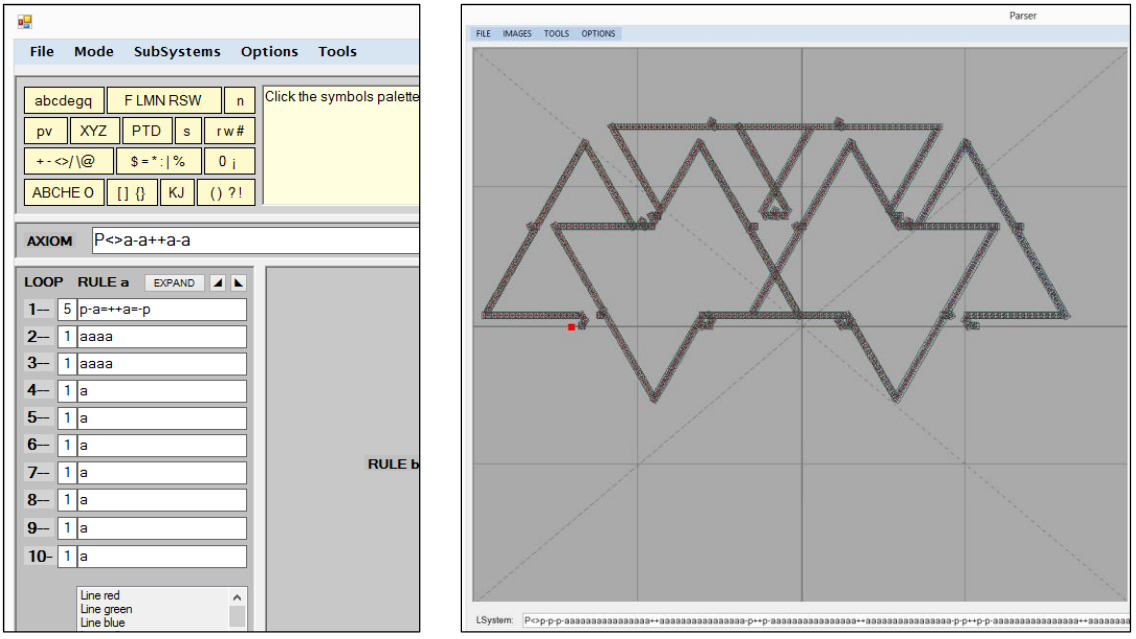


Fig. 15. Variaciones sobre Sierpinsky utilizando 5 repeticiones, "=" y el símbolo terminal "p"

Otro aspecto interesante de los bucles de reglas es que permiten variar la producción de cada símbolo con respecto a los demás, pues con el flujo normal todos producen una vez por símbolo.

Repeticiones de cadenas de símbolos dentro una regla

Otra forma para hacer bucles en repetir una determinada secuencia de símbolos dentro otra secuencia de símbolos.

El bucle de reglas repite todo el contenido de la regla. Para esto se utilizan los símbolos “{“, “}” y “1,2,3,4,5,6,7,8,9”.

Supongamos que tenemos “a” → ”P+ap+ppa”, y queremos repetir 6 veces la secuencia definida por la substringa “p+pp”.

Entonces definimos: “a” → ”P+a{6p+pp}a”. Lo que devuelve:

1 2 3 4 5 6
P+a p+pp p+pp p+pp p+pp p+pp p+pp a

Ciertamente se obtiene el mismo efecto escribiendo directamente:

“a”c “P+ap+ppp+ppp+ppp+ppp+ppp+pppa”

Pero la ventaja es que el valor de la repetición de esta forma puede ser un parámetro ligado al contexto del sistema. Es decir se puede determinar en base a los símbolos anteriores o posteriores o en base a la edad[5] del símbolo, o cualquier otra ocurrencia.

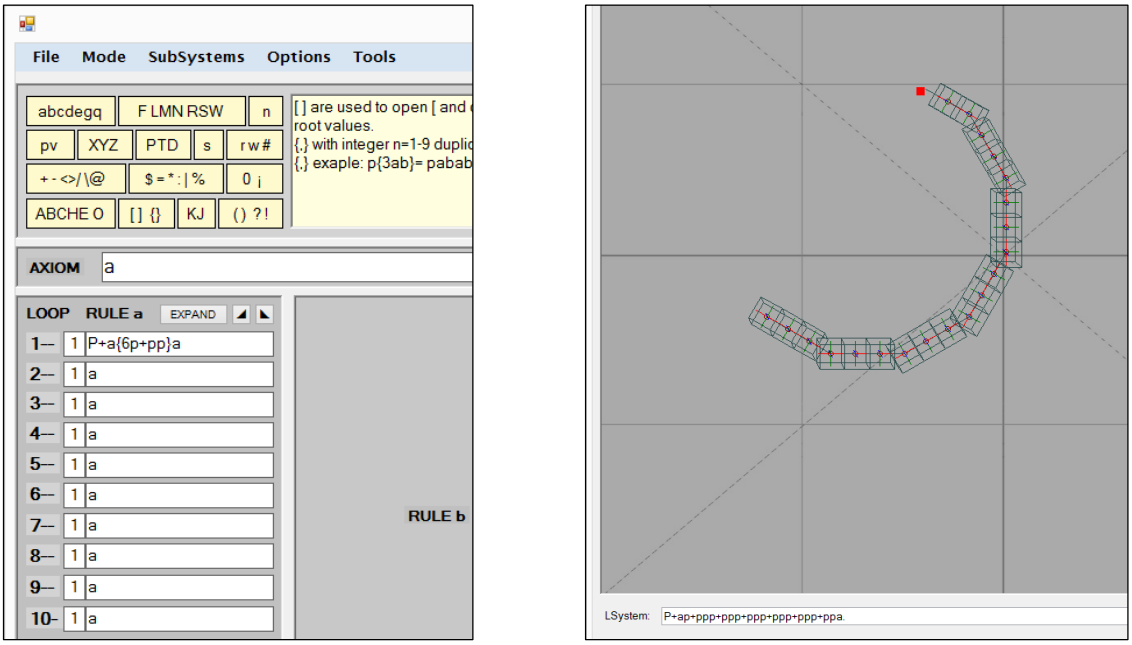


Fig. 16. Repeticiones de partes de una regla.

[5] Se entiende por cuantas repeticiones un cierto símbolo se encuentra en la misma posición

2.1.5 Símbolos híbridos “u” y “t”

Se trata de símbolos que son una mezcla entre símbolos no terminales y símbolos terminales. Estos se pueden utilizar en cualquier etapa del proceso, pero solo se activan al final, cuando todo el sistema ha finalizado.

Esto permite insertar elementos en las partes iniciales de una estructura, cuando si inserción normal daría lugar a producciones indeseadas, es como atrasar el desarrollo de algunas partes de la estructura.

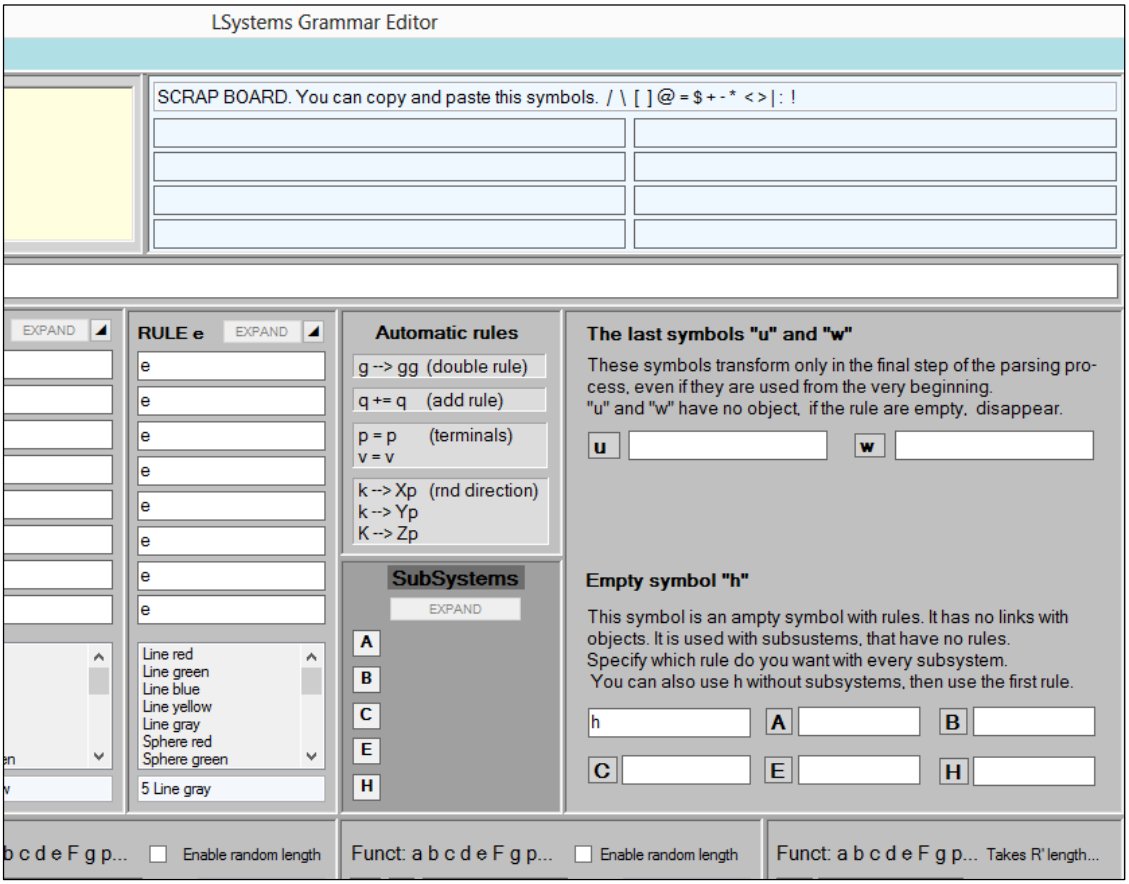


Fig. 17. Símbolos terminales con regla.

2.1.6 Implementación de *if … then* (verificación de condiciones)

Es posible introducir un símbolo para modificar el ángulo, el tamaño o la dirección de acuerdo a ciertas condiciones que se pueden verificar en el sistema.

En Opciones se establece la condición, por ejemplo podría ser el valor de la posición en X, Y o Z, la distancia de la raíz, la edad. El símbolo produce solo si se verifica

la condición fijada. Esto implica tener al día el arreglo de los objetos, que ya se hace pero hay que agregar un campo al dato.

```
//inicializamos los datos de la geometría
Diseñador.MyData = new Diseñador.objdata[30000];
for (int t = 0; t < 30000; t++)
{
    Diseñador.MyData[t].pos = 0;
    Diseñador.MyData[t].size = 0;
    Diseñador.MyData[t].Center = new Point3D(MyG);
    Diseñador.MyData[t].Center.X = 0;
    Diseñador.MyData[t].Center.Y = 0;
    Diseñador.MyData[t].Center.Z = 0;
    Diseñador.MyData[t].cubo = new Cube(MyG);
    Diseñador.MyData[t].cubo.V[0] = new Point3D(MyG);
    Diseñador.MyData[t].cubo.V[1] = new Point3D(MyG);
    Diseñador.MyData[t].cubo.V[2] = new Point3D(MyG);
    Diseñador.MyData[t].cubo.V[3] = new Point3D(MyG);
    Diseñador.MyData[t].cubo.V[4] = new Point3D(MyG);
    Diseñador.MyData[t].cubo.V[5] = new Point3D(MyG);
    Diseñador.MyData[t].cubo.V[6] = new Point3D(MyG);
    Diseñador.MyData[t].cubo.V[7] = new Point3D(MyG);
}
```

Fig. 18. La estructura de datos de un objeto del Sistema-L en GDesign.

2.2 Símbolos con reproducción automática

Se trata de implementar símbolos que se reproducen automáticamente, aumentando de uno en uno o duplicándose (o más).

2.2.1 Símbolo “g” y “q”

Para esto se usan: “q” y “g”. Cada vez que se encuentra un “q” se extiende la secuencia con un nuevo “q”, en el caso de “g” se dobla. Ejemplos:

1 “q” → ”qq”

2 ”qq” → ”qqq”

3 ”qqq” → ”qqqq”

4 ”qqqq” → ”qqqqq”

Este efecto de crecimiento es imposible utilizando el normal proceso de sustitución.

El caso de “g” que es doblar el símbolo, efectivamente se puede obtener con reglas normales, pero en modo automático resulta más sencillo.

1 “g” → ”gg”

2 ”gg” → ”gggg”

3 ”gggg” → ”gggggggg”

4 ” gggggggg” → ” gggggggggggggggg”

Esto es muy útil porque permite controlar las dimensiones de un sistema en base a las ocurrencias de un símbolo, que no se puede controlar por reglas directamente, pues no siempre se sabe dónde va el proceso de sustitución, es decir, no siempre se puede intuir. Además la producción automática se puede controlar reiniciando la cuenta o bloqueando “q” o “g” mediante el símbolo “=”.

Otra posibilidad es que el valor de multiplicación de “g” pueda variar de acuerdo a la etapa del proceso o al contexto, es decir en la etapa 1 “g” → ”gg”, pero en la etapa 2 “g” → ”ggg”, en la 3 “g” → ”g”, etc.

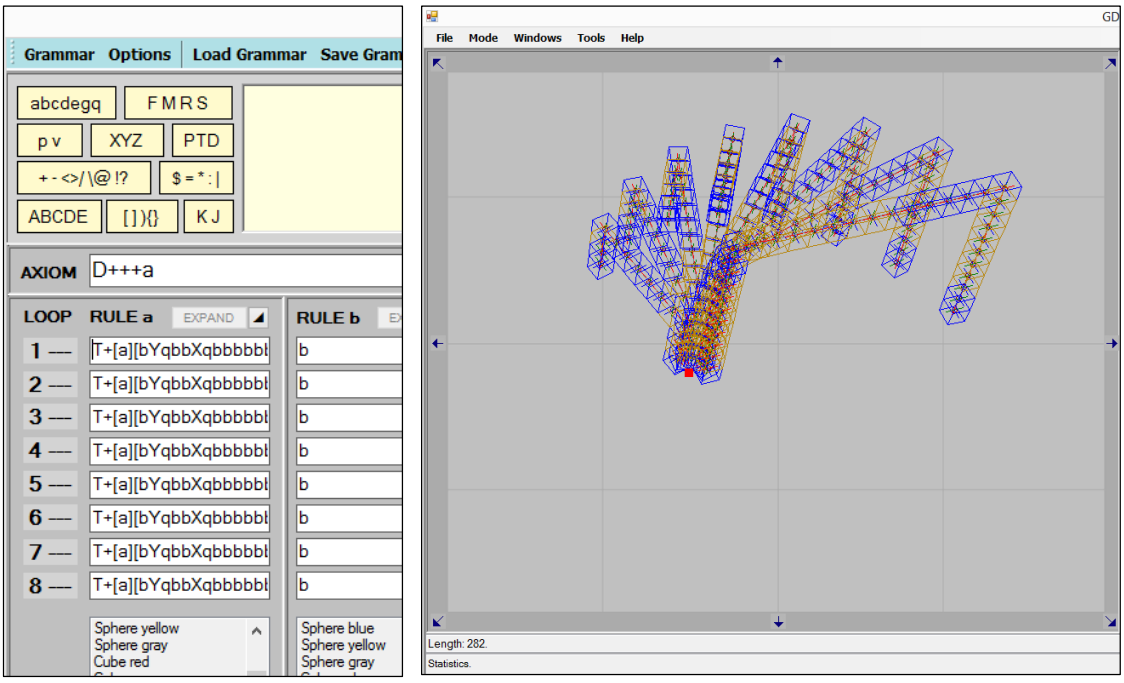


Fig. 17. Ejemplo de uso de “q”.

La regla completa de “a” es “T+[a][bYqbbXqbbbbbbP>qbb]”. Como “q” aumenta de 1 en 1, para aumentar la secuencia se puede copiar el resultado final al aplicarse la regla 8 en el axioma, y así se producen sistemas mucho más grandes.

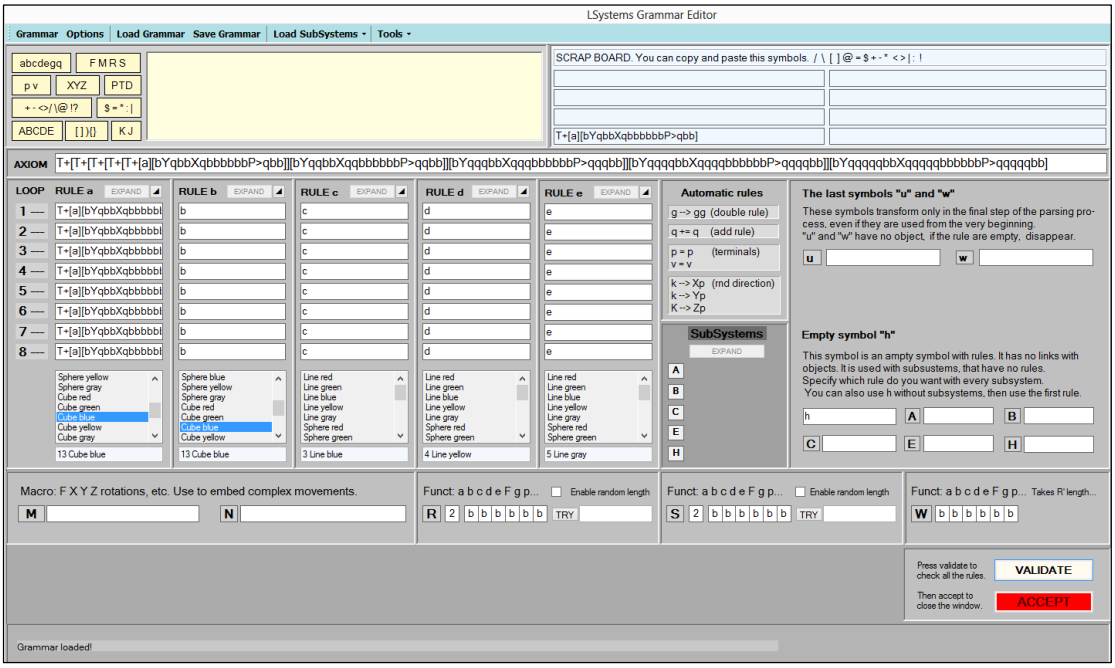


Fig. 18. Nótese el axioma, las reglas son iguales a las de la fig. 17.

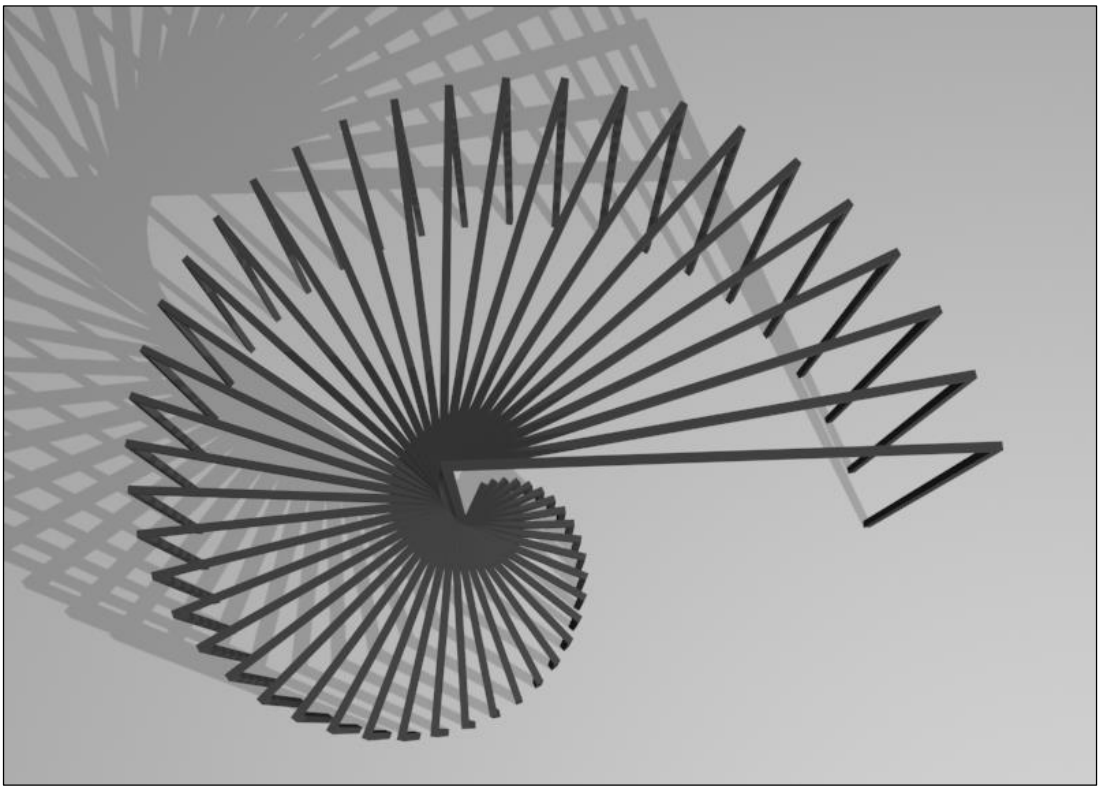


Fig. 19. *Render* del modelo

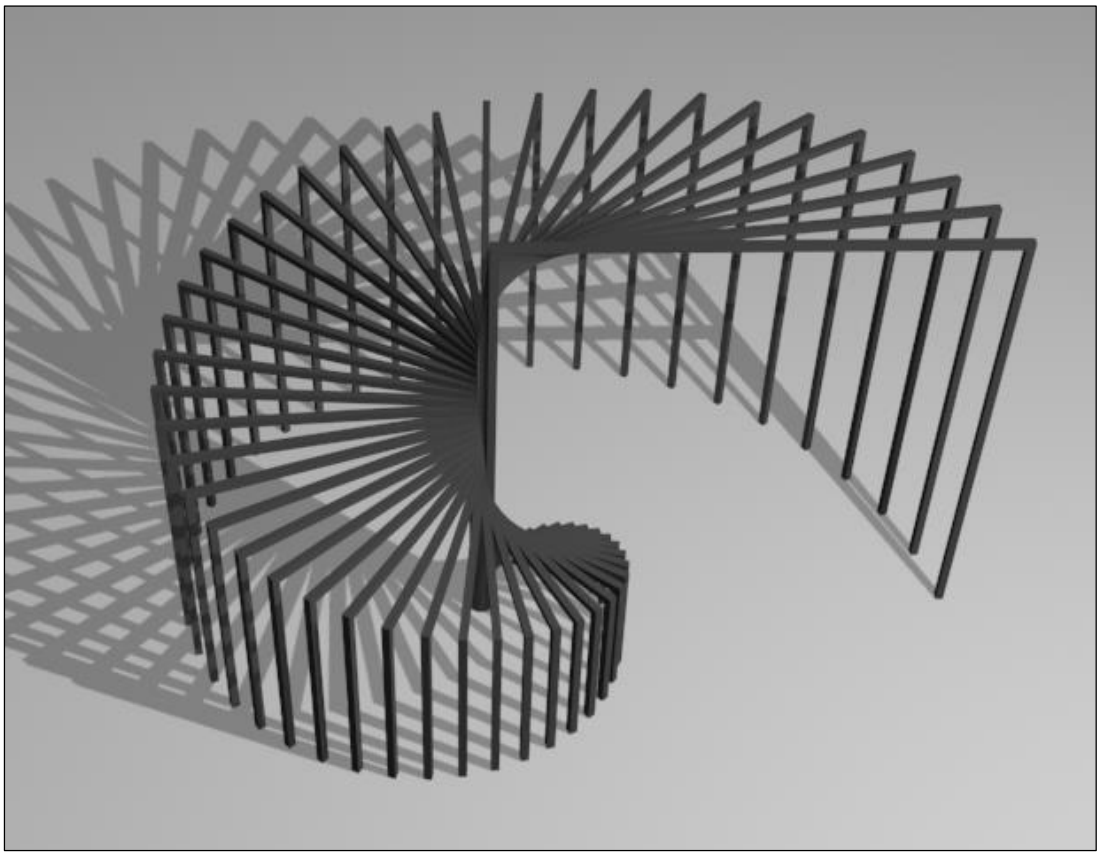


Fig. 20. *Render* del modelo.

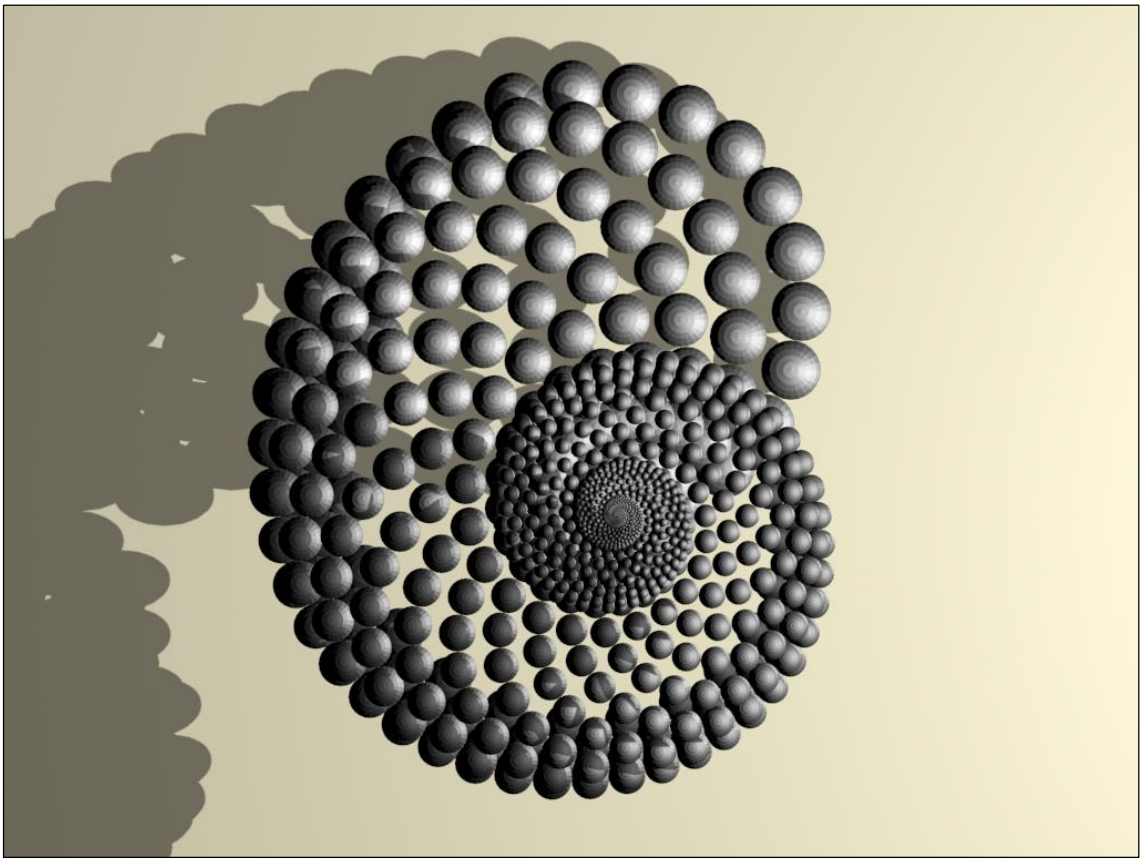


Fig. 21. Variación.

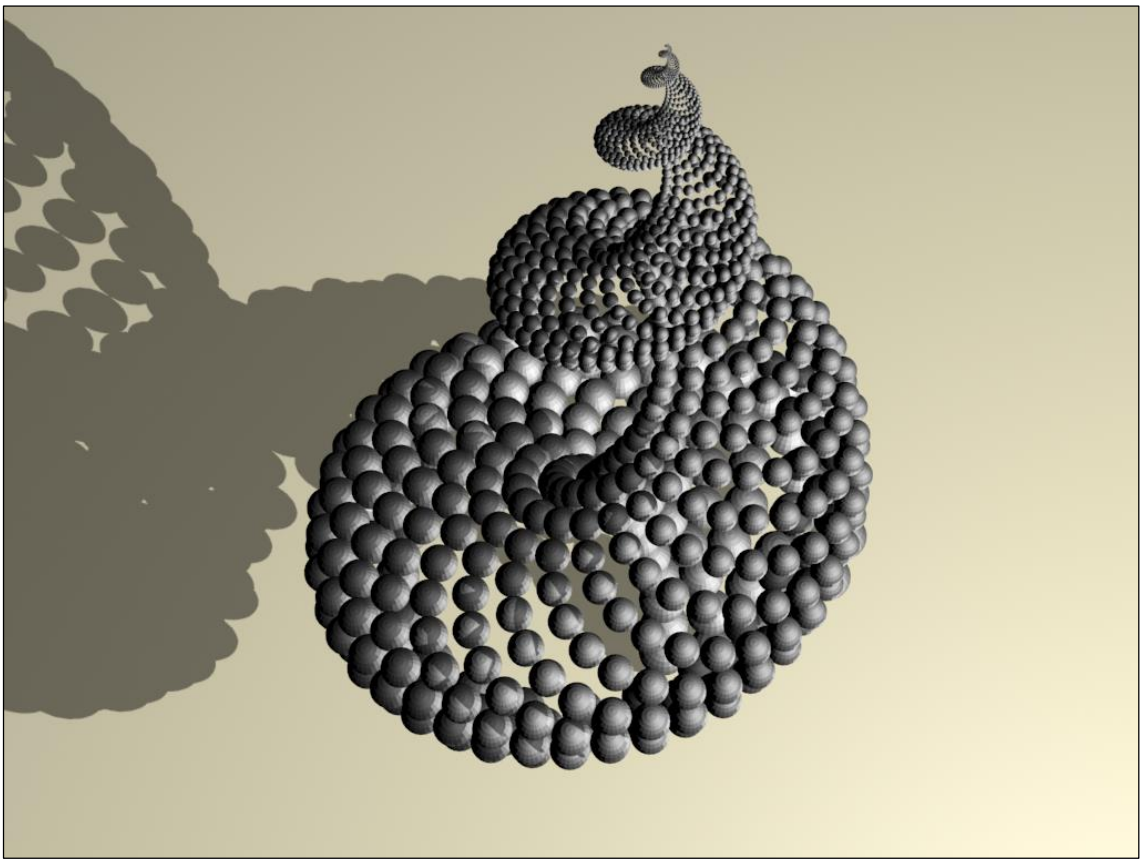


Fig. 22. Variación de las reglas de la fig. 19.

2.2.2 Símbolo “n”

El símbolo “n” es similar al símbolo q, en cuanto crece en cada ocurrencia que se encuentra en cualquier punto del sistema. Por otro lado tiene algunas características especiales: en primer lugar, cada ocurrencia se sustituye con una regla definida por el usuario, en segundo lugar puede ser reinicializado utilizando el símbolo cero “0”.

En el sistema :

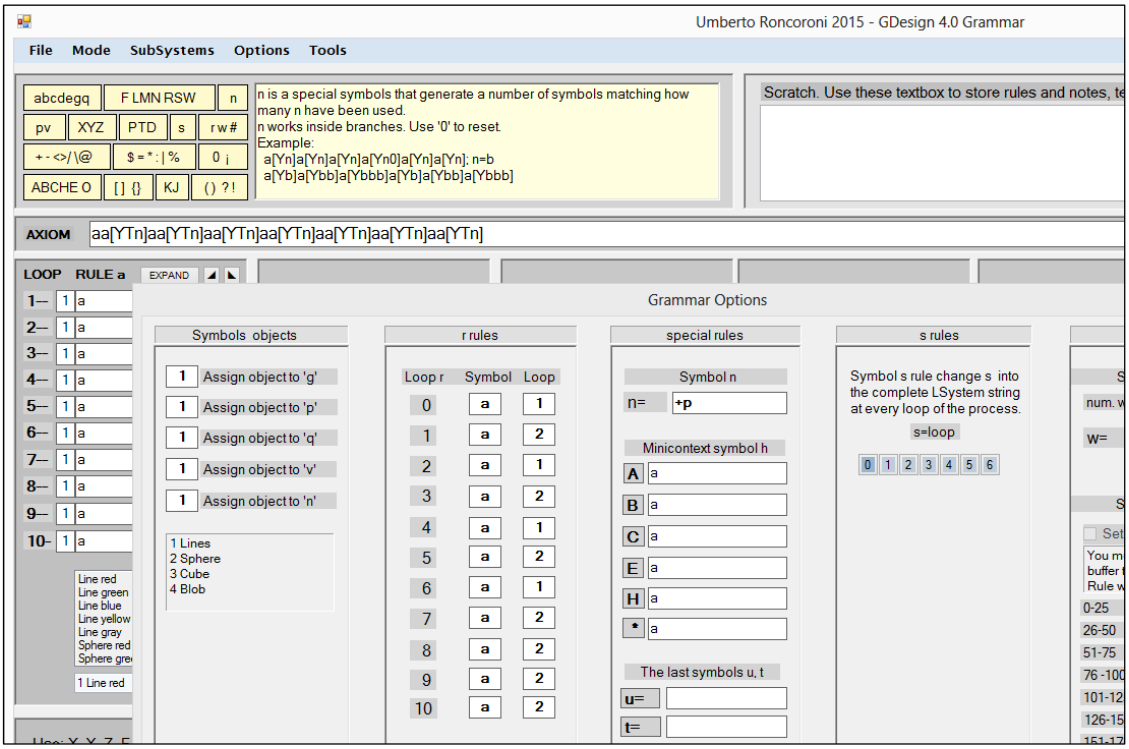
“aa[YTn]aa[YTn]aa[YTn]aa[YTn]aa[YTn]aa[YTn]“ y “n” → “+p”

La primera ocurrencia de “n” → “+p”, la segunda → “+p+p”, la tercera → “+p+p+p” etcétera.

Lo que da como resultado final:

“aa[YT+p]aa[YT+p+p]aa[YT+p+p+p]aa[YT+p+p+p+p]aa[YT+p+p+p+p+p]aa[YT+p+p+p+p+p+p]aa[YT+p+p+p+p+p+p]”.

El símbolo “n” puede ser sustituido con cualquier secuencia de símbolos que pertenecen al lenguaje.



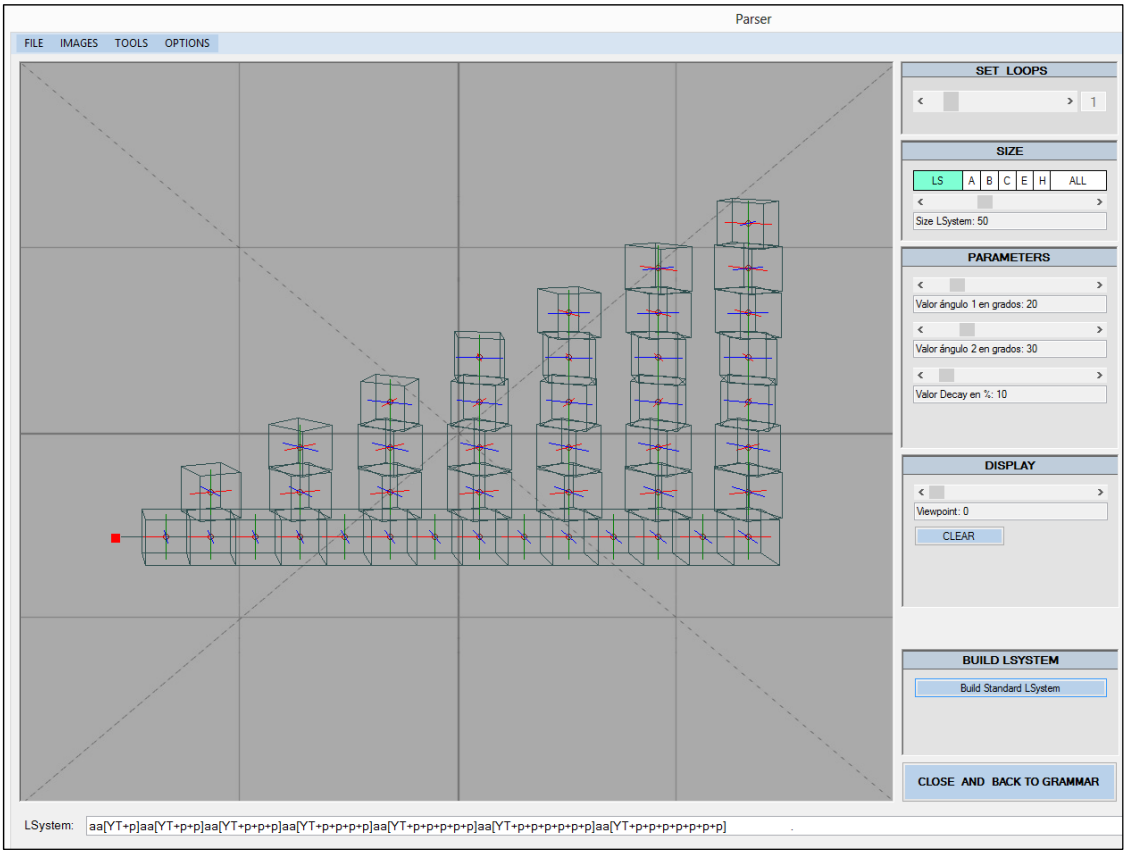


Fig. 23- 24. Ejemplo de “n” y *rendering.*

Utilizando en algún punto el símbolo “0” la secuencia recomienza de 1. En el sistema: “aa[YTn]aa[YTn]aa[YTn]aa[YT0n]aa[YTn]aa[YTn]aa[YTn]“ el resultado será: “aa[YT+p]aa[YT+p+p]aa[YT+p+p+p]aa[YT+p]aa[YT+p+p]aa[YT+p+p+p]aa[YT+p+p+p+p]”

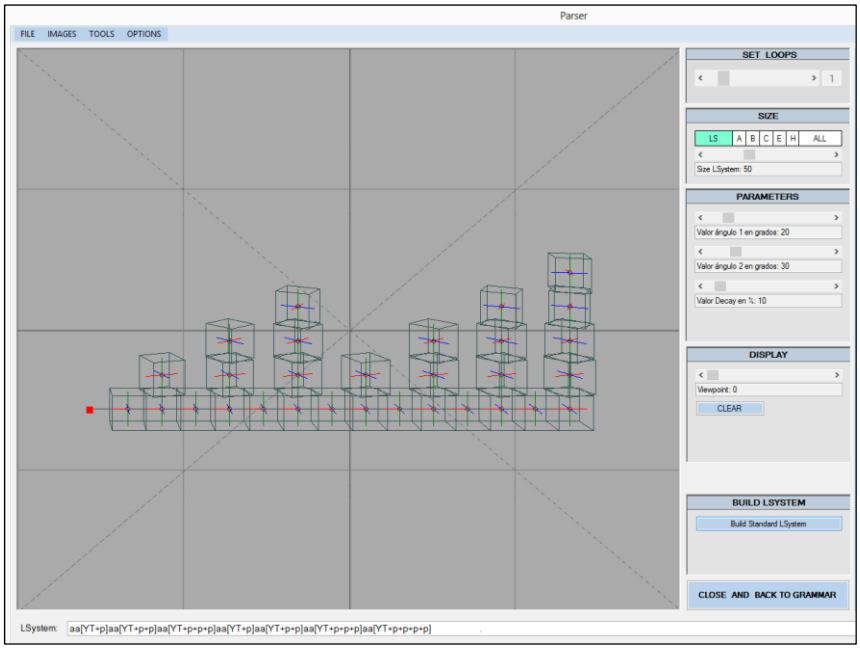


Fig. 25. Variación de las reglas de la fig. 19.

Este símbolo permite programar estructuras secuenciales ordenadas, como en el ejemplo que sigue.

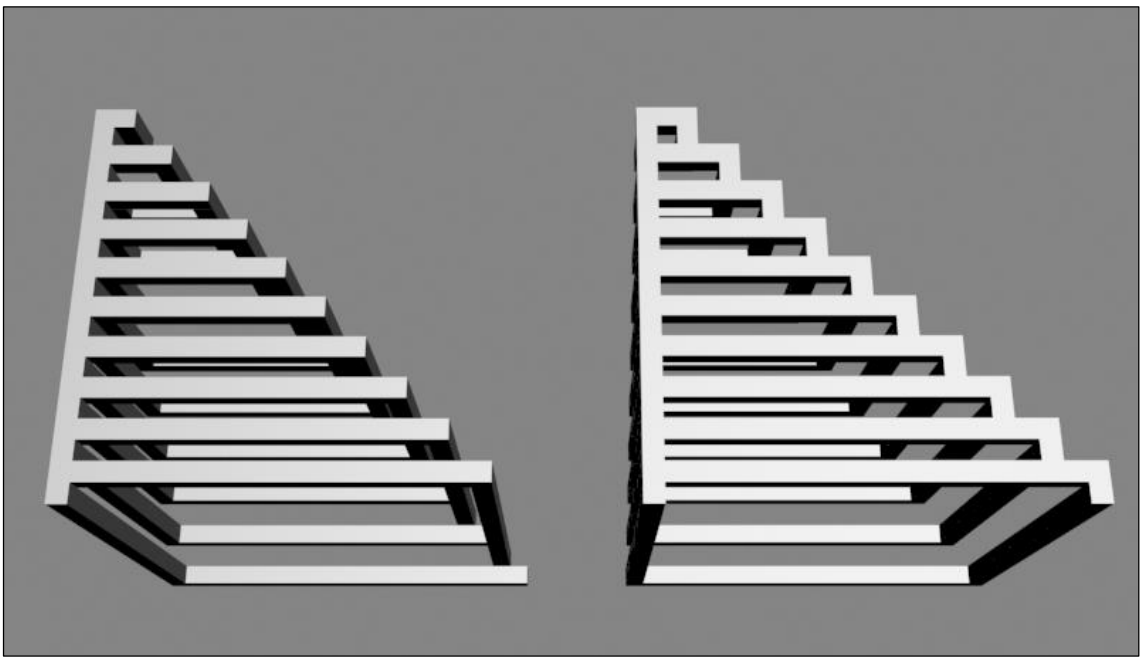


Fig. 26. Estructuras secuenciales

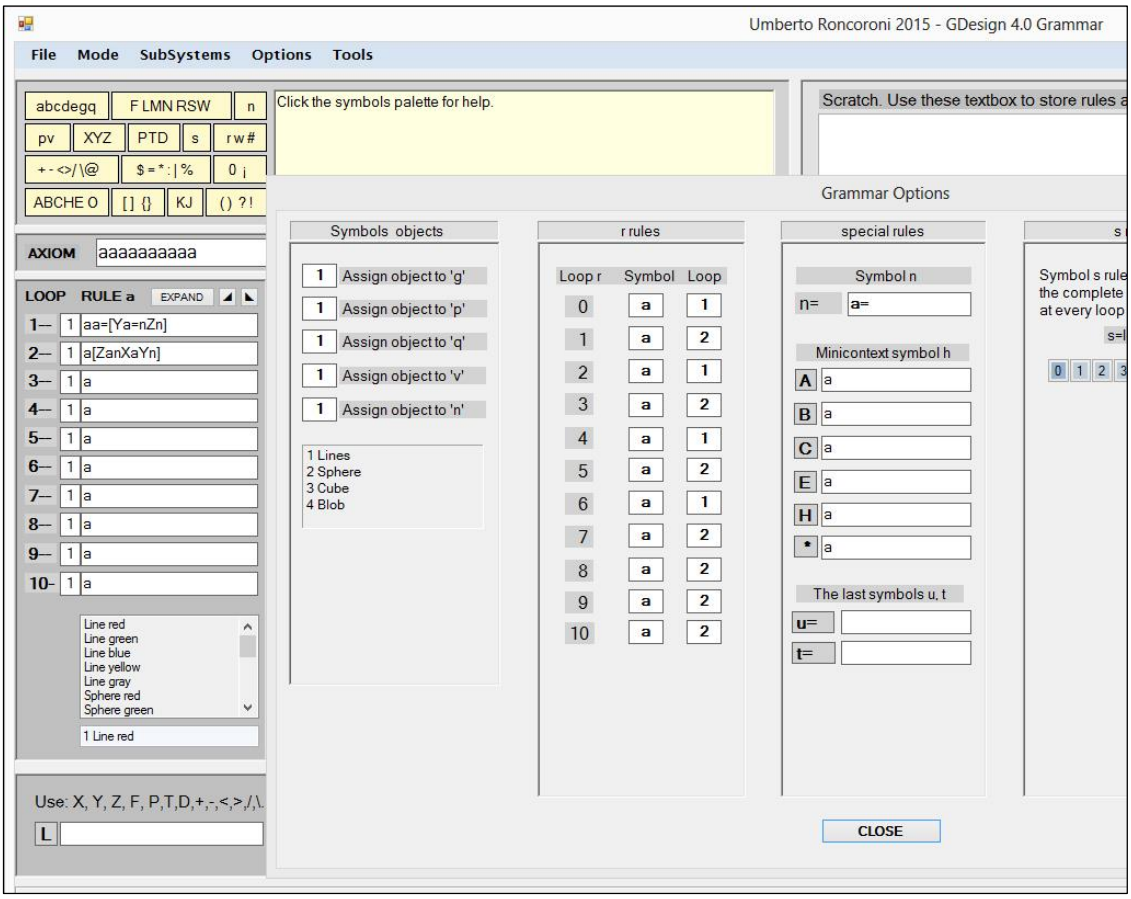


Fig. 27. Las reglas de la fig. 25.

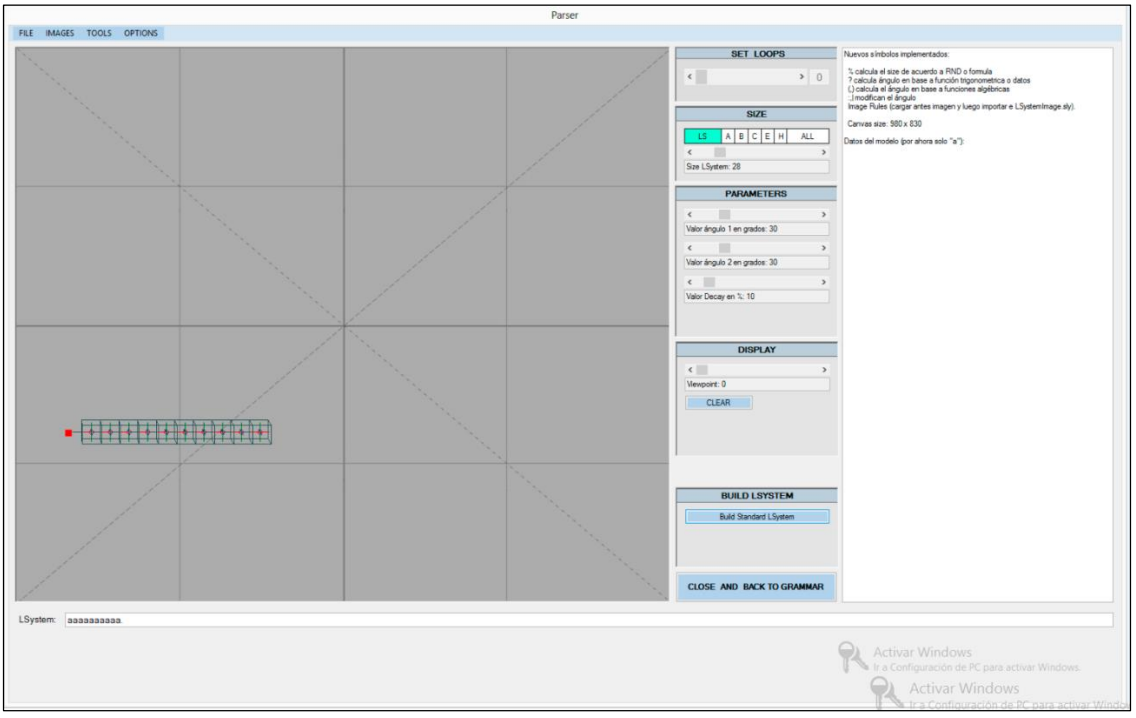


Fig. 28. Axioma de la gramática di fig. 26.

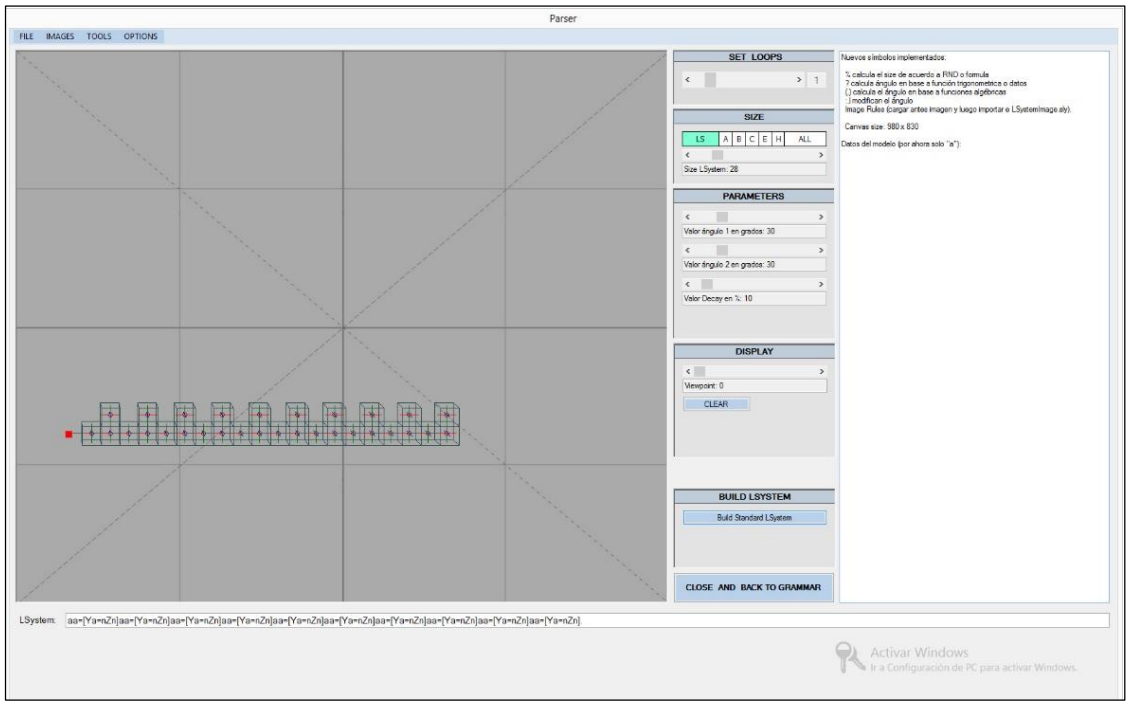


Fig. 29. Paso 1.

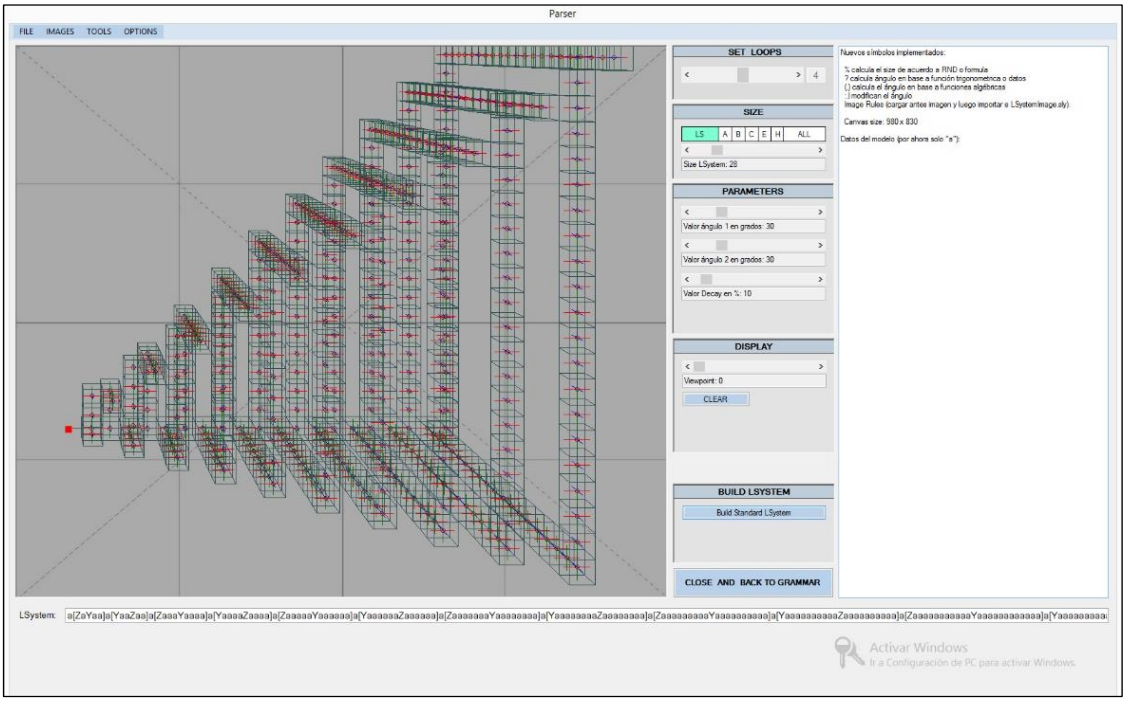


Fig. 30. Paso 2.

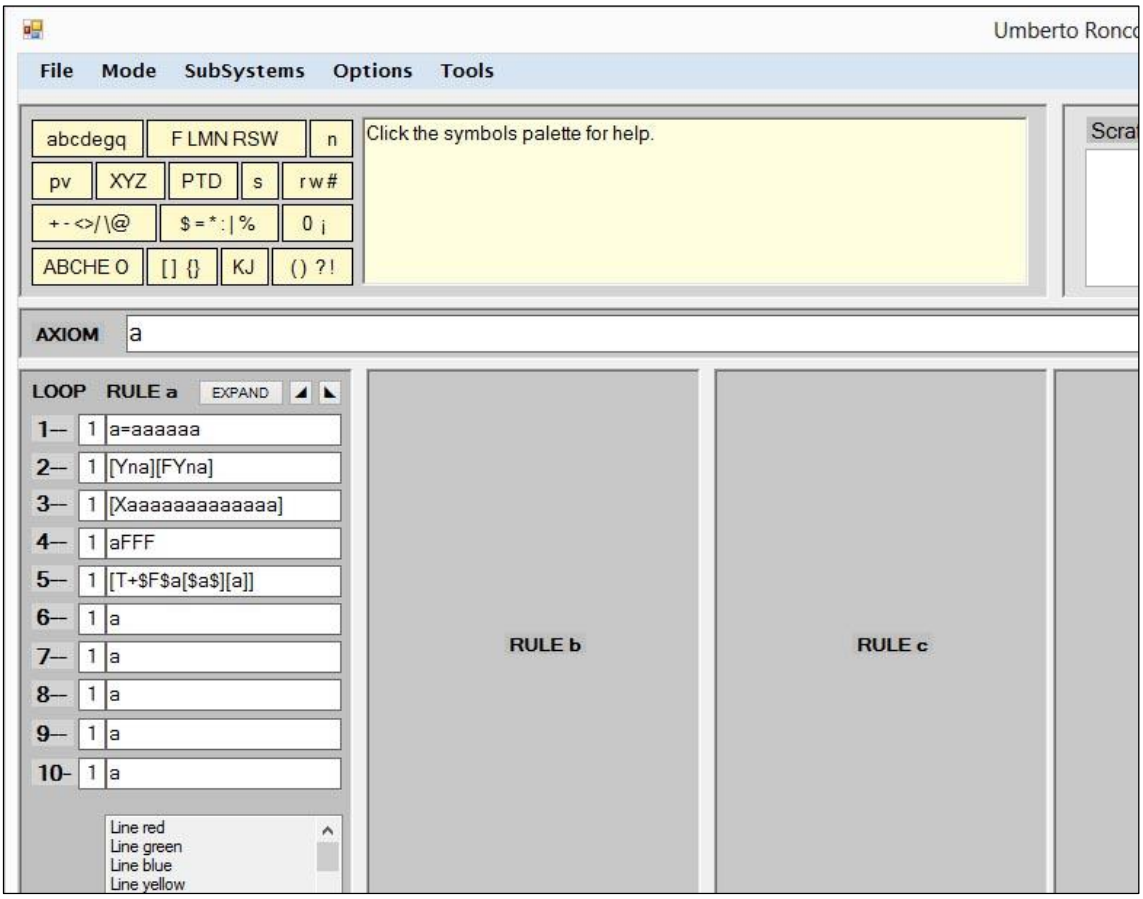


Fig. 31. Gramática de ladrillos con “n”.

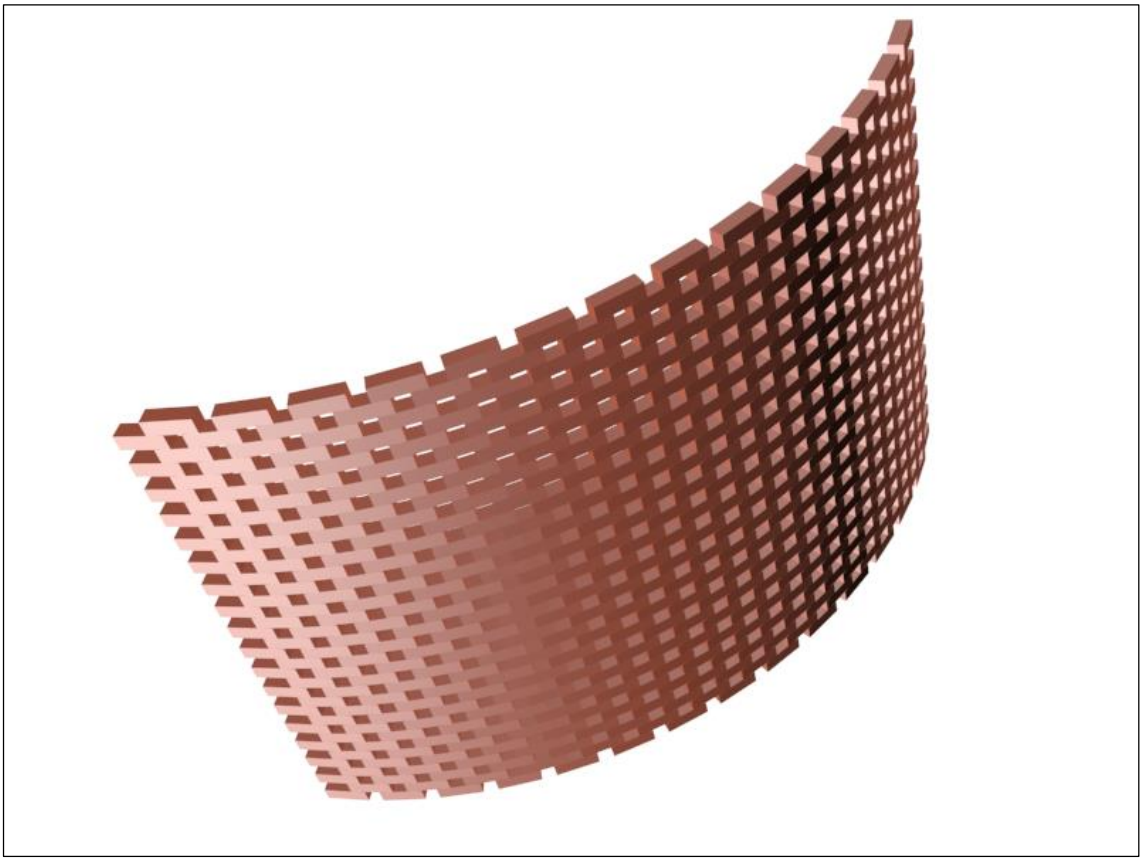


Fig. 32. Gramática de ladrillos con “n”.

Programación de “n”

Posibilidad:

- restar en lugar de sumar. “ñ” o “n!” para indicar que se comienza a restar
- bloquear con “=”
- poner un tope máximo
- poner diferentes reglas para cada ciclo (etapa del proceso o *loop*)
- o poner n1, n2, n3, n4, n5, n6

El valor de “n” dentro la regla de “n”. El símbolo puede ser “ñ”.

Supongamos que “n” → “aañ”, y que “ñ” → “b”. Si “n” vale 1 (el primer “n” de una serie”, entonces “ñ” → “b”, es decir vale 1, entonces “n” → “aab. Si “n” vale 3 (el 3 “n” de la serie) entonces “ñ” → “bbb”, es decir vale 3, entonces “n” → “aabbb”.

También ñ puede ser ñ1, ñ2, etc.

Con este sistema, utilizando varias reglas de “n”, el bloqueo con “=” y el bucle interno a “n” con “ñ”, es posible hacer estructuras basadas en bucles.

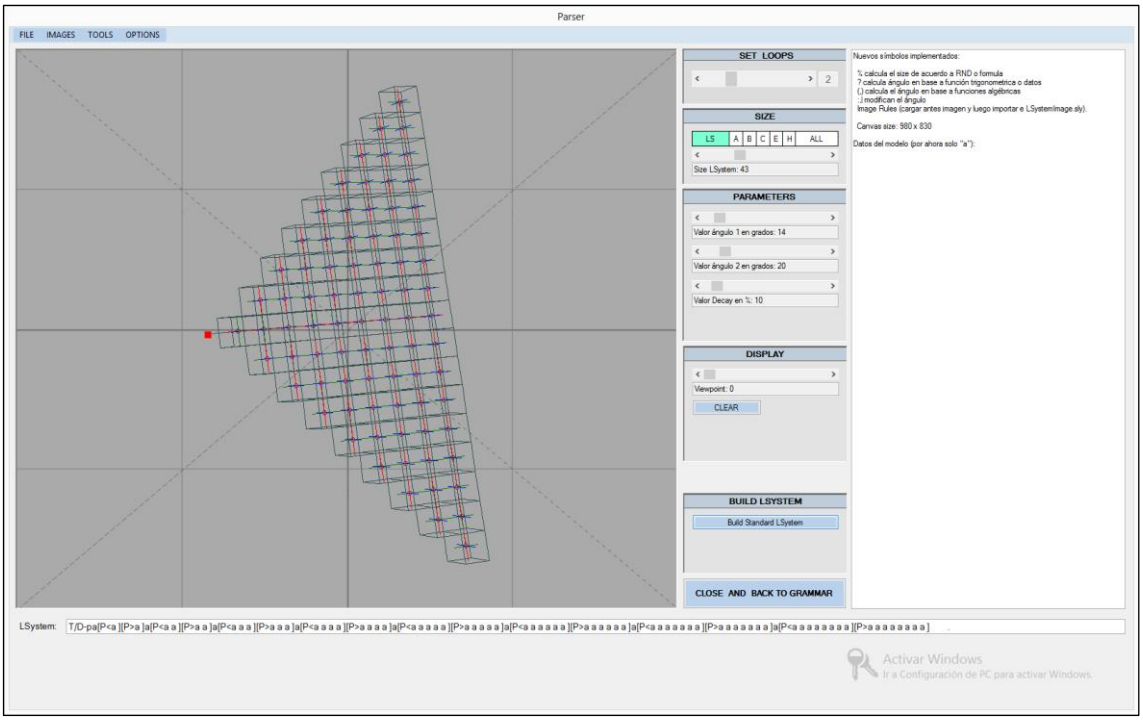


Fig. 33. Programación de “n” y bloqueo para hacer la estructura simétrica.

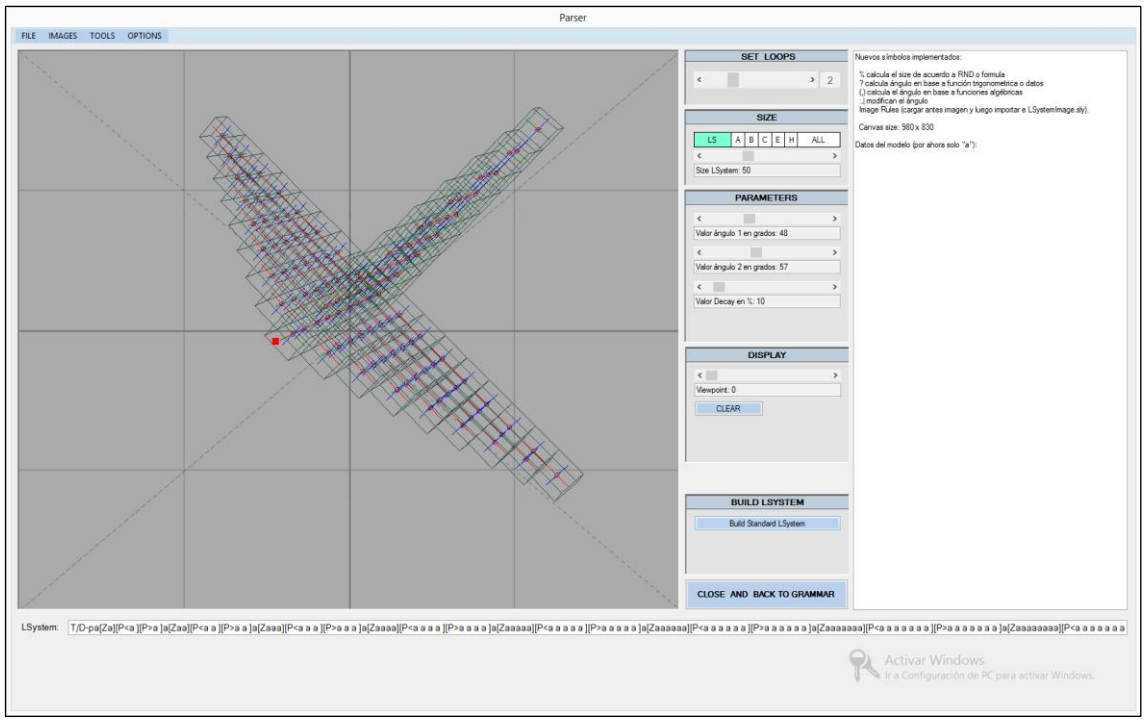


Fig. 34. Programación de “n”, bloqueo para hacer la estructura simétrica y “n1” para la parte vertical.

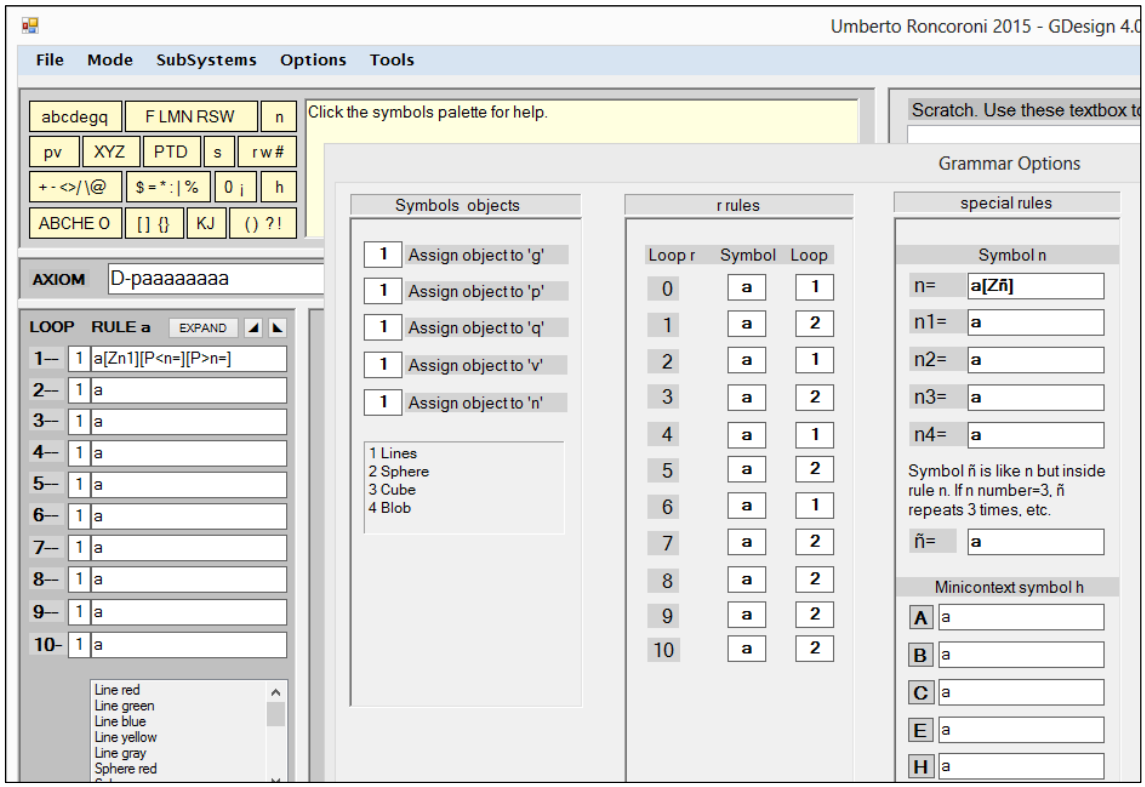


Fig. 35. Programación de “n”, “n1”, “n2” hasta 4 y símbolo “ñ”.

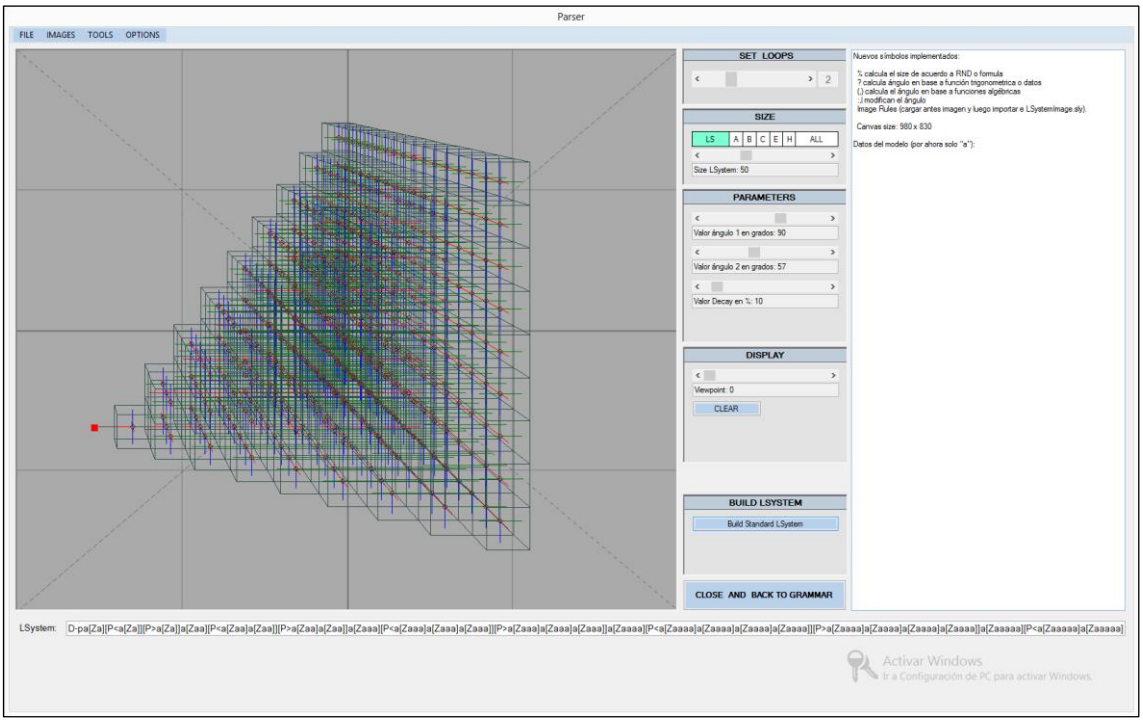


Fig. 36. Para completar es necesario utilizar “n”, “n1”, “=” y “ñ”.

Con este sistema, utilizando varias reglas de “n”, el bloqueo con “=” y el bucle interno a “n” con “ñ”, es posible hacer estructuras basadas en bucles.

2.3 Símbolos aleatorios y procesos estocásticos

Un elemento esencial de los procesos generativos (de los cuales los sistemas de sustitución recursiva son parte) es el azar. El punto es que los procesos computacionales son deterministas, por lo tanto lo impredecible y emergente que caracteriza lo auténticamente generativo es en realidad un artificio que se debe al azar. Por lo tanto la gestión de la aleatoriedad es una tarea imprescindible para controlar estos sistemas.

2.3.1 Reglas estocásticas

Usualmente se resuelve con las reglas estocásticas, es decir se decide en base a un generador de números aleatorios cual regla se escoge. Obviamente se puede escoger entre 2, 3 o más reglas.

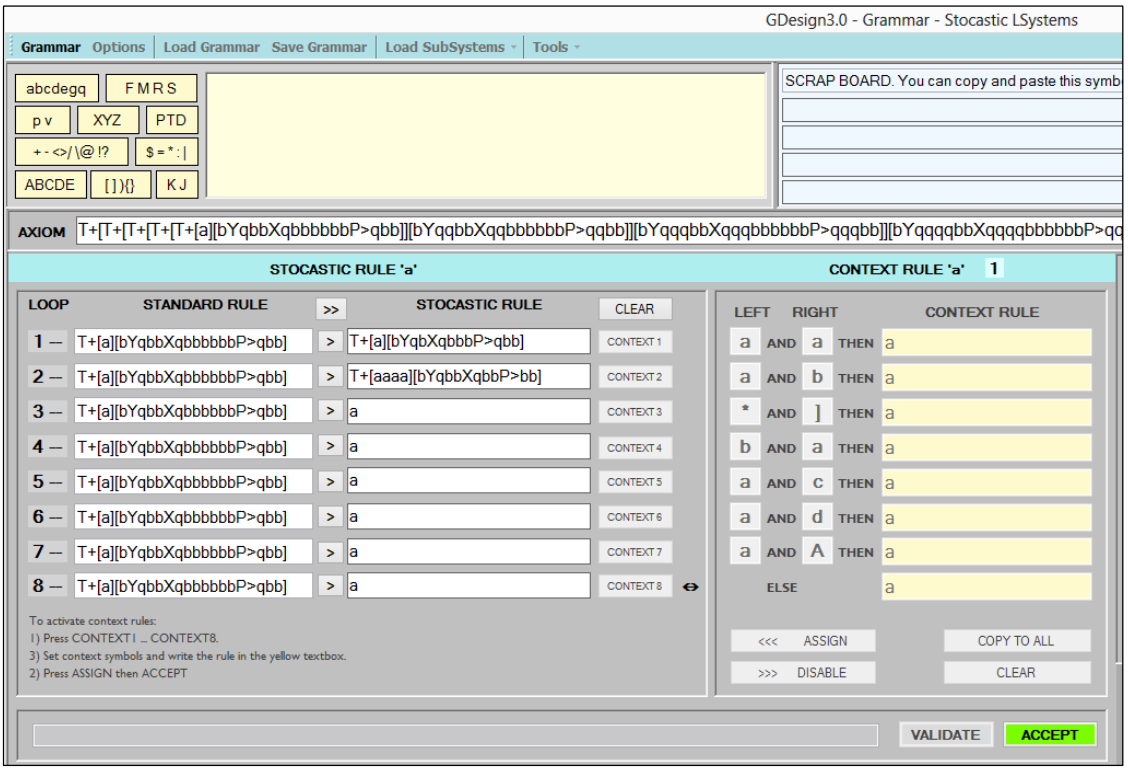


Fig. 37. Ejemplo de gramática con reglas estocásticas.

Es importante el diseño de la interfaz, pues la gestión de las reglas puede ser tediosa. En GDesign 3.0 se provee la posibilidad de copiar y editar las reglas con facilidad.

2.3.2 Funciones aleatorias. Símbolos “R”, “S”, “W” y “=”

Otra posibilidad es implementar funciones. Las funciones son símbolos cuyas reglas se determinan en base a procesos más complejos que la simple sustitución.

Los símbolos “R” y “S” permiten generar una secuencia de símbolos al azar dentro un conjunto que el programador puede establecer (utilizando obviamente los símbolos del lenguaje). Es posible escoger entre 6 símbolos diferentes, por ejemplo:

“a,a,a,b,b,b” resulta en una cadena donde “a” y “b” tiene el 50% de probabilidad

“a,b,b,c,c,c” resulta en una cadena donde “a” tiene el 20% de probabilidad, “b” el 30% y “c” el 50% de probabilidad.

El usuario puede fijar la longitud de la cadena que quiere generar, la posibilidad interesante es que la longitud puede ser también aleatoria. Y el usuario puede escoger

si activar solo la longitud aleatoria, por lo cual usa el mismo símbolo en los 6 casos posibles, solo los símbolos aleatorios o ambos.

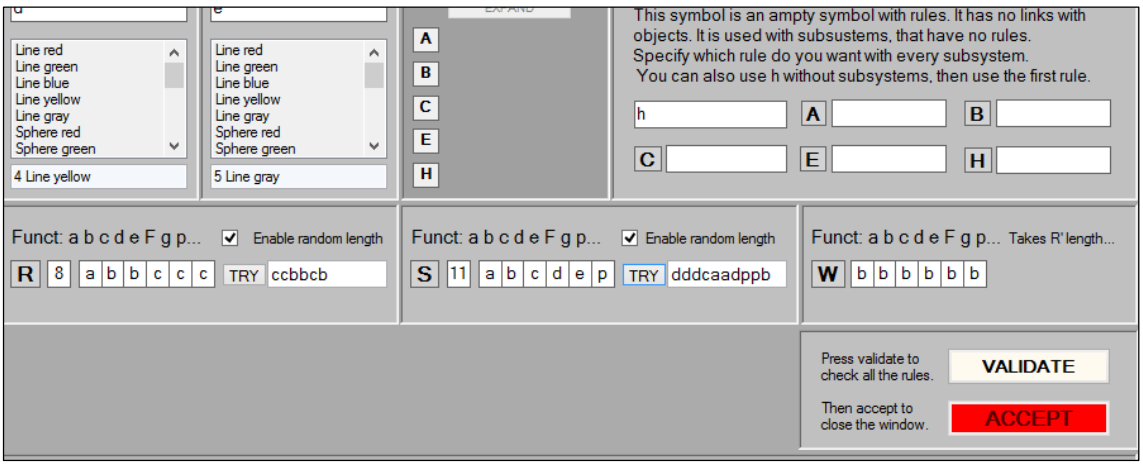


Fig. 38. Controles para utilizar las funciones en GDesign 3.0.

La longitud puede ser parametrizada de acuerdo al contexto (distancia de la raíz, edad, etc.).

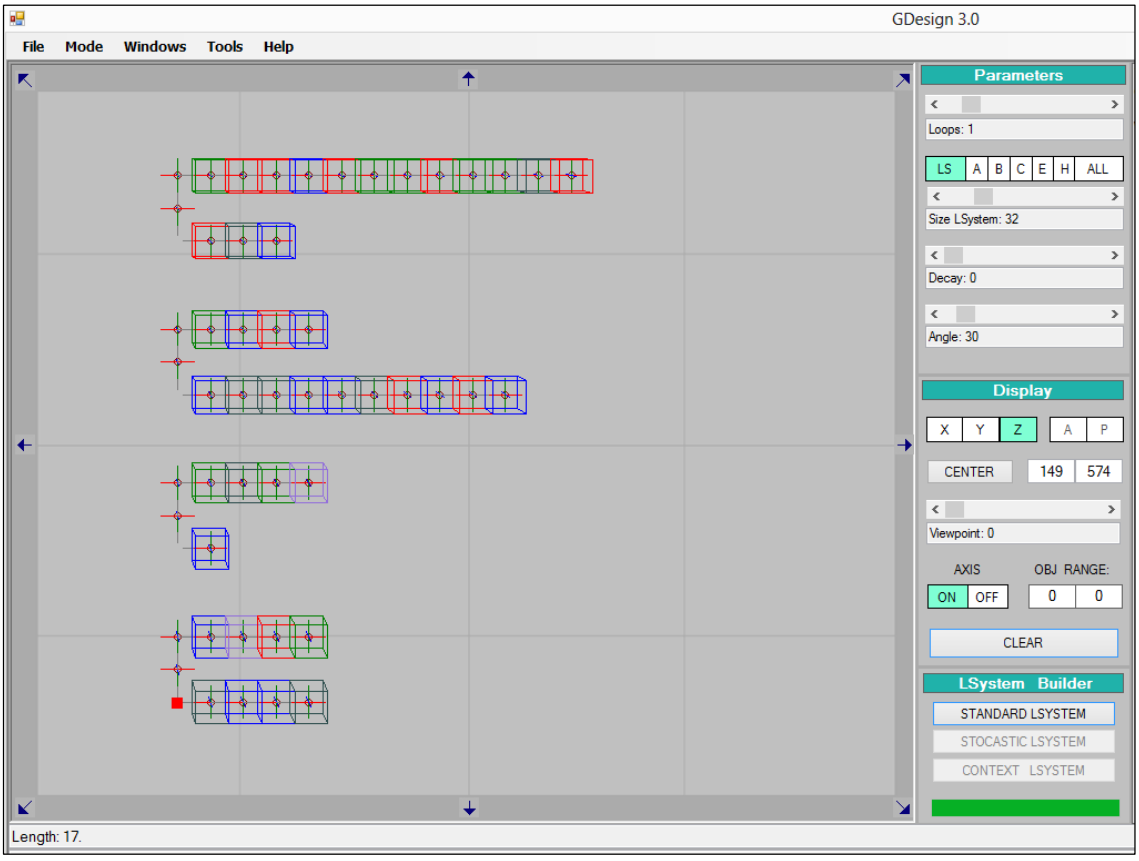


Fig. 39. Grupos de 2 funciones aleatorias “R” la superior y “S” la inferior de cada dupla.

La secuencia aleatoria se generar cada que vez que se encuentra “R” o “S” en la cadena que se está procesando. Esto conlleva lagunas cuestiones adicionales.

Por ejemplo, ¿cómo se hace a generar una superficie formada por 10 elementos y que cada uno tenga la misma longitud aleatoria? Normalmente no sería posible pues el valor aleatorio es diferente para cada elemento, como se ve en la imagen 21 y 22.

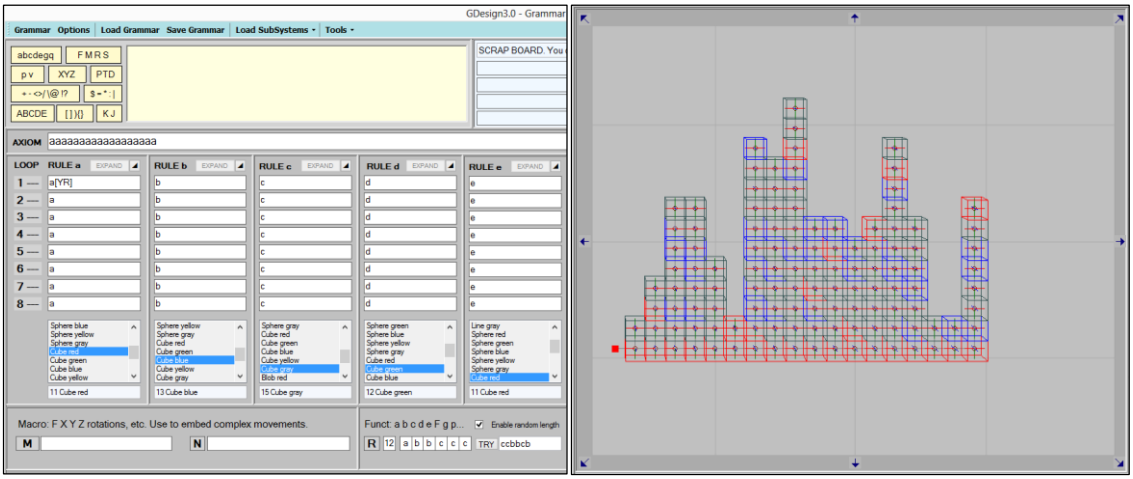


Fig. 40. Ejemplo del uso de funciones. Para cada símbolo “a” en rojo “R” genera una secuencia diferente

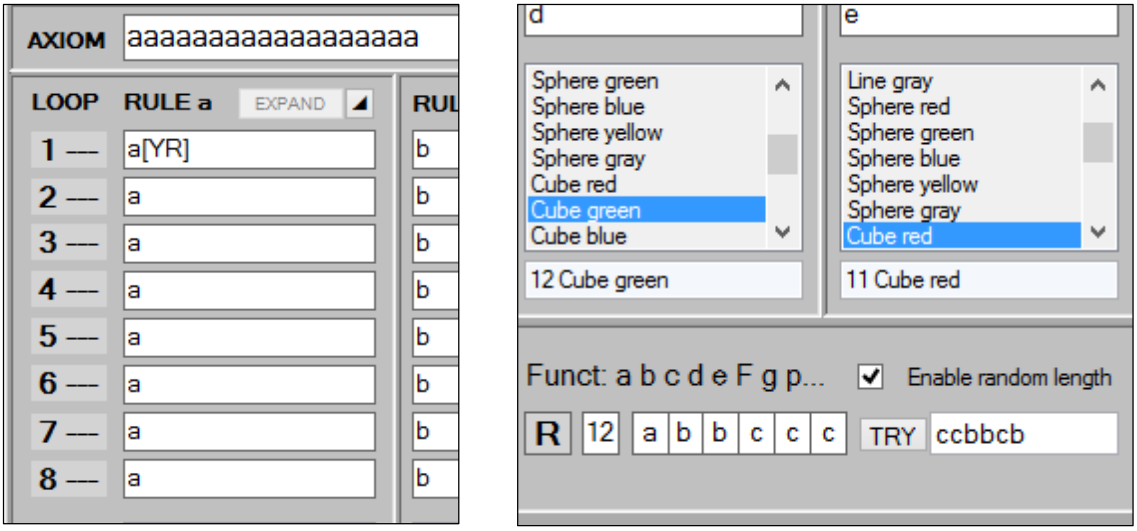


Fig. 41. Detalle de las reglas

La solución adoptada es utilizar el símbolo “=”. Puesto después de “R” o “S” copia la secuencia aleatoria del “R” o “S” anterior. Para generar un rectángulo aleatorio, por ejemplo, comenzamos con el axioma “abbbbbbb”. Luego:

1. “a” → ”[YR]” “b” → ”b” “R” → nueva cadena aleatoria ”aabcbba”
2. “a” → ”a” “b” → ”[YR=]” “R” → misma cadena ”aabcbba”

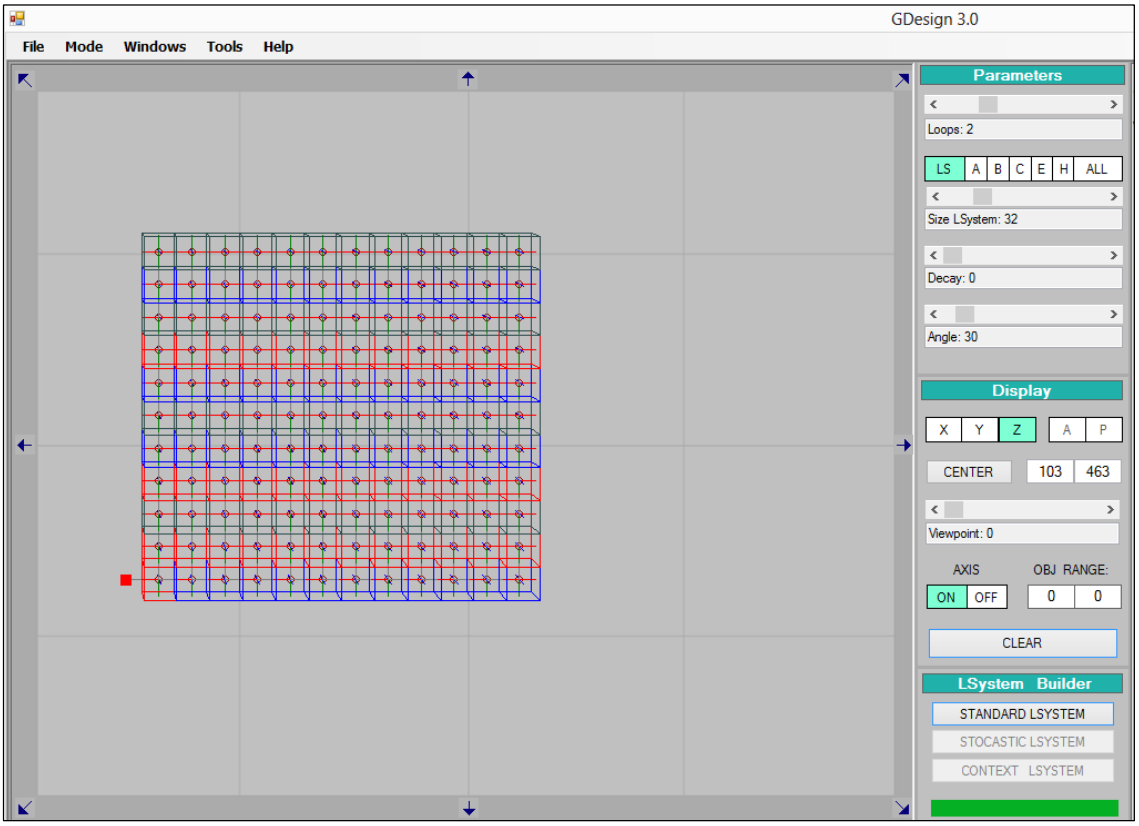


Fig. 42. Paso 1 y 2

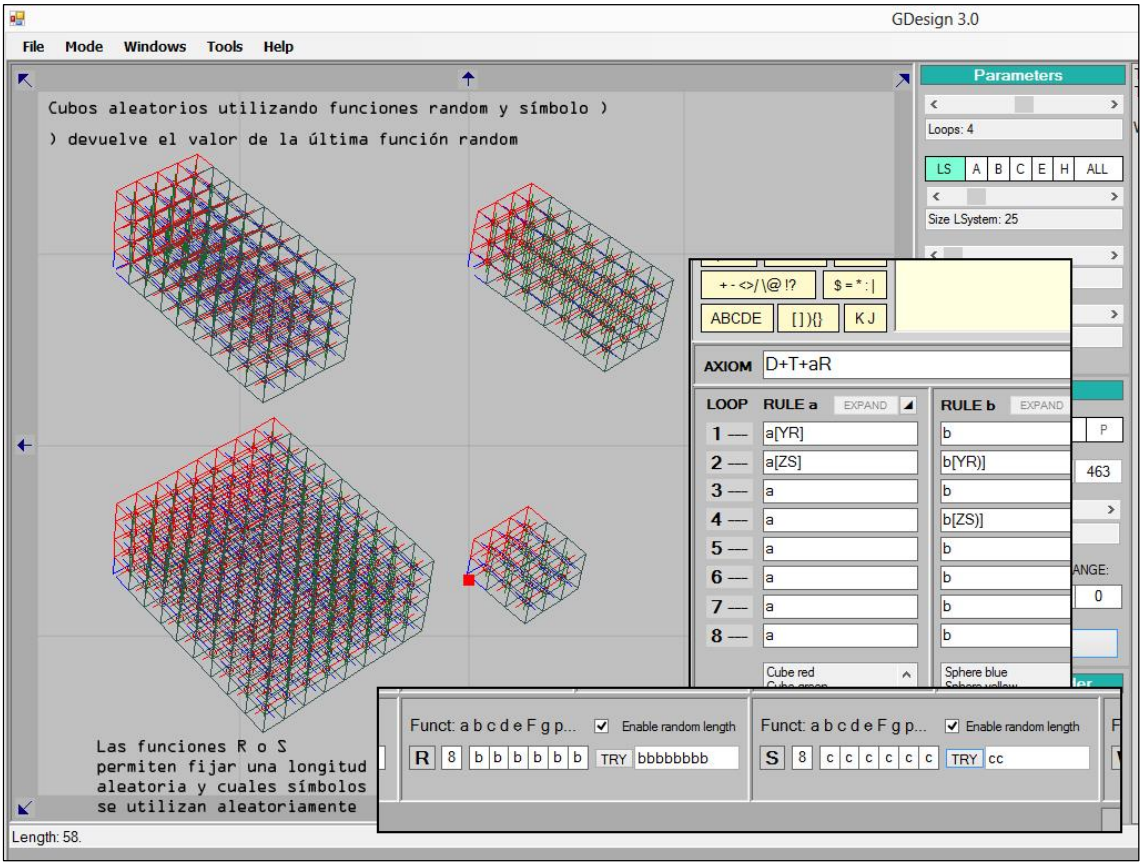
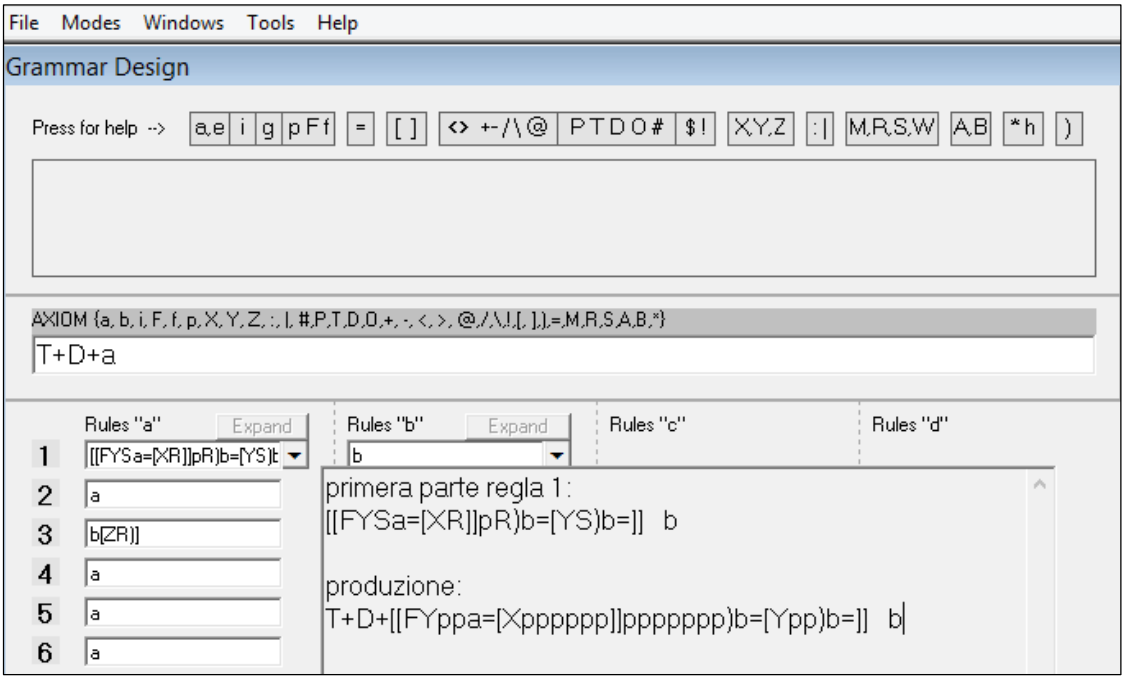


Fig. 43. Para hacer un cubo aleatorio necesitamos “R” y “S”. En GDesign 3.0 se utiliza “)” en lugar de “=”

Para hacer un cubo aleatorio con columnas y vigas se requiere una programación más compleja. En la regla 1 creamos un rectángulo aleatorio, “R” da la longitud aleatoria y “S” la altura aleatoria, todo el rectángulo está dentro una rama.



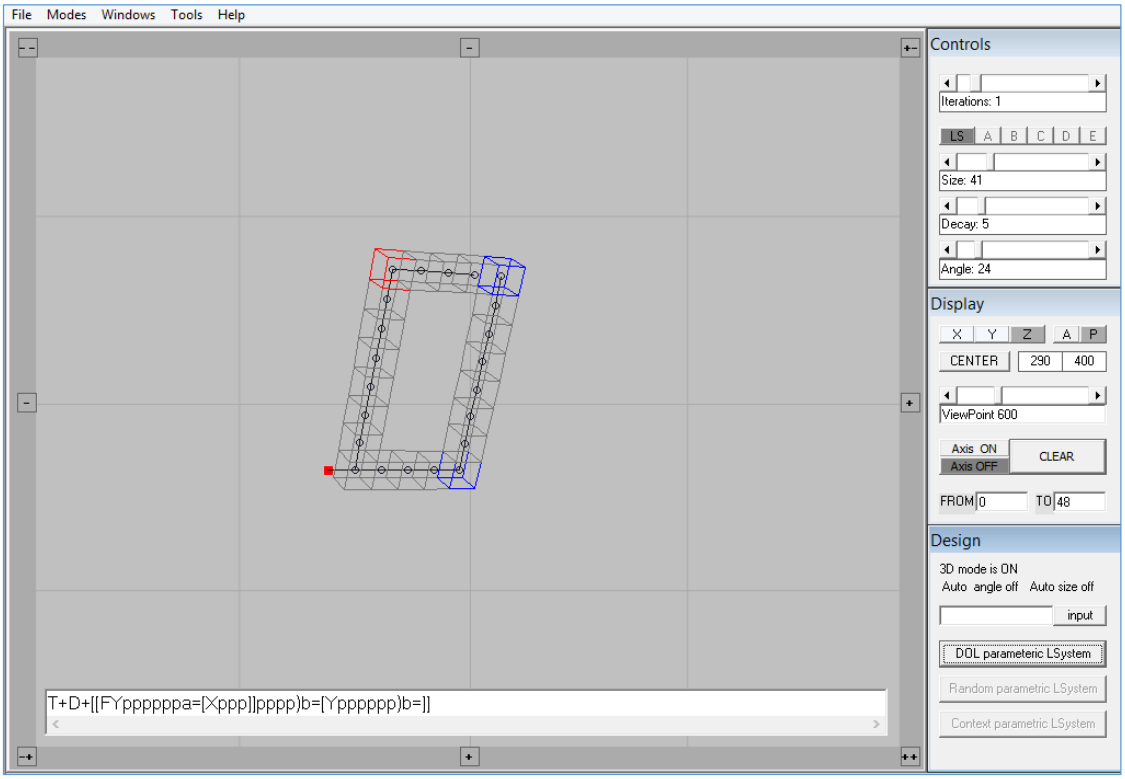
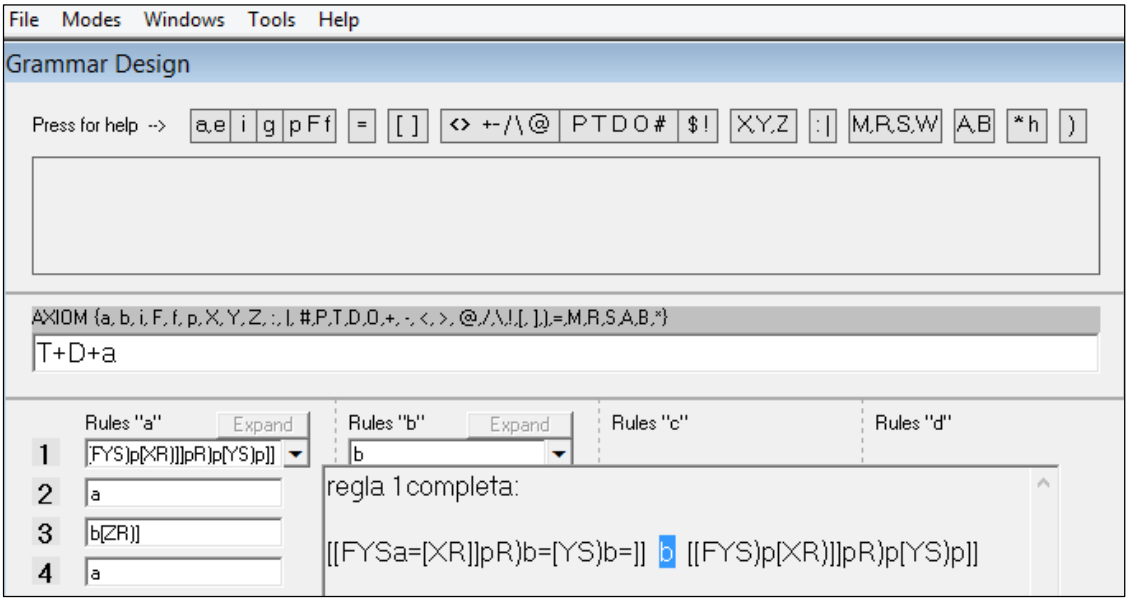


Fig. 44-45. Construcción del primer lado del andamio aleatorio con la primera parte de la regla 1 de “a”

La segunda parte de la regla sirve para generar un rectángulo igual distanciado del primero de una distancia aleatoria. En esta etapa el rectángulo está todavía pegado al primero, pero se moverá después dando una regla al símbolo “b”



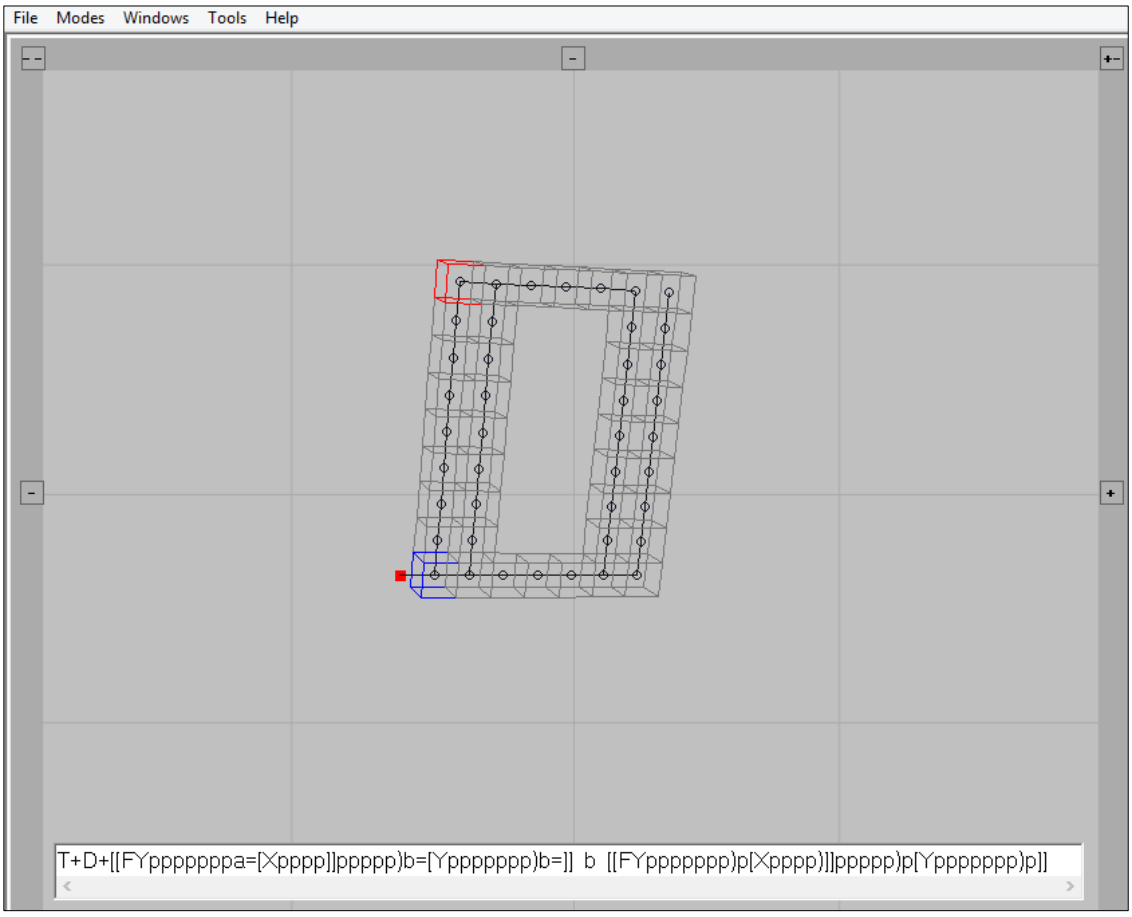


Fig. 46-47. La 1 regla completa con el segundo lado todavía pegado al primero

El segundo rectángulo es igual pues se usan “R” y “S” del primer rectángulo, utilizando “=” o “)” en GDesign 3.0 (como este caso).

Ahora el símbolo “b” se sustituye con una viga de longitud aleatoria, lo que separa los dos rectángulos. Nótese (aunque no se ven porque están tapados) que hay un “a” (rojo) y otros “b” (azul) que están bloqueados. Servirán para crear las vigas faltantes en base al valor aleatorio del símbolo “b” subrayado en verde.

[[FYpppppppa=[Xpppp]]ppppp)b=[Ypppppppb=]]b[[FYpppppppp)p[Xpppp)]]ppppp)p[Ypppppppp)p]]

En la siguiente imagen se nota el “b” (subrayado verde) sustituido por una viga de longitud aleatoria.

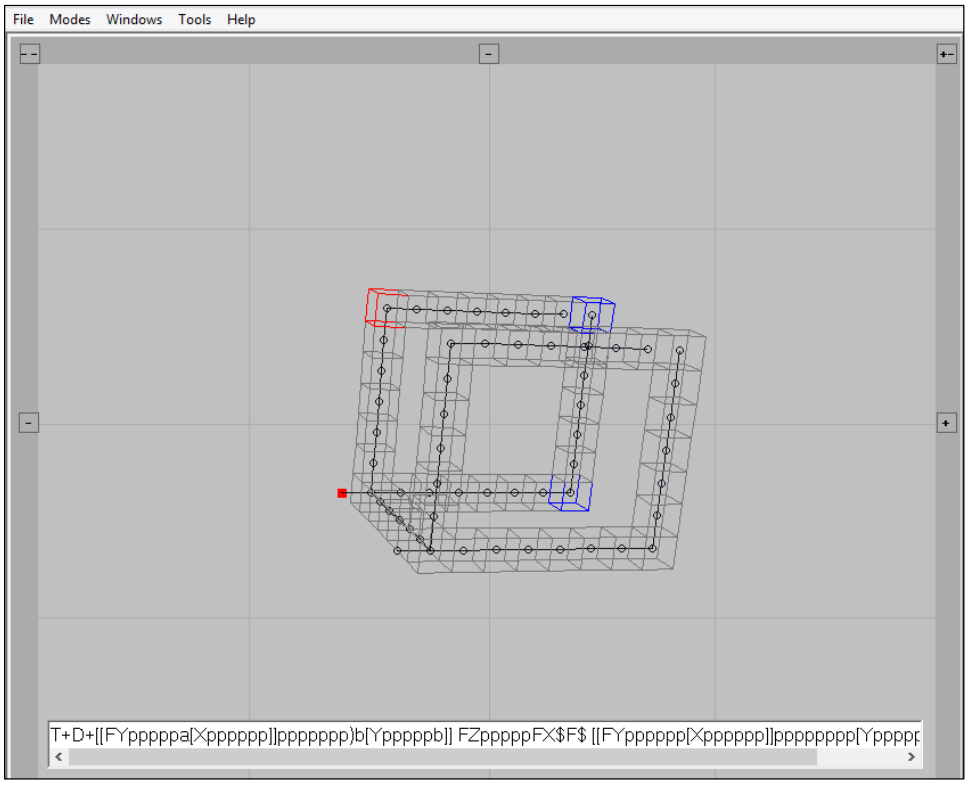
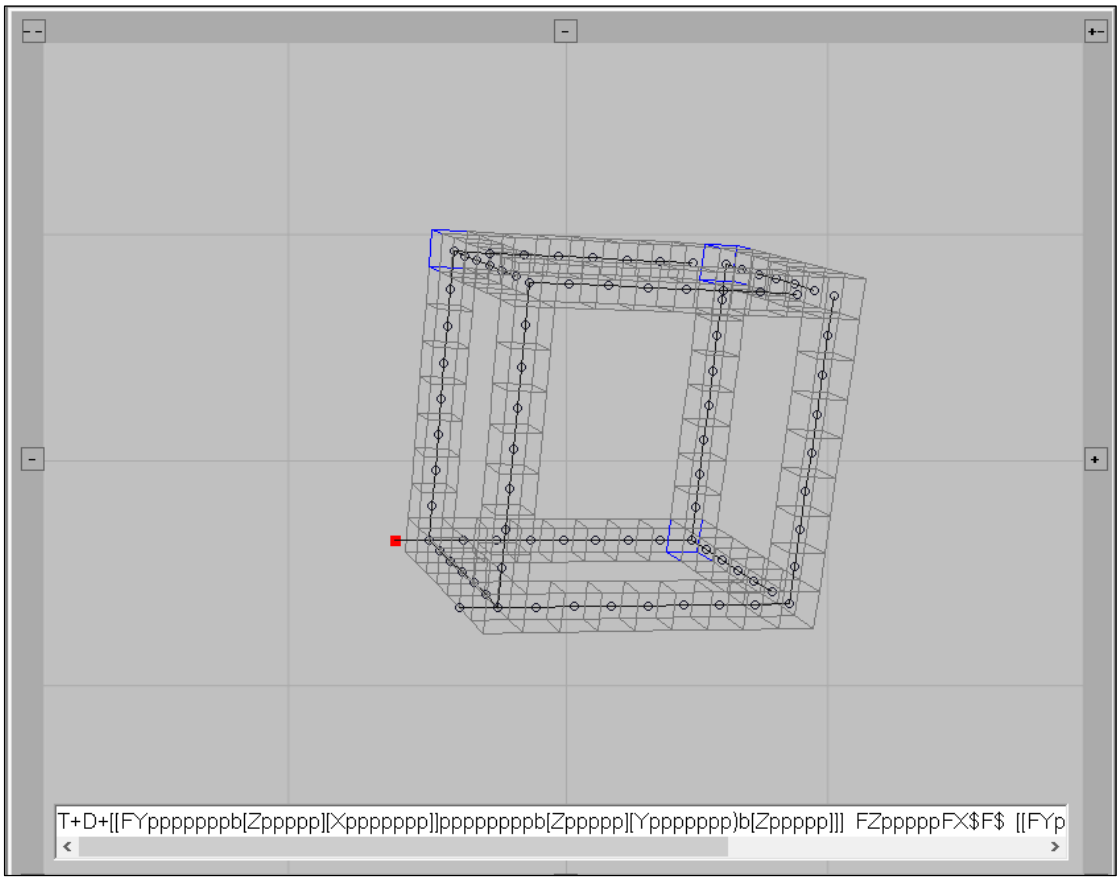


Fig. 48. El segundo rectángulo posicionado por la producción de “b”

Falta ahora generar las vigas para el “a” y los “b” que estaban bloqueados en la regla anterior.



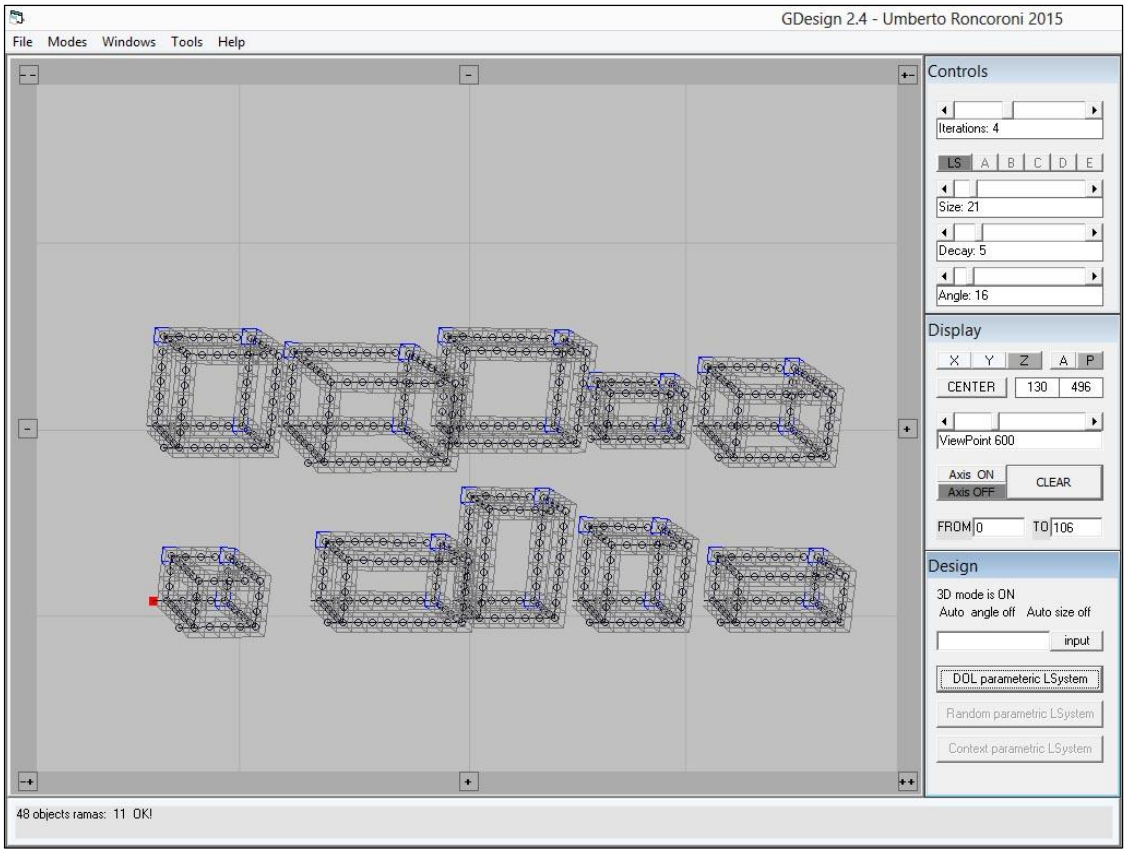


Fig. 49-50. El andamio completo y ejemplos.

2.3.2 Símbolos aleatorios. Símbolos “@”, “%”

Se pueden utilizar símbolos para determinar valores aleatorios del ángulo de rotación y del tamaño de los objetos. “@” permite establecer un ángulo aleatorio entre un mínimo y un máximo que se establece en la ventana Options de la ventana principal.

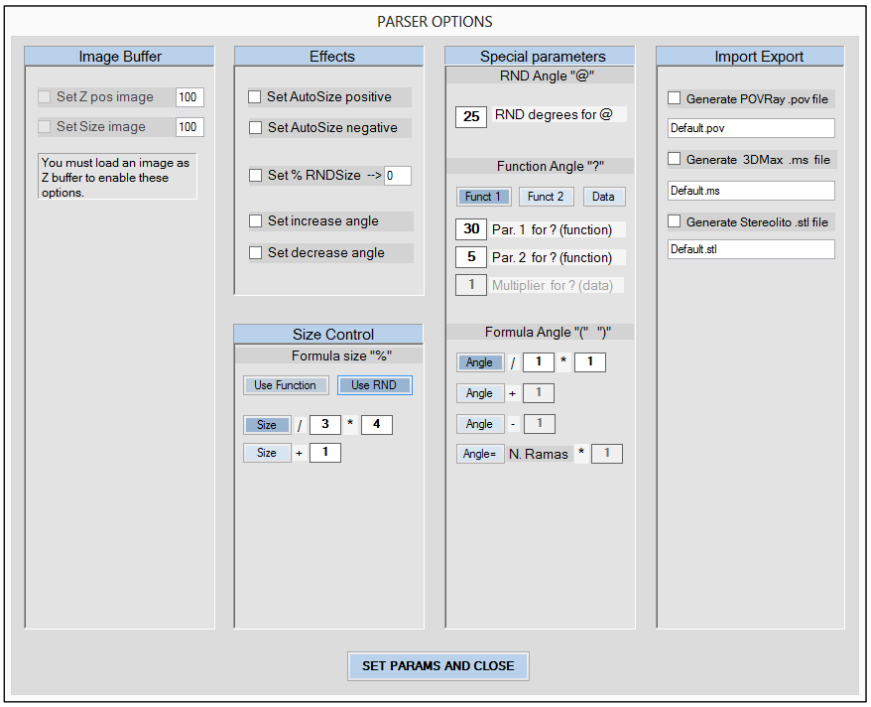


Fig. 51. Ventana Options.

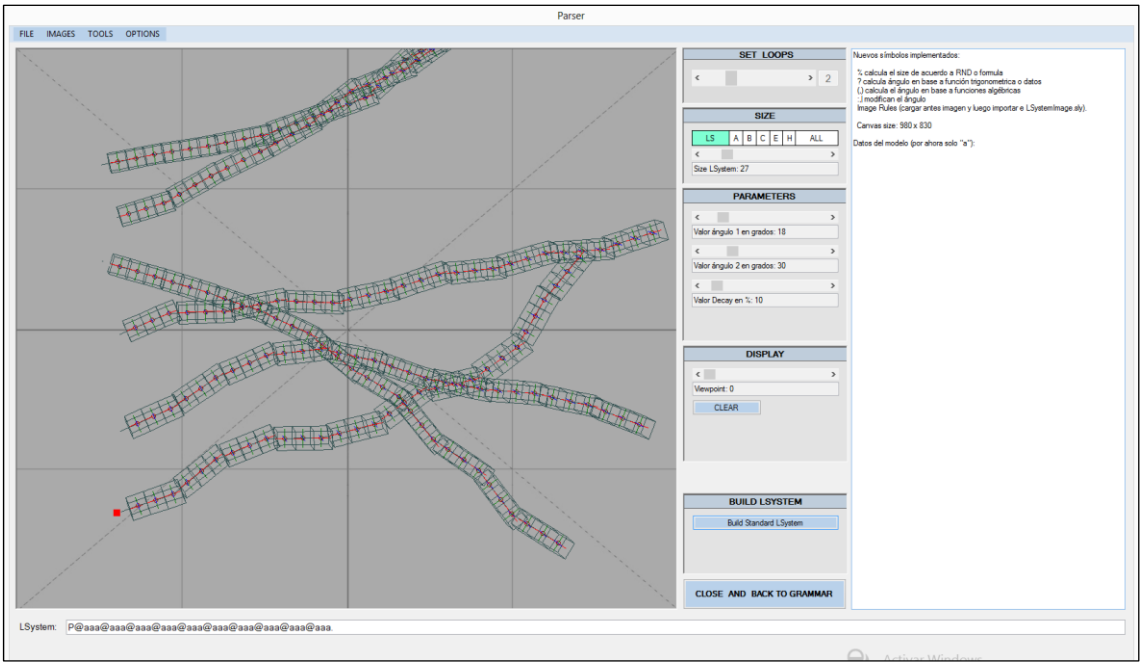


Fig. 52. Secuencias con ángulos aleatorios

"%" si es puesto como valor *random* permite establecer un tamaño aleatorio entre un mínimo y un máximo que depende del parámetro Size.

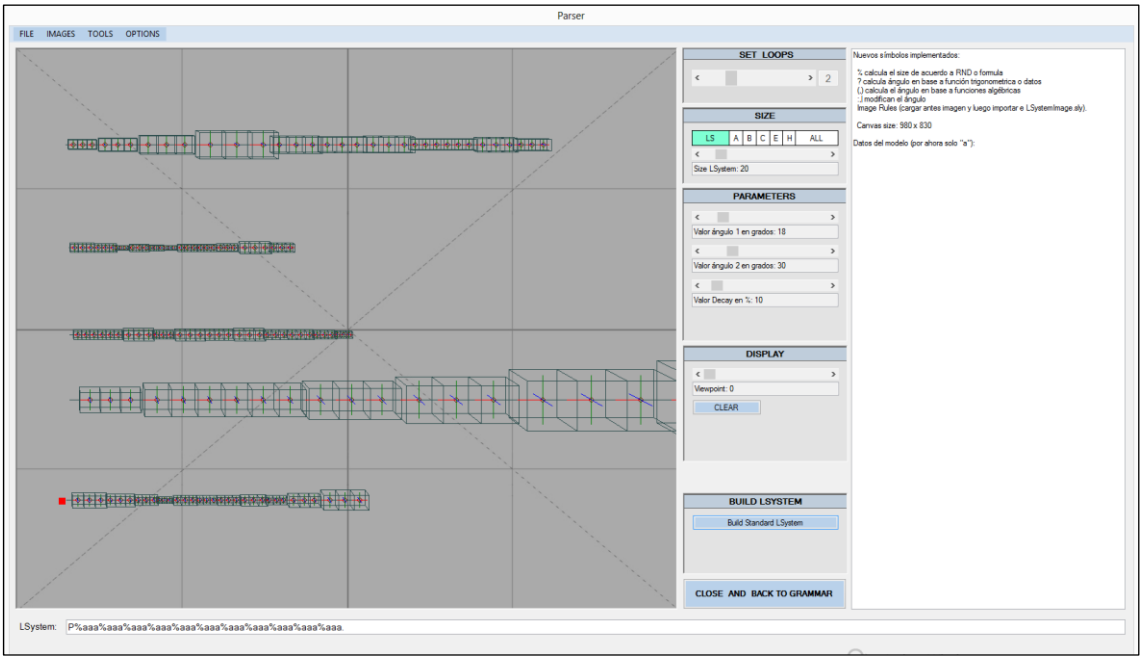


Fig. 53. Secuencias con dimensiones aleatorias

Obviamente se pueden combinar ambos efectos.

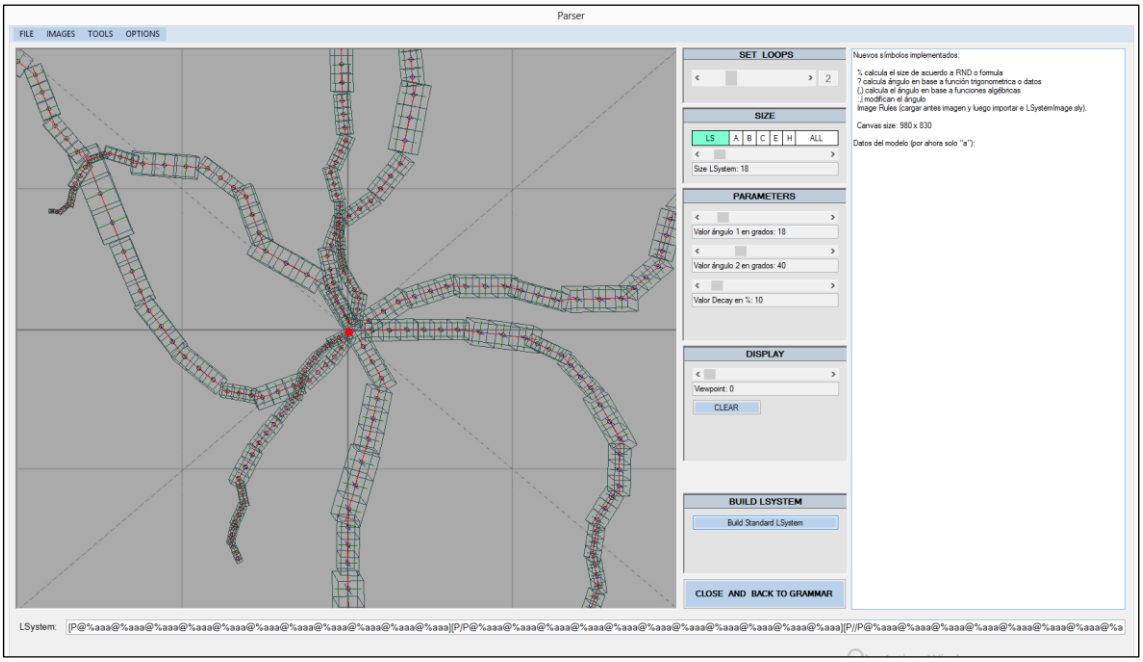


Fig. 54. Dimensiones y ángulos aleatorios

Programación de símbolos "@" y "%"

El valor aleatorio del ángulo y del tamaño se puede establecer:

- en base a la distancia de la raíz (que es un campo del objeto)
- en base a un símbolo antecesor (Previous)
- en base a un sucesor (Next) (estos ya se conocen en el proceso de sustitución, pues siempre se detectan para cada símbolo)

Pueden también tener efectos automáticos, como aumentar o disminuir según la jerarquía de la rama en la cual están (cómo se hace con los efectos automáticos para el ángulo y el tamaño que se establecen en la ventana Options).

Utilizando el símbolo "=" guardan el último valor aleatorio utilizado "@=", o "%=".

2.4 Fórmulas y datos

A veces es necesario controlar una secuencia de rotaciones en modo determinado. Para esto se pueden utilizar símbolos que permiten las rotaciones paramétricas, utilizando formulas trigonométricas o datos guardados en un archivo ASCII.

Para esto se utiliza el símbolo "?". Para cada "?" se recalcula la formula y se incrementan sus parámetros. El símbolo "¿" reinicializa los parámetros para recomenzar la formula con sus valores iniciales.

Se puede escoger si "?" utiliza fórmulas o datos, en este último caso el programador puede cargar un archivo precedentemente creado.

Esto permite controlar las formas complejas en un modo más creativo y eficiente. Además las variables de las fórmulas pueden ser parametrizadas de acuerdo al contexto (edad, distancia de la raíz, etc.).

2.4.1 Símbolo “?” con fórmulas

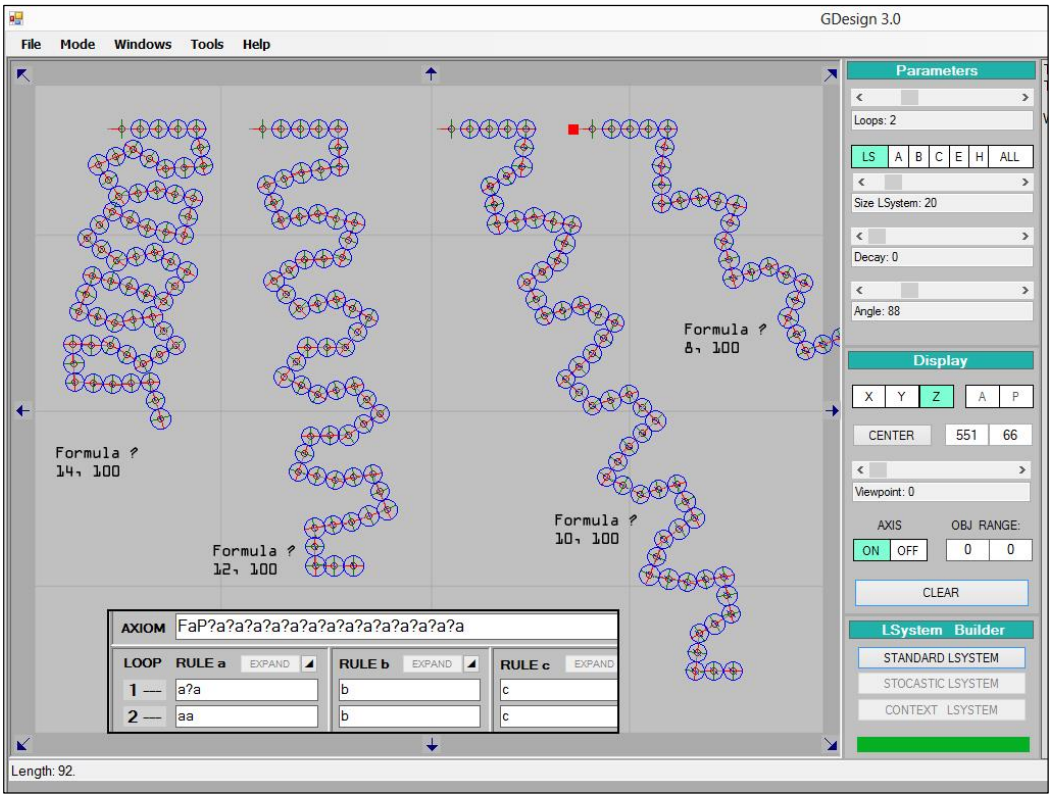


Fig: 55. Rotaciones con fórmulas con diferentes parámetros

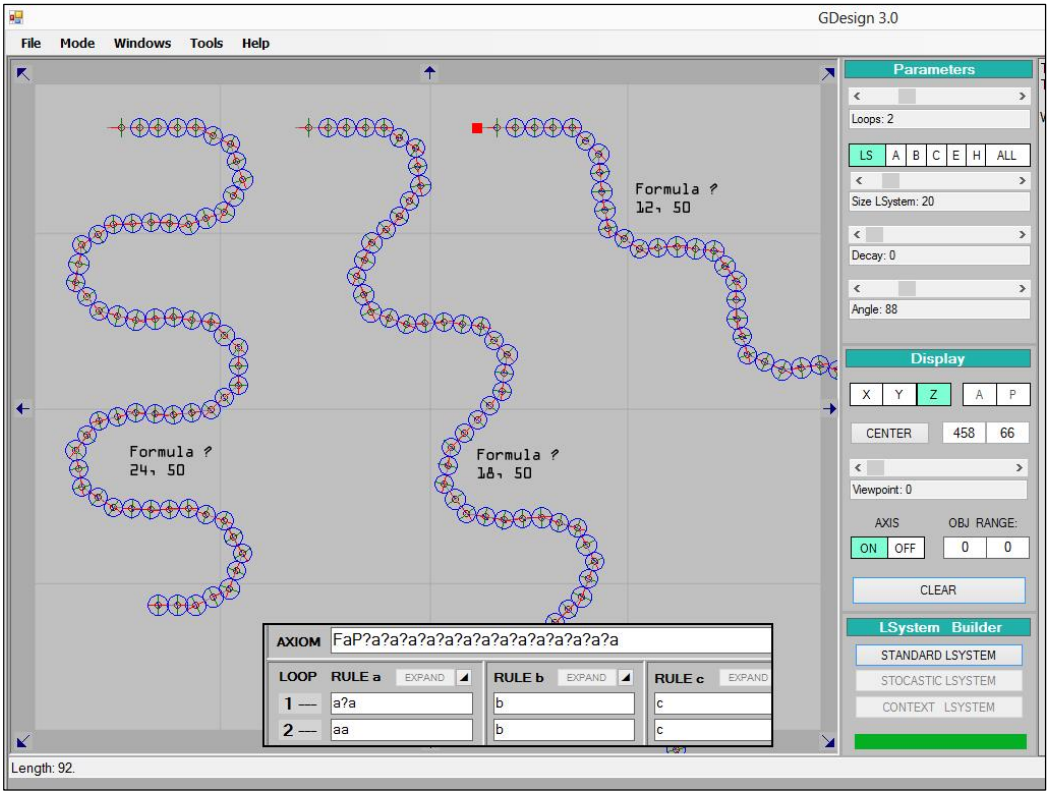


Fig: 56. Rotaciones con fórmulas con diferentes parámetros

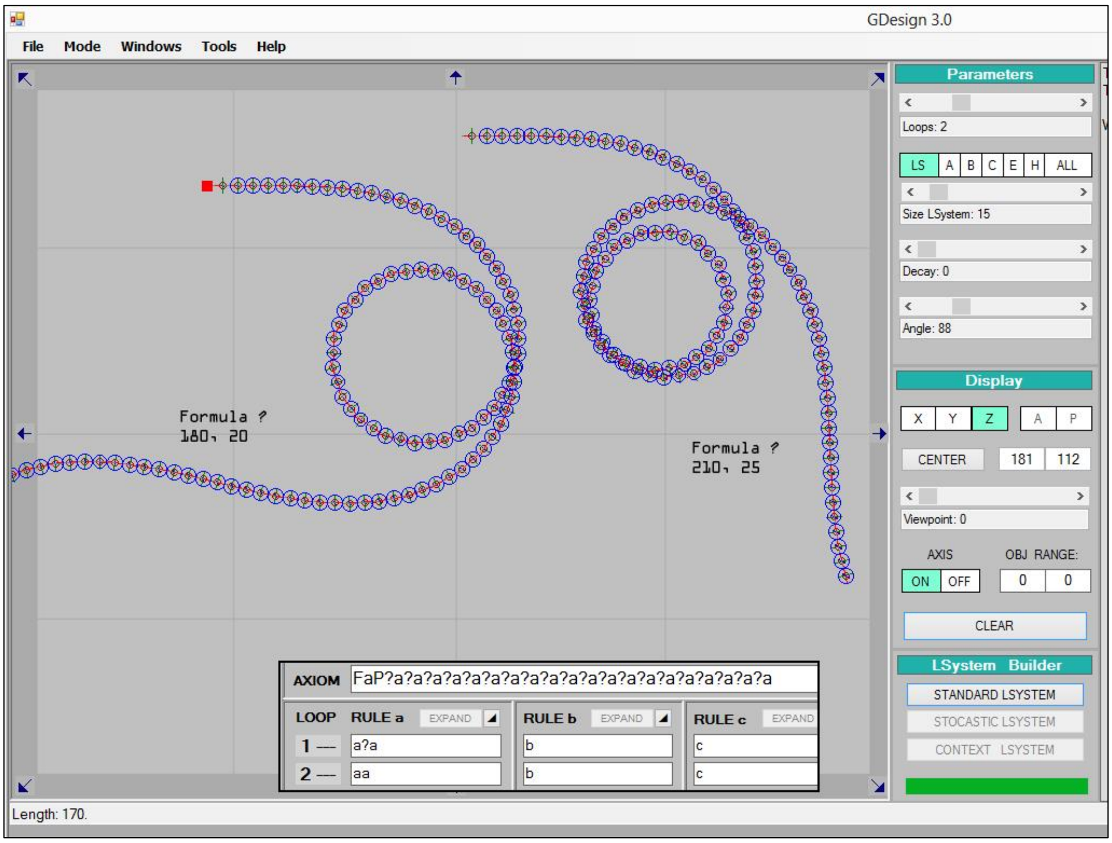


Fig: 67. Rotaciones con fórmulas con diferentes parámetros

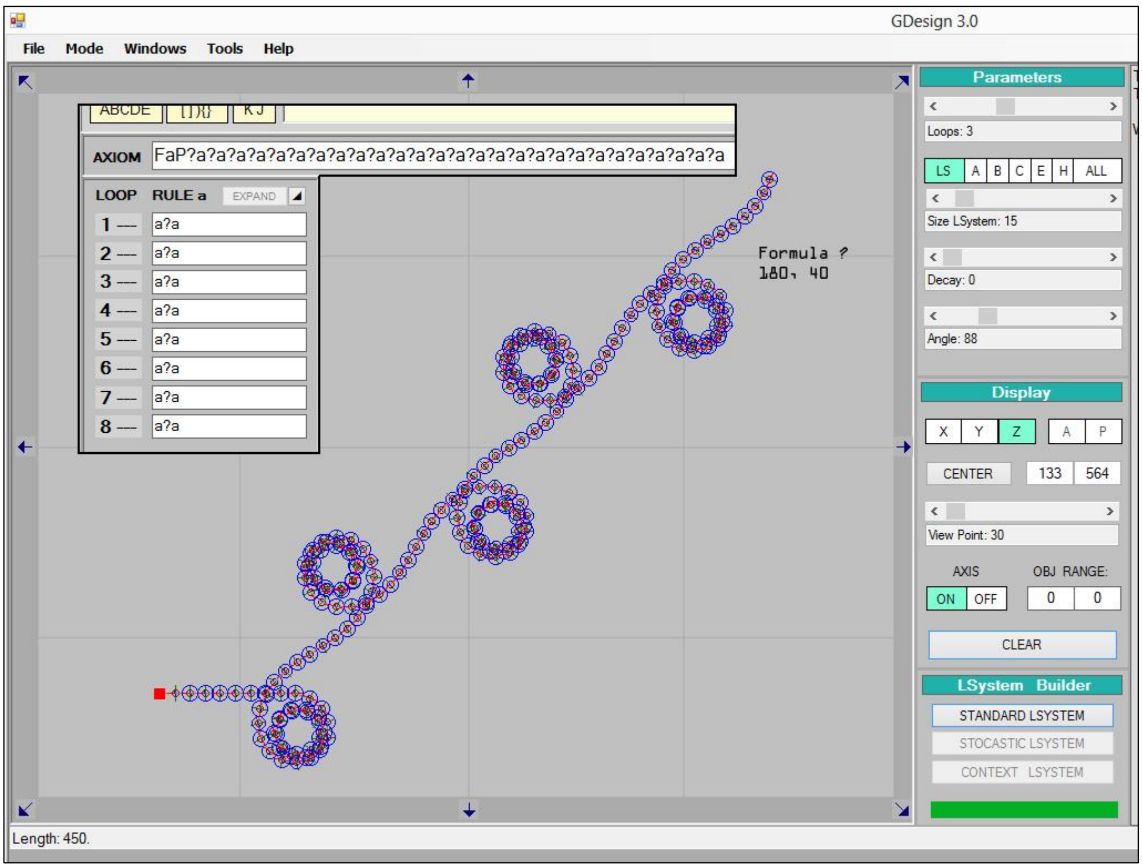


Fig: 58. Rotaciones con fórmulas con diferentes parámetros

2.4.2 Símbolo “?” con datos

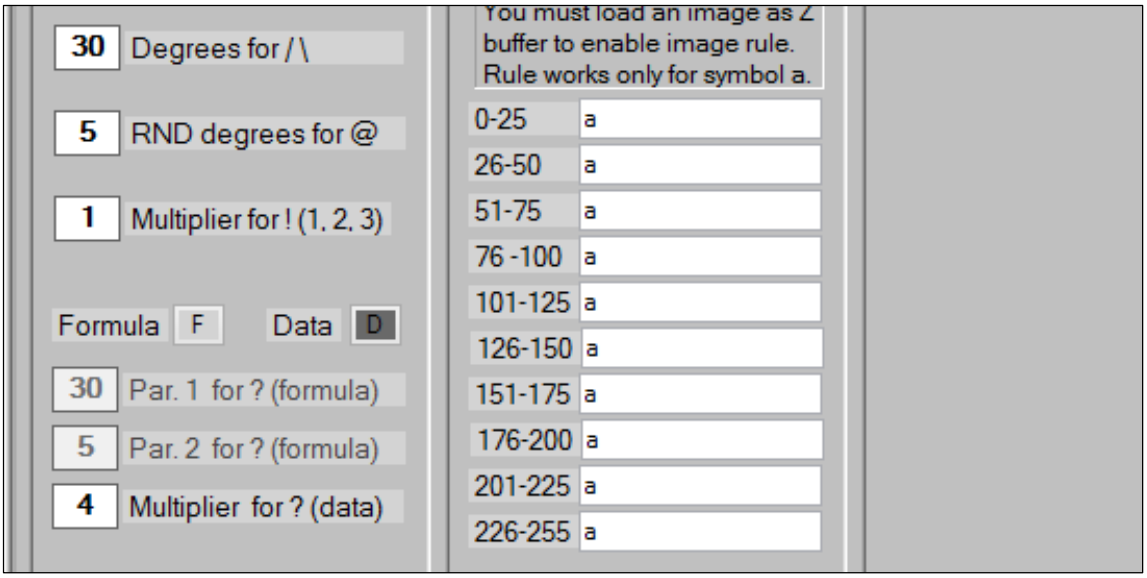


Fig. 59. El menú de la fórmula para “?”

Los datos se guardan en un *array* o se cargan desde un archivo ASCII:

```
double[] DataAngle = {-15, 0, 15, 0, -15, 5, -5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

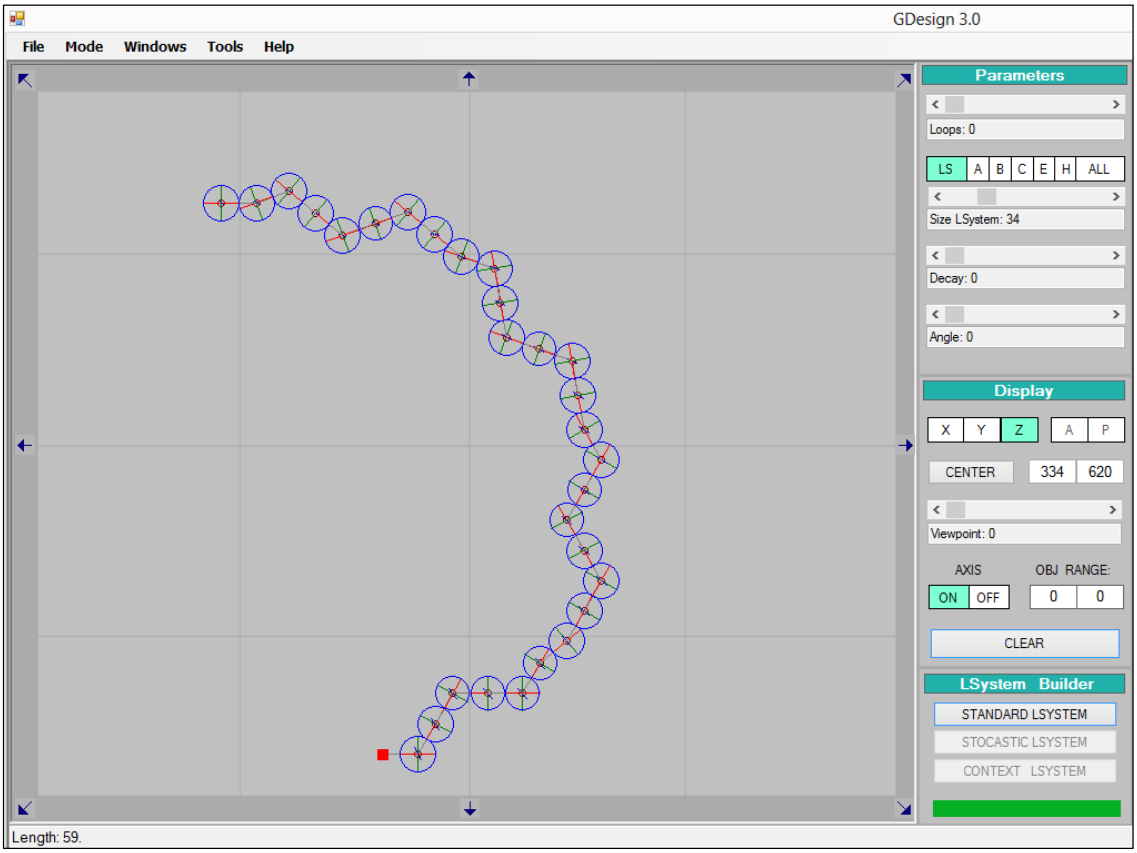


Fig. 60. Secuencia de rotaciones con datos

La ventaja es que se pueden sacar los datos interpretando otros datos o que se pueden recibir a través de sensores o similares.

2.4.3 Símbolo “w” con datos

Este símbolo produce una cadena de símbolos tantas veces cuantos son los valores contenidos en un archivo de datos “data.txt” en formato ASCII. Ejemplo:

```
6   Número de datos contenidos en el archivo
    Empty
7   data[0]
12  data[1]
2   data[2]
5   data[3]
8   data[4]
19  data[5]
```

El arreglo contiene hasta 2500 datos, que se inicializan con valor 1.

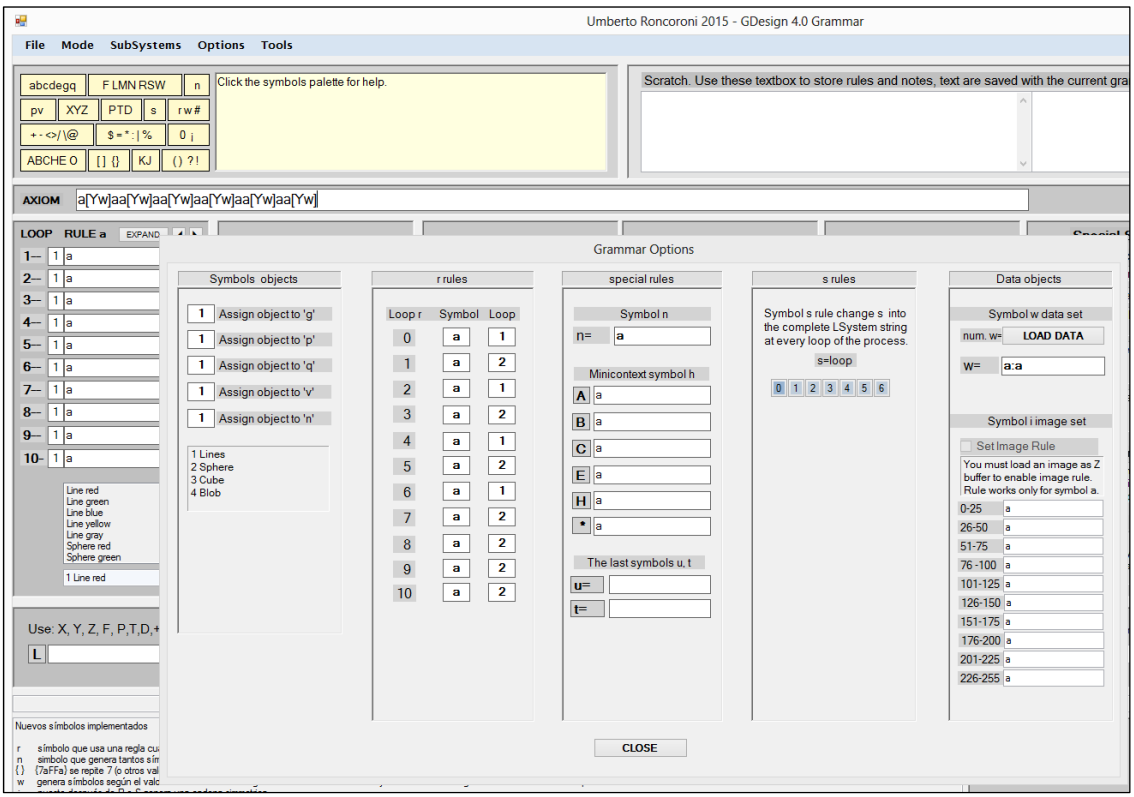


Fig. 61. Secuencia de rotaciones con datos

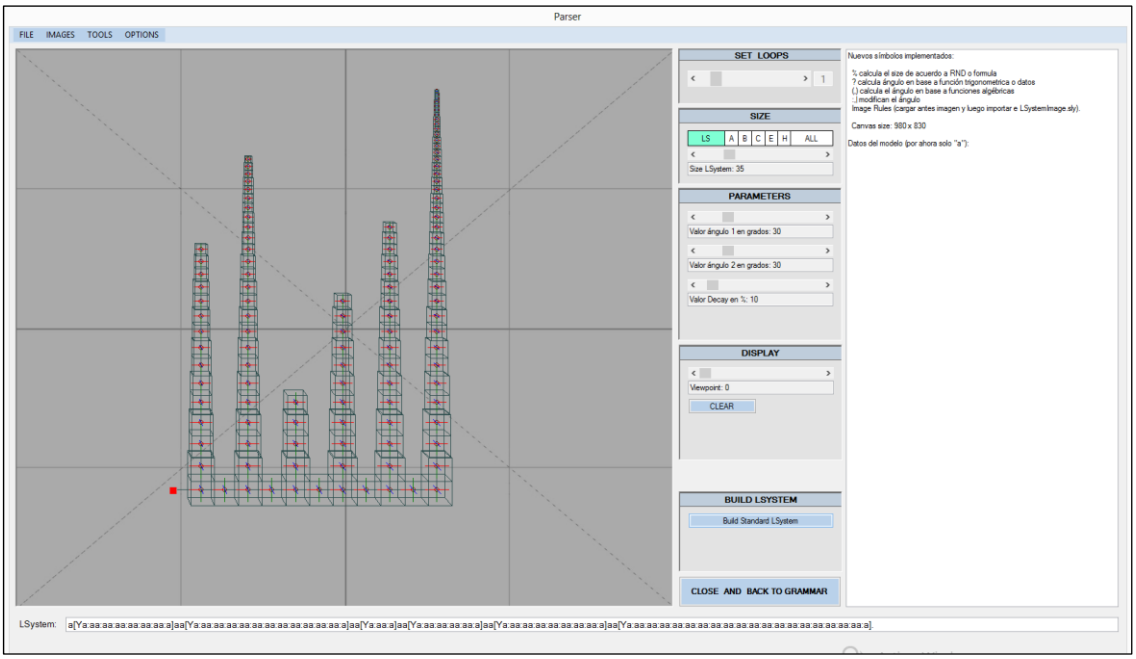


Fig: 62. Rotaciones con fórmulas con diferentes parámetros

La cadena que se produce “a[Yw]aa[Yw]aa[Yw]aa[Yw]aa[Yw]aa[Yw]”: “a[Ya:aa:aa:aa:aa:aa:aa:a]aa[Ya:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:a]aa[Ya:aa:a]aa[Ya:aa:aa:aa:aa:a]aa[Ya:aa:aa:aa:aa:aa:aa:aa:a]aa[Ya:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:a]”.

El símbolo “w” puede ser reseteado utilizando el símbolo “0” como en el caso de “n” y puede conservar el último dato utilizando “=”.

Programación de “w”

“w” se puede programar:

- utilizando “0” para resetear el índice
- con “=” para utilizar un mismo dato
- también se pueden utilizar varios archivos de datos, por ejemplo “w1”, “w2” … “w9” indica el archivo de datos data1.txt, data2.txt … data9.txt
- el valor del dato puede depender de un parámetro

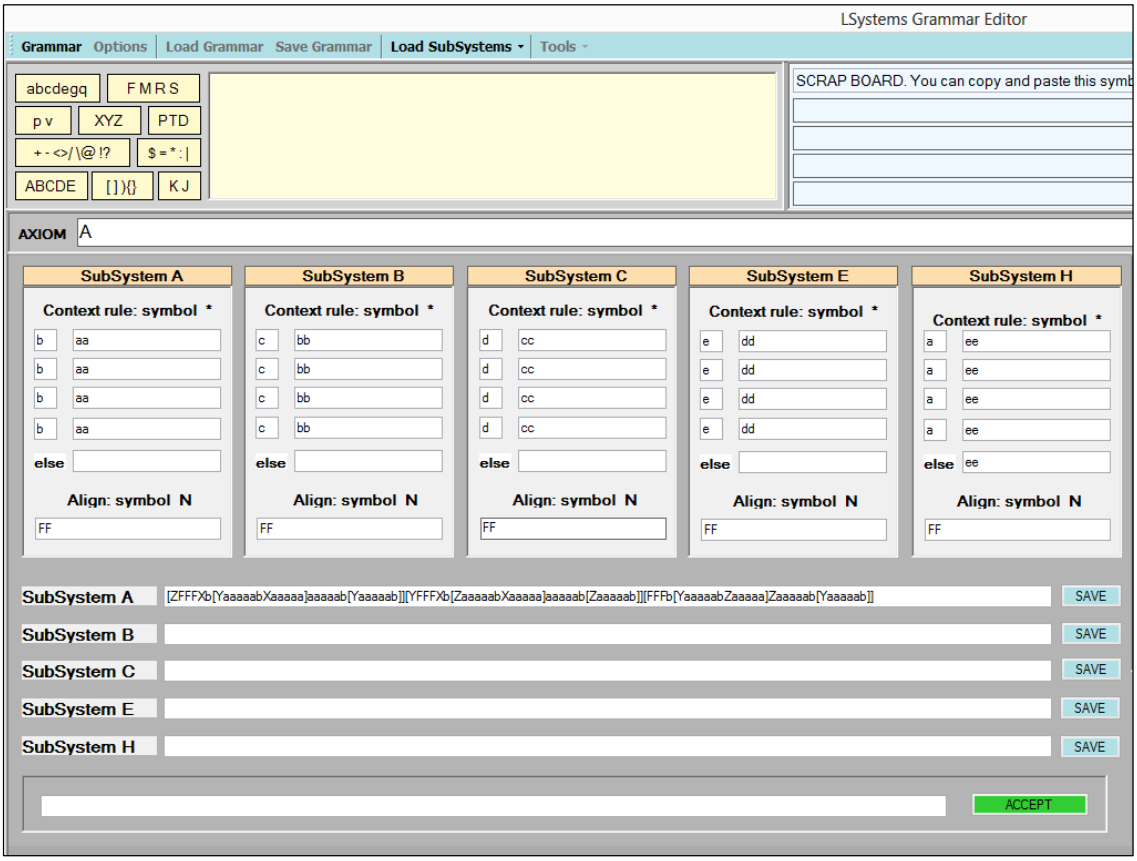
2.5 Subsistemas

Las funciones permiten dividir un programa complejo en subprogramas, los que además se pueden reutilizar en otros programas, mediante librerías.

En las gramáticas generativas se puede obtener la misma facilidad mediante subsistemas, que son L-Systems desarrollados que se pueden utilizar en las gramáticas de otros L-Systems. En GDesign esto se obtiene generando un L-System cualquiera, grabándolo como subsistema y luego importándolo en otra gramática mediante los símbolos “A”, “B”, “C”, etc.

Los subsistemas conservan sus propios parámetros y también incluyen elementos sensibles al contexto, lo que permite que estos se adapten a los diferentes símbolos vecinos[6].

Los subsistemas incluyen también símbolos de contexto para su alineación.



[6] Término tomado de los Autómatas Celulares. Se entiende los símbolos contiguos antes o después del símbolo considerado.

Fig. 63. Paleta de subsistemas.

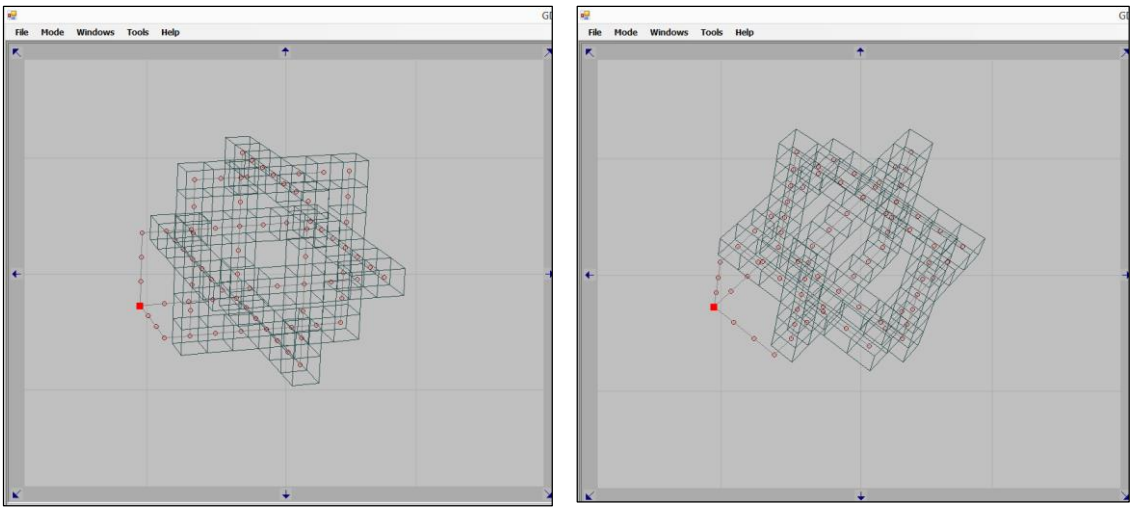


Fig. 64. Vistas del subsistema de fig. 37.

En GDesign se pueden utilizar 5 subsistemas diferentes, cada uno con su propio L-System (aunque se puede utilizar uno para todos). Cada subsistema tiene sus reglas de contexto y alineación.

Un subsistema puede ser de un solo elemento, esto sirve para tener objetos de diferentes dimensiones y comportamientos. Son como objetos inteligentes.

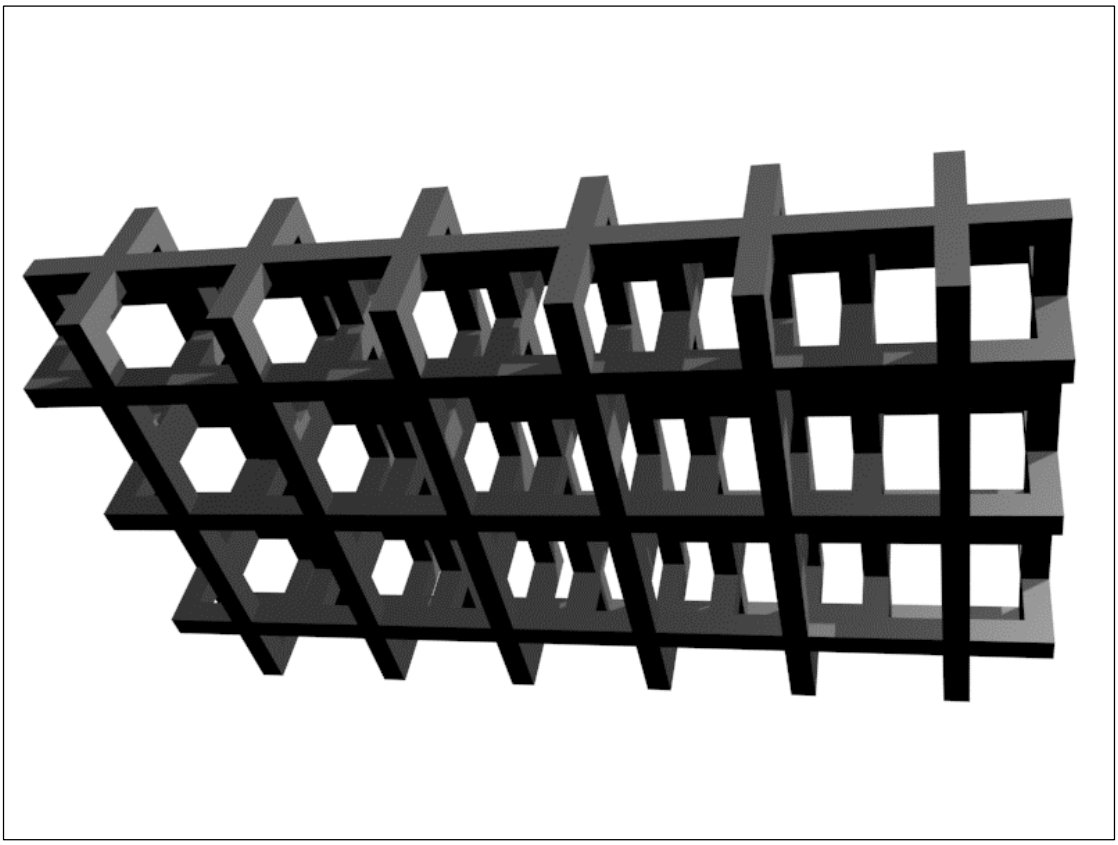


Fig. 65. Composición con subsistemas de fig. 37

Los subsistemas se pueden usar como objetos inteligentes, programables, dentro el sistema principal.

Tamaño

Ejemplo con construcción de vigas y columnas. “B” es la columna azul, “A” la viga gris.

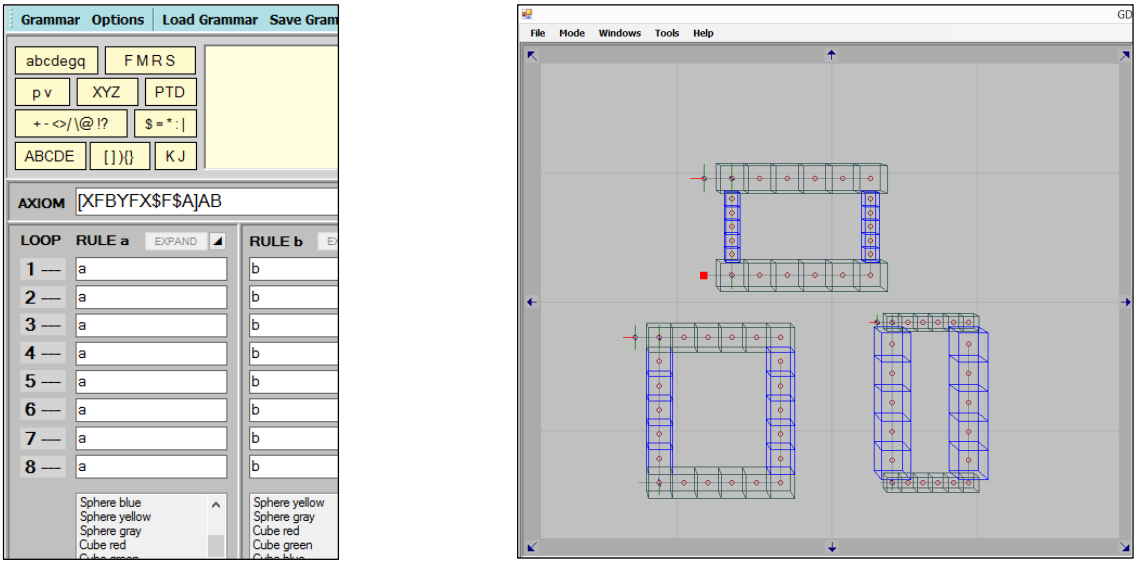


Fig. 66. La gramática con los subsistemas A y B. Modelos jugando con diferentes tamaños de los subsistemas.

Símbolo de contexto “*”

Este símbolo se inserta en un Sistema-L cualquiera, en este caso no produce nada. Pero se activa cuando este sistema es importado como subsistema. Dependiendo del contexto se le puede asignar una regla que modifica el subsistema.

Por ejemplo, en el caso anterior

A → ”bbbbb” y B → ”Yaaaaa”.

Ahora modificamos los subsistemas (directamente en la paleta, sin necesidad de volver a generarlos):

A → ”bb*bbb”. Nótese el asterisco.

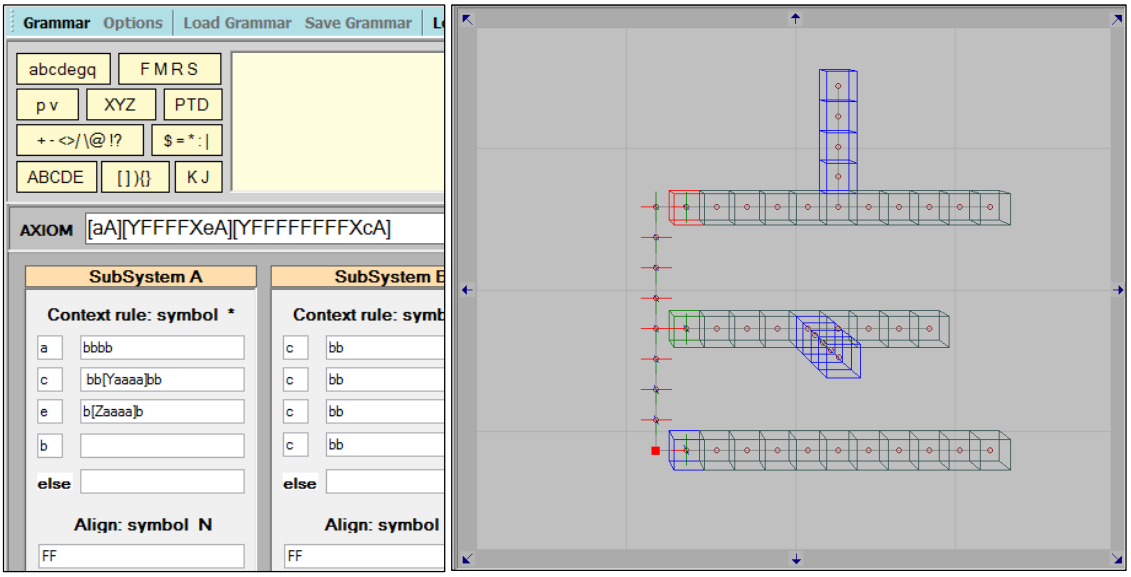


Fig. 67. La paleta para programar los subsistemas. Efectos de la programación de A después de “a”, de “e” y de “c”.

Se pueden utilizar varios “*” en un mismo subsistema. Hay 4 casos de contexto que se pueden programar.

Símbolo de alineación “O”

La alineación de un subsistema, siendo una estructura compleja, depende del primer objeto de la estructura, lo que dificulta su posicionamiento en relación a otras partes del sistema en que se utiliza. Para esto se ha creado el símbolo de contexto “O”, que puede ser programado de acuerdo a cada subsistema (son 5 disponibles).

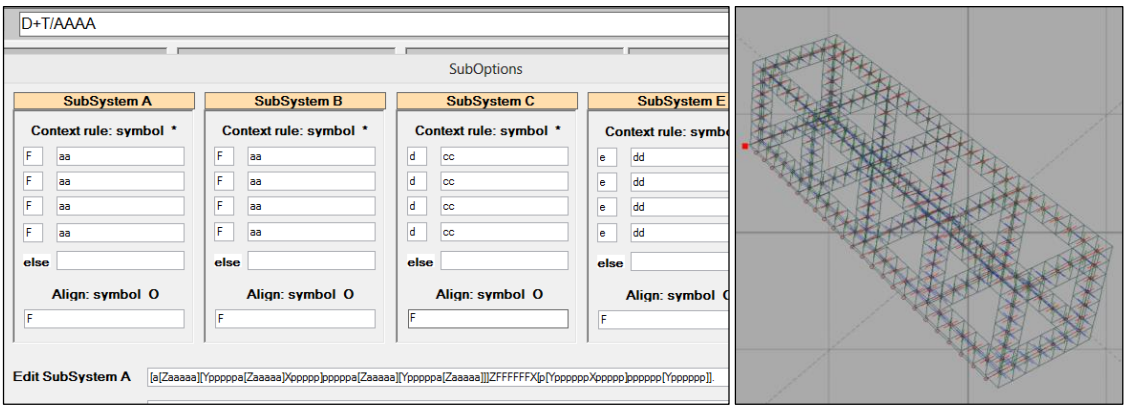


Fig. 68. Subsistemas alineados en Z, en el punto final del andamio.

Así es posible, por ejemplo, utilizar dos subsistema con el mismo contenido, pero que se ubican de forma diferente. Por ejemplo uno se ubica siempre arriba de otro, y el segundo al costado, como en el ejemplo que se muestra enseguida.

En la imagen siguiente se muestran tres subsistemas A (andamio) cuya alineación se da en el eje Z, pues en esta esquina termina el diseño del andamio.

Para alinearlos en el eje X necesito mover el punto inicial utilizando “O”.

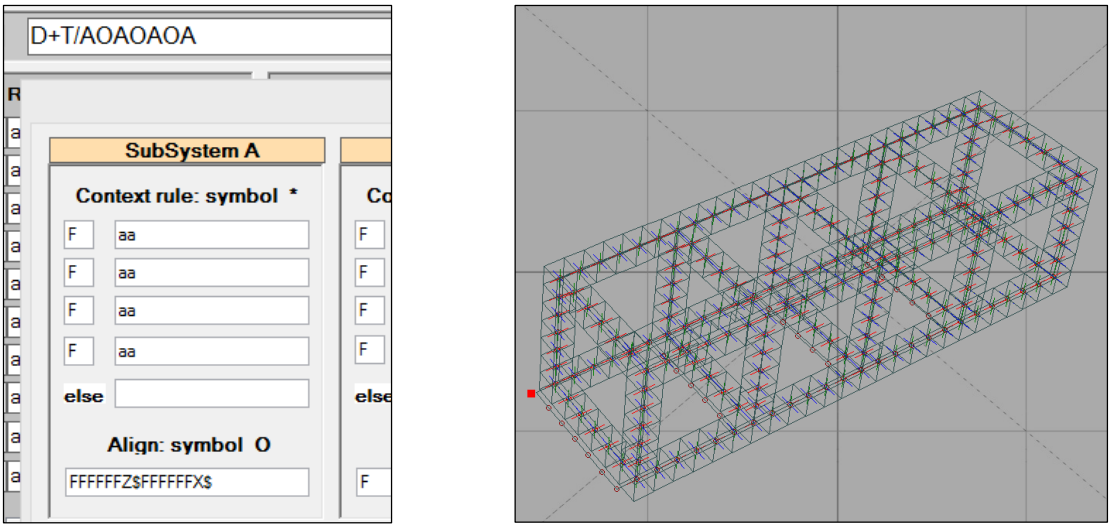


Fig. 69.

Entonces A se alinea en Z o en X utilizando “O”. Así mismo, cargando el mismo subsistema en B, este se puede alinear en Z o en Y programando su “O”, etcétera.

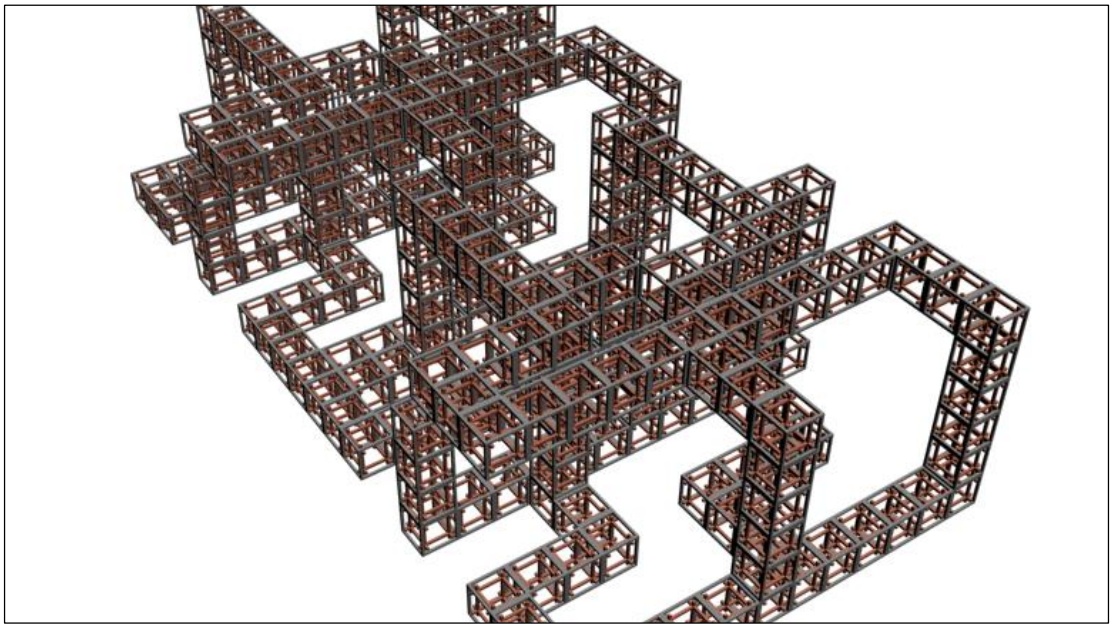


Fig. 70.

De esta forma se pueden generar estructuras muy complejas. En las siguientes imágenes diferentes combinaciones de vigas y columnas con subsistemas A, B y alineación con O.

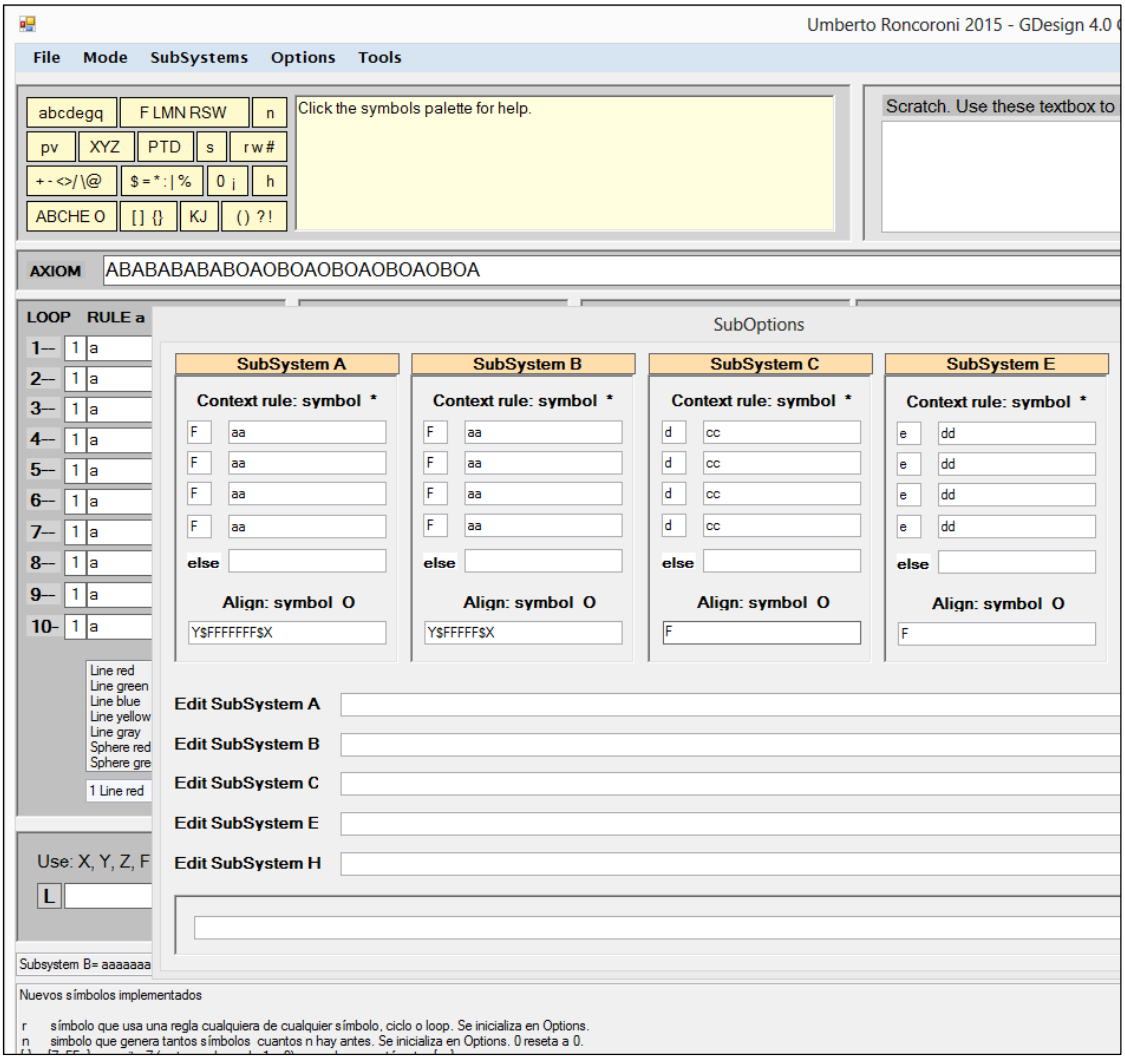


Fig. 71.

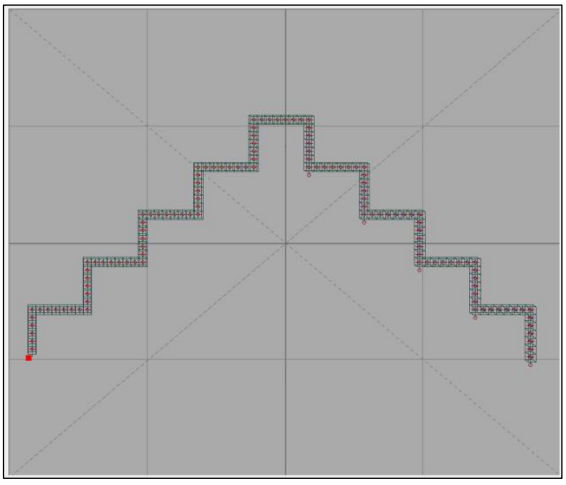


Fig. 72. Modelo de la gramática anterior.

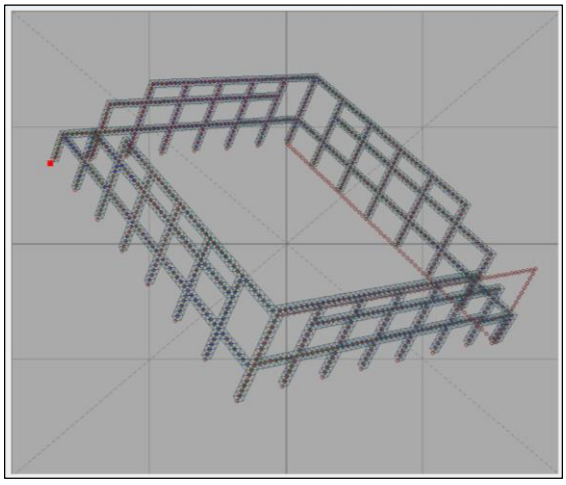


Fig. 73. Una estructura más compleja.

Símbolo de contexto vacío “h”

Este símbolo no tiene objeto ni posición, sino que sirve en asociación a un subsistema para que el subsistema pueda producir como un símbolo cualquiera.

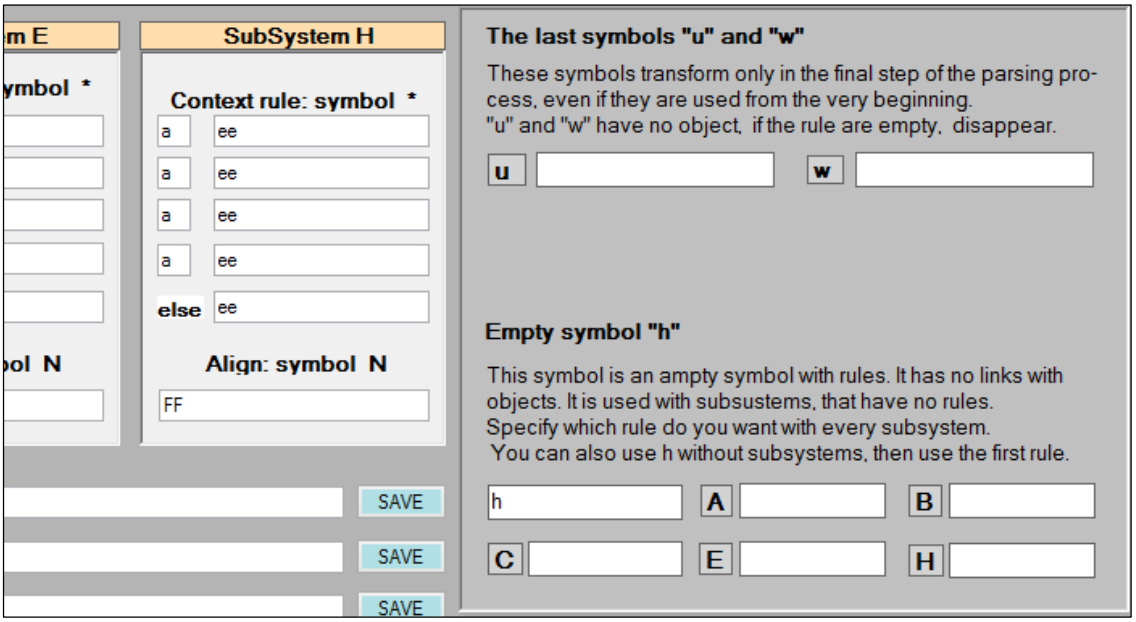


Fig. 74. Una estructura más compleja

Subsistemas, programación de “*”

“*” se puede programar:

- Poniendo también un sucesor para ampliar el contexto
- Con “=” bloqueando en un cierto contexto. Por ejemplo si A es “aaa*aaaa*=aaa”, el 2 asterisco con “=” está bloqueado si el subsistema es contiguo, supongamos, a un símbolo “a”.
- Modificando los ángulos de rotación del subsistema de acuerdo al contexto.
- Ignorando ciertos símbolos del subsistema de acuerdo al contexto. Por ejemplo si A es “aaabbaaa”, “b” no se consideraría, sería como “aaaaaa”.
- Utilizando un ancla, es decir un punto donde se conecta el siguiente objeto.

2.6 Reglas en base a imágenes

Es posible utilizar una imagen *bitmap* para establecer las reglas de cualquier símbolo. Se proyecta una imagen en el espacio de trabajo en el eje Z y se calcula la posición relativa de cada objeto en el plano con respecto al plano de la imagen. Luego se determina el valor de la zona de la imagen que coincide con el objeto y en base a este dato se aplican las reglas.

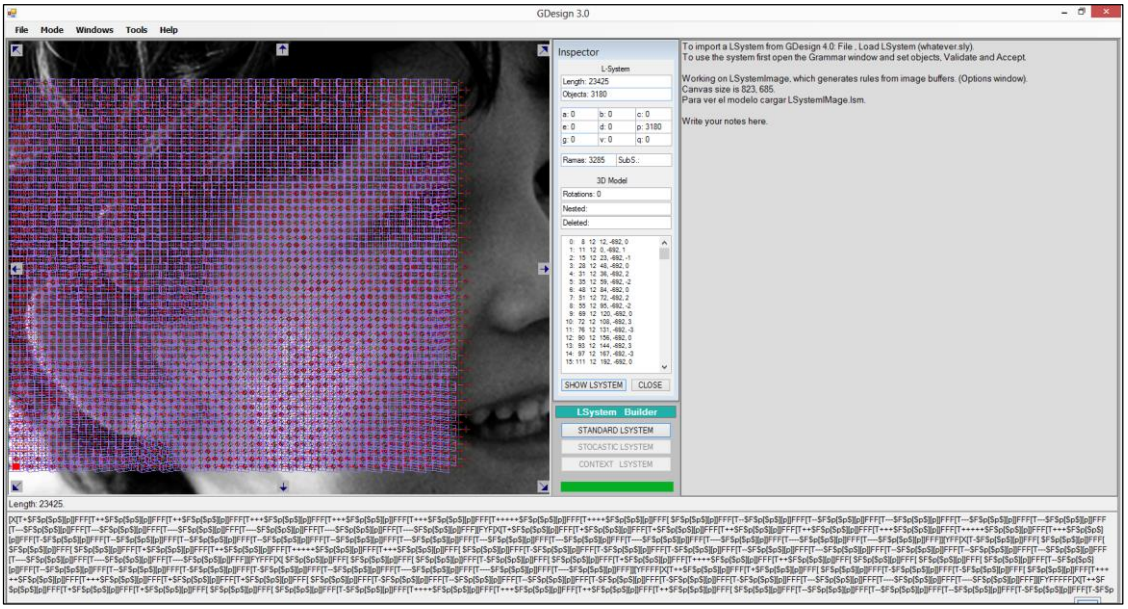


Fig. 75. Vistas del subsistema de fig. 37.

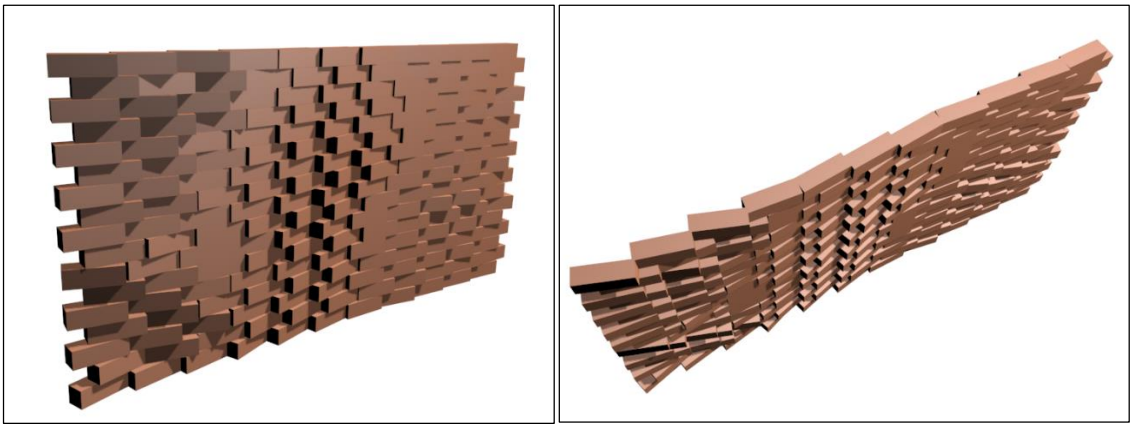


Fig. 76. Orientación de los ladrillos de acuerdo a los valores RGB e la imagen.

Las reglas pueden determinar, por ejemplo, la rotación de un objeto. La rotación de los ladrillos de este ejemplo hace un juego de luz y sombra que reproduce los tonos

de la imagen original. La imagen además debería ser pre elaborada (por ejemplo con Photoshop) para optimizar el resultado.

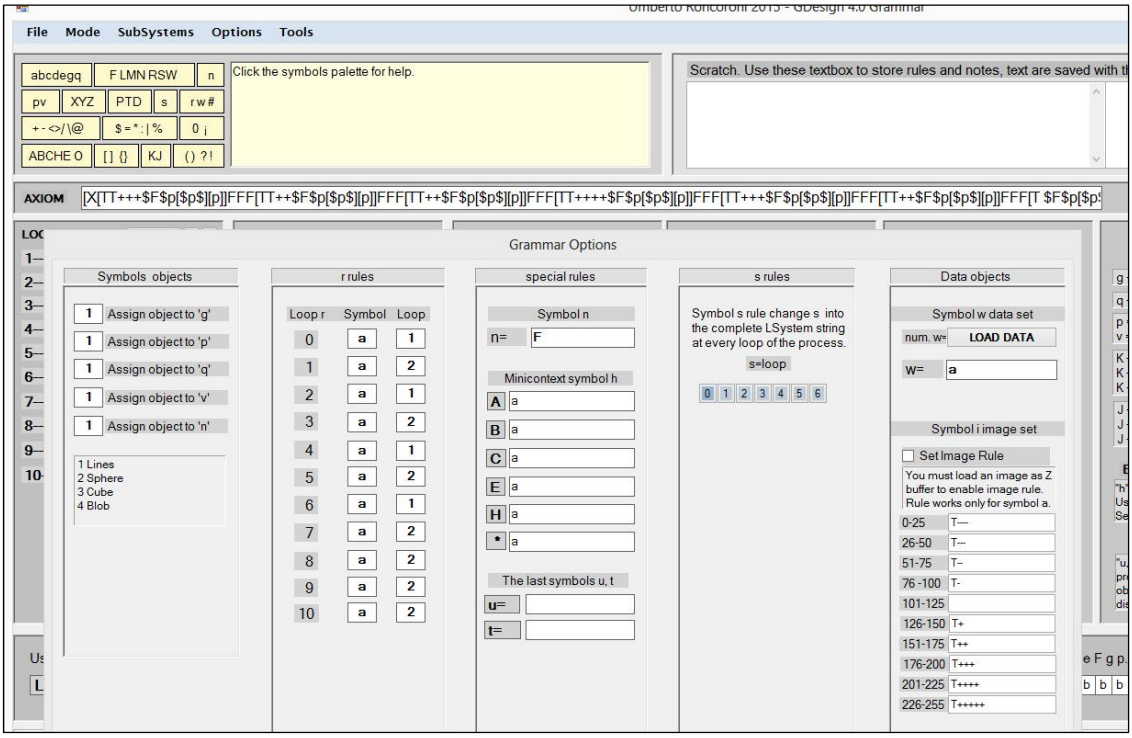


Fig. 77. Vista de las reglas de imagen de fig. 76.

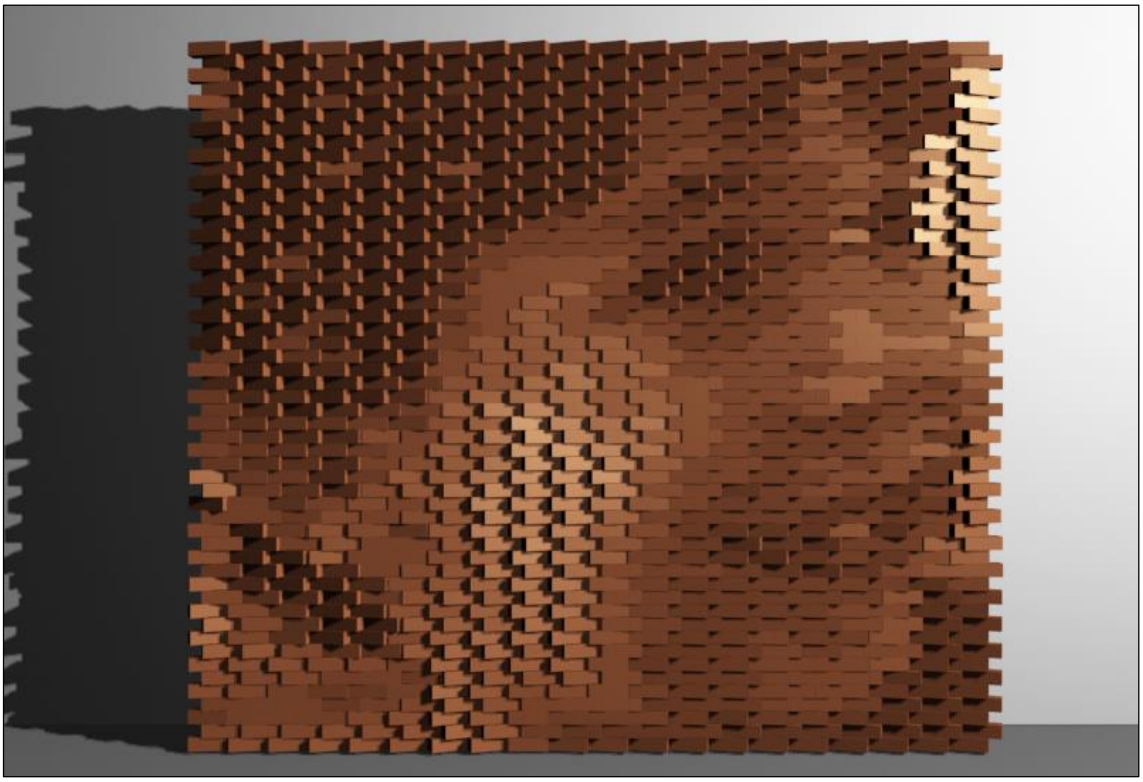


Fig. 78. Vista de la gramática de fig. 77.

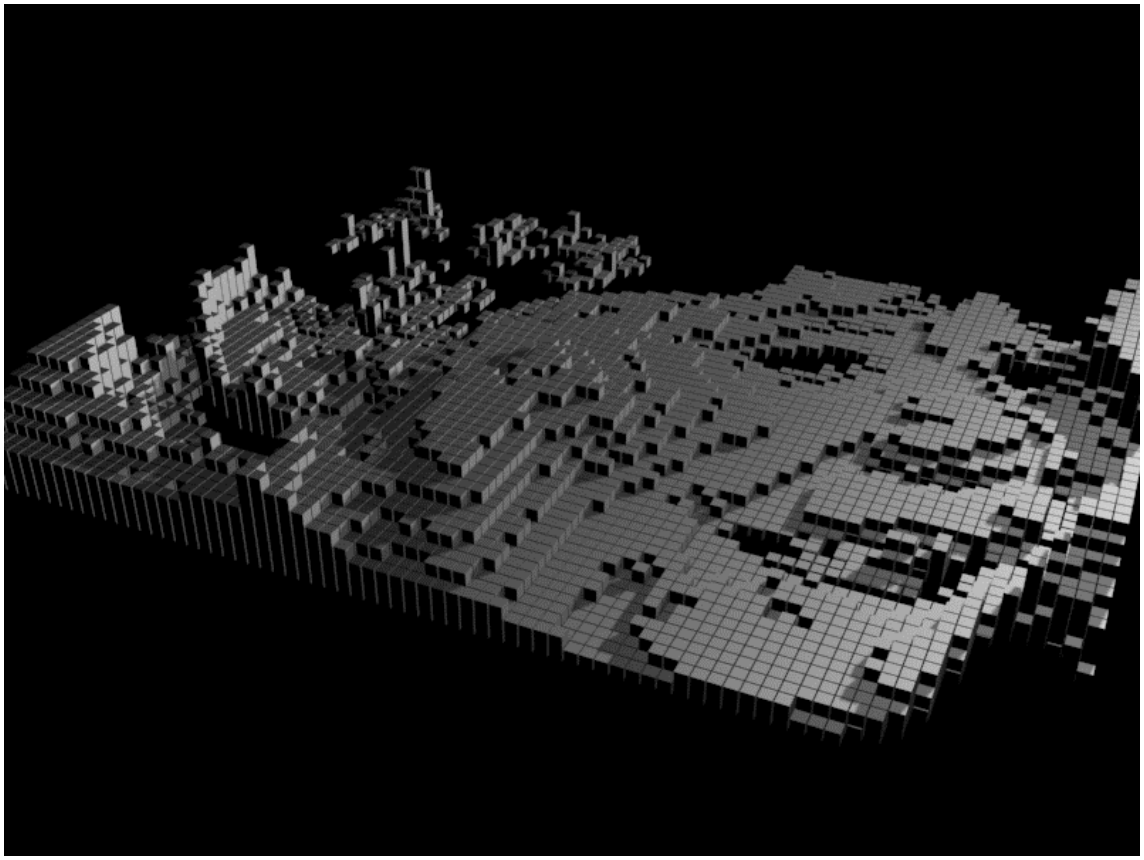


Fig. 78. Vistas del subsistema de fig. 37.

2.7 Símbolos especiales

Se trata de símbolos que permiten hacer tareas específicas, por ejemplo para controlar las rotaciones, los tamaños, etcétera.

Se trata de los símbolos “!”, “%”.

2.7.1 Símbolo “!”

Permite invertir el ángulo para hacer rotaciones simétricas. Tiene un multiplicador que permite invertir el doble, el triple, etc. cómo se muestra en el ejemplo. Por ejemplo, en el sistema “aaaP+aaa!aaa!aaa!aaa!aaa!aaa!aaa!aaa!aaa”, suponiendo que “+” = 30°, y que el multiplicador de “!” sea 2, el resultado del *parser* sería: 3 objetos de tipo “a”, rotación en el eje Z de 30°, 3 objetos de tipo “a”, rotación en Z de -60°, 3 objetos de tipo “a”, rotación en el eje Z de 60°, … etcétera.

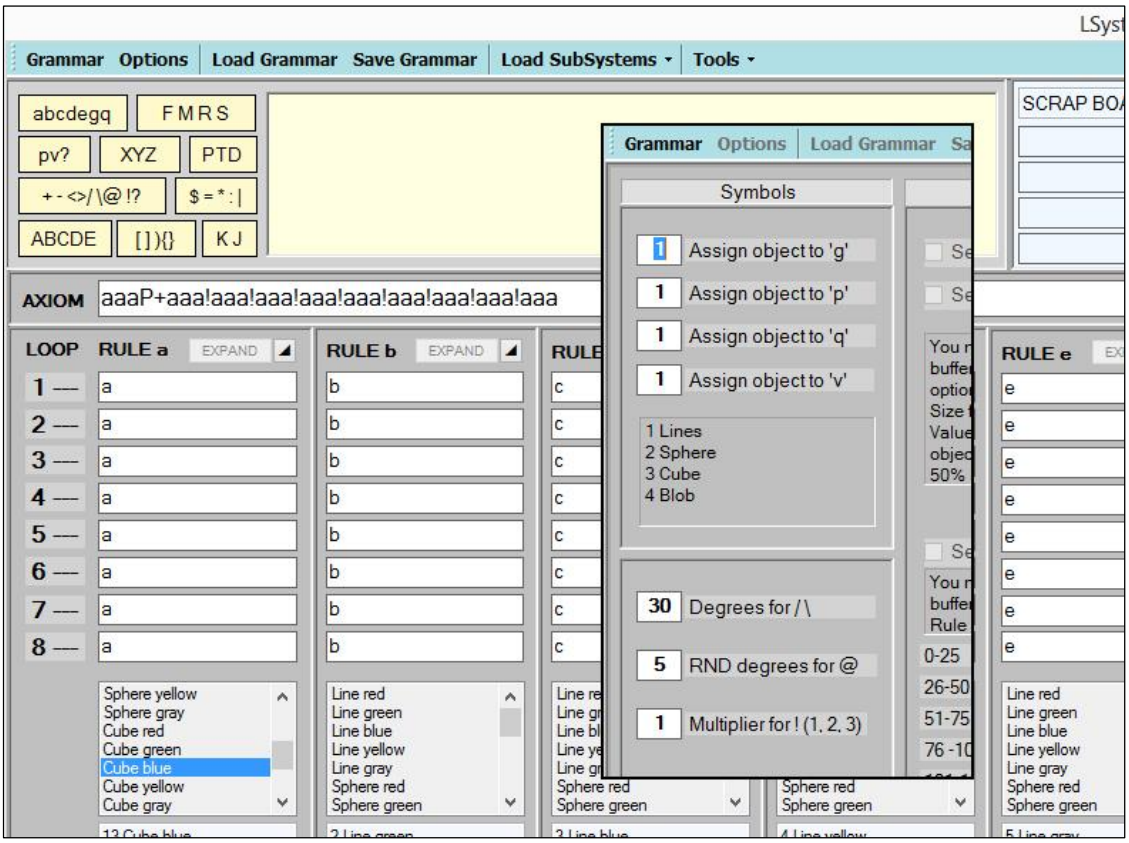


Fig. 79. Gramática con símbolo "!"

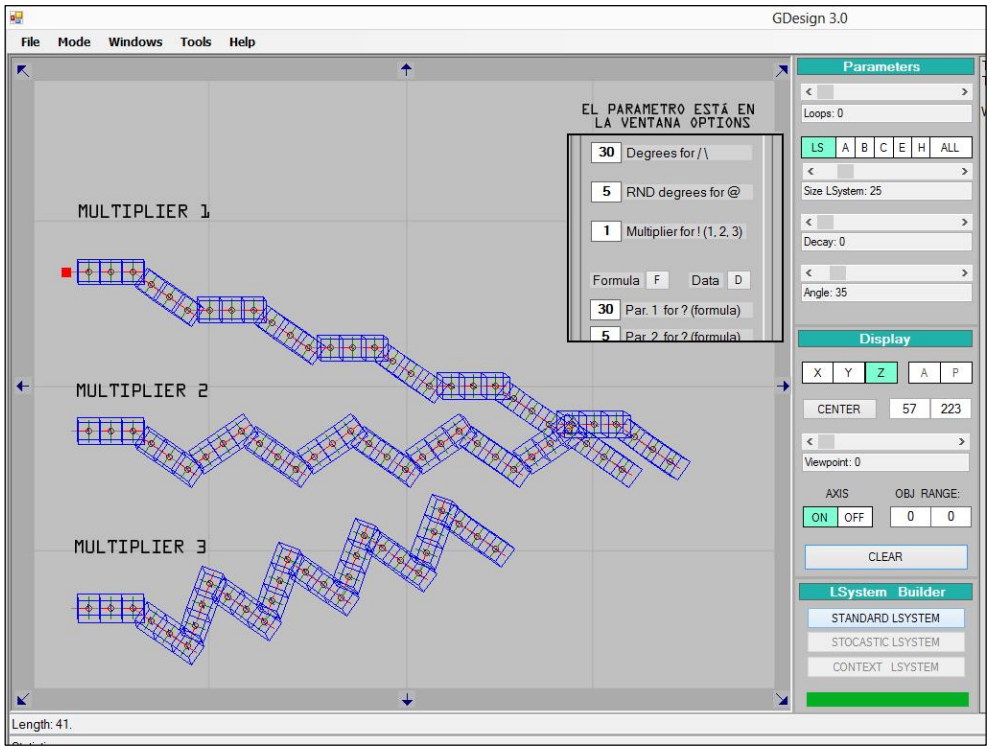
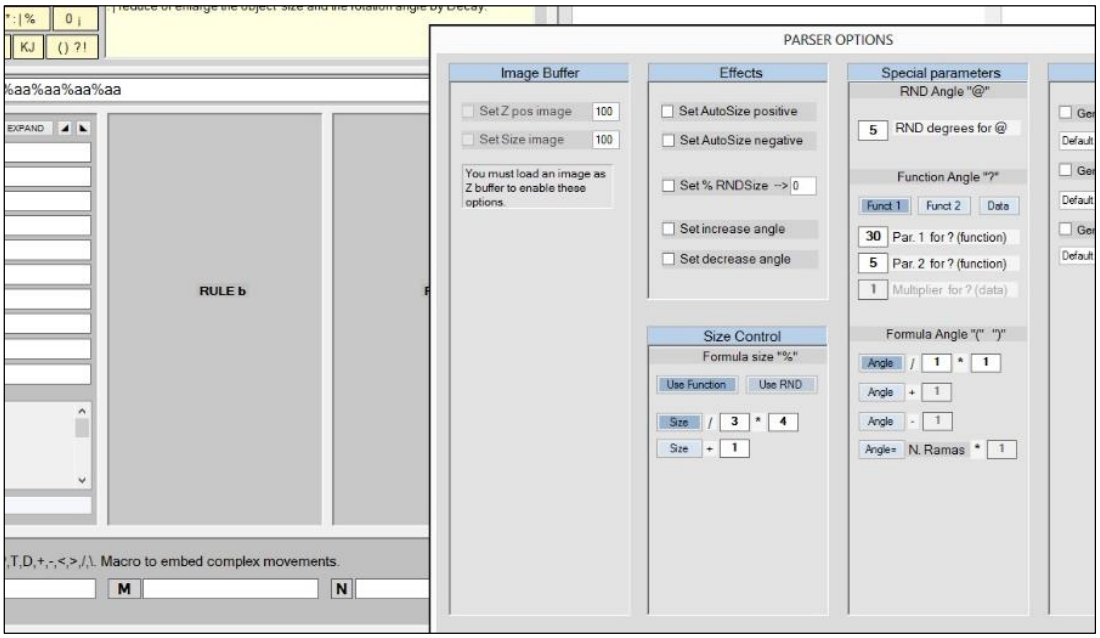


Fig. 80. Efectos con multiplier.

2.7.2 Símbolo “%”

El símbolo “%” permite definir el tamaño de los objetos de acuerdo a un valor preciso o de acuerdo a unas expresiones matemáticas o funciones. Para volver al tamaño normal (fijado con el parámetro Size en la ventana de diseño) se utiliza el símbolo “#”.



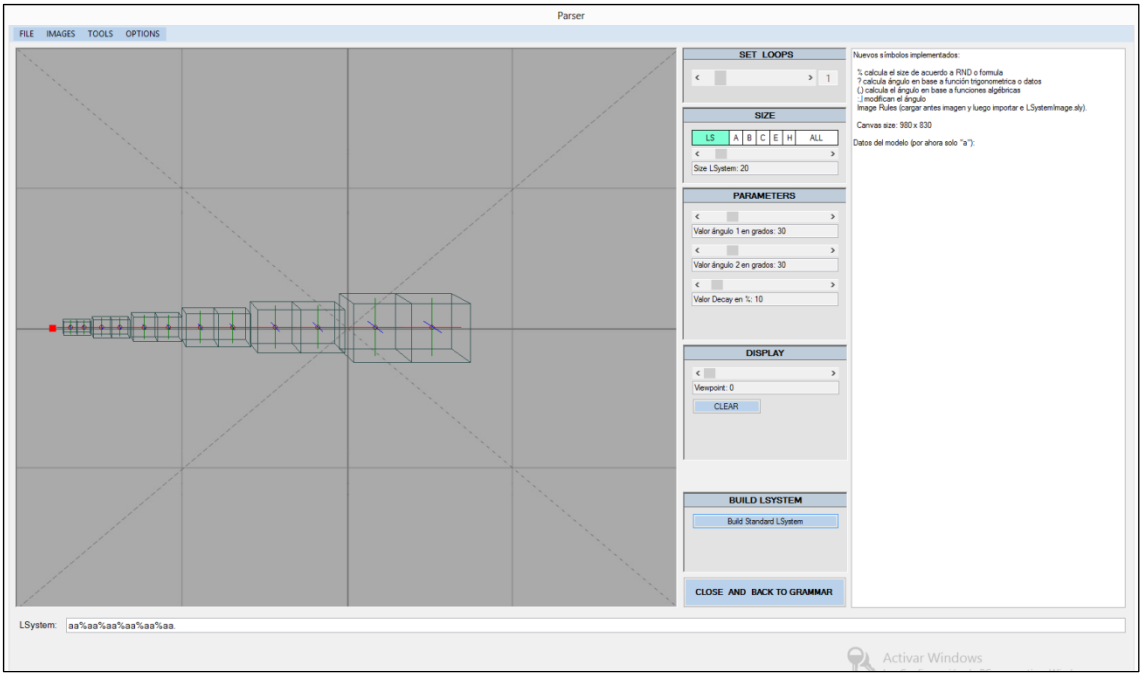


Fig. 81-82. Medición del tamaño con “%”.

Notar que “#” se puede utilizar también para reestablecer el valor original del ángulo de rotación. Notar también el uso y el efecto del símbolo “#” modificando el axioma de la gramática anterior:

“P+aa+%aa+%aa+%aa+%aa+%aa+#aa+%aa+%aa+%aa+%aa+%aa#aa+%aa+%aa+%aa+%aa+%aa#aa+%aa+%aa+%aa+%aa+%aa”.

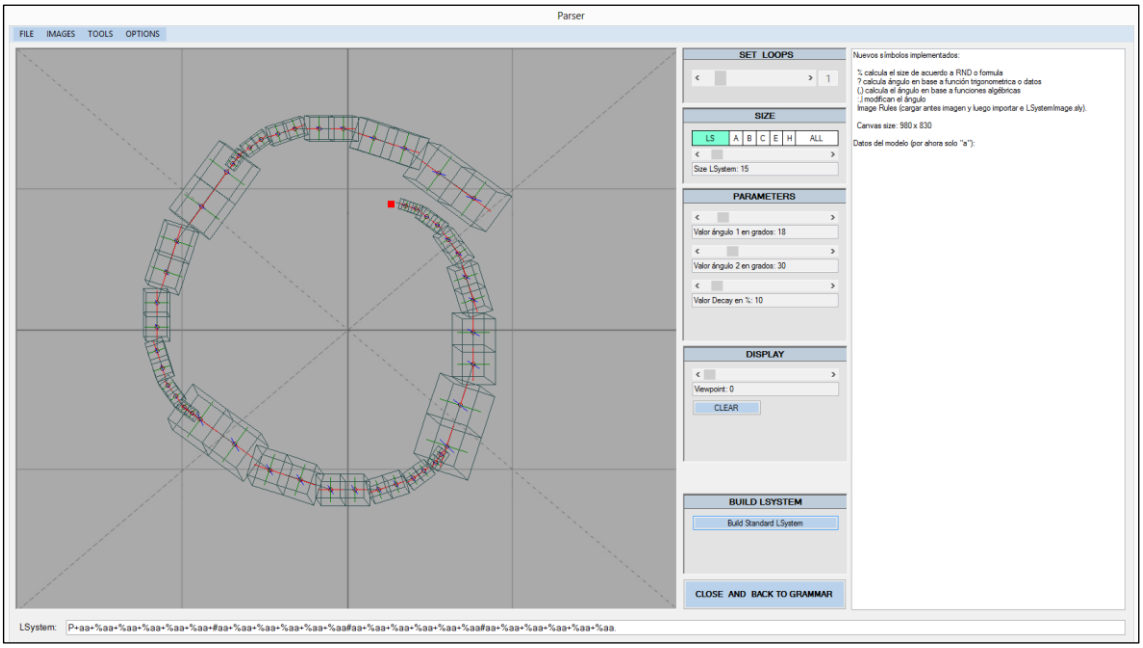


Fig. 83. Medición del tamaño con “%”.

2.7.3 Símbolo recursivo “s”

“s” es un símbolo que se sustituye con el mismo sistema en un determinado paso del proceso (anterior a cuando aparece el símbolo “s”).

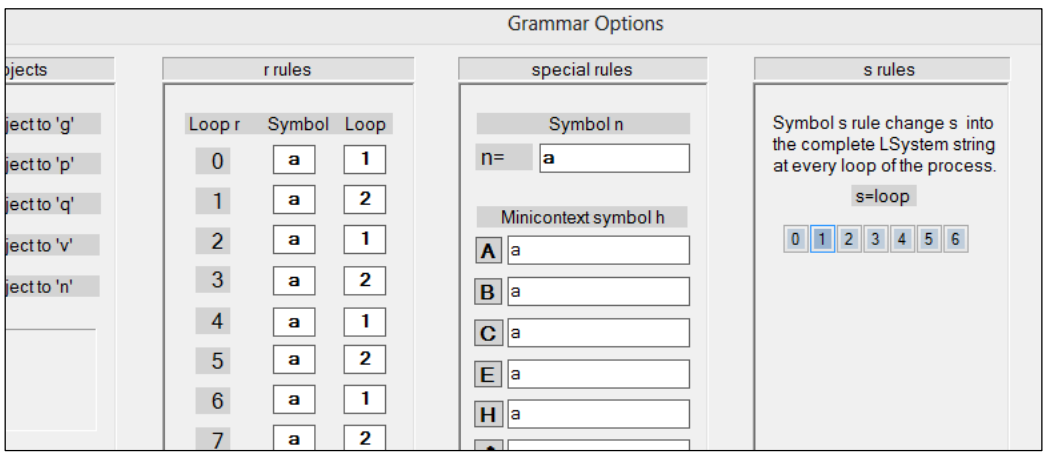


Fig.84. Paleta del símbolo “s” a la derecha.

En el ejemplo “s”, que aparece en la regla 2, tomará el valor del sistema al paso 1, como se ve en la siguiente figura.

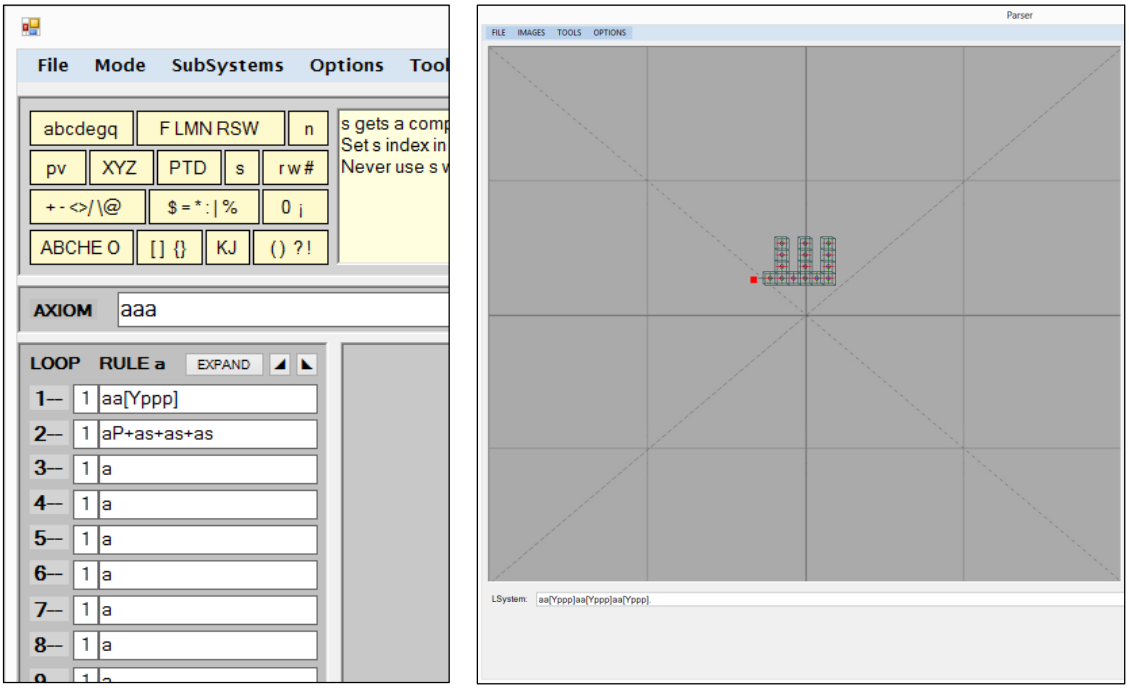


Fig. 85. La gramática y el sistema como aparece hasta la regla 1.

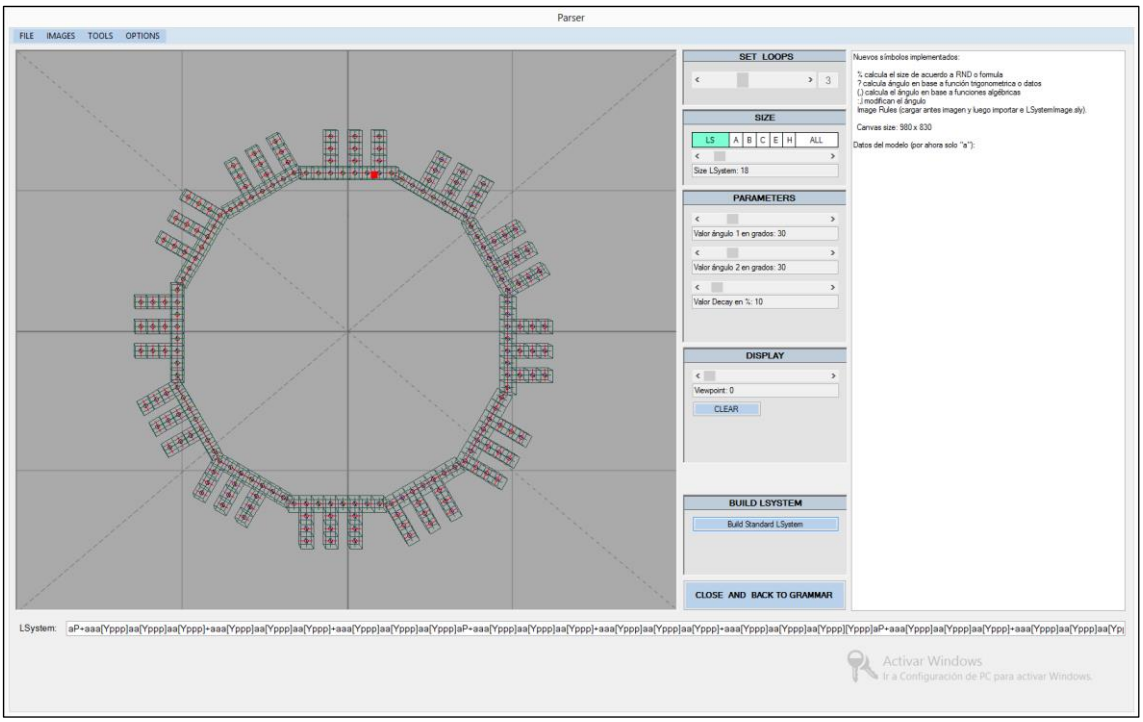


Fig. 86. El sistema al paso 3. “s” (regla 2) ha tomado el valor de la imagen anterior (sistema al paso 1).

“s” puede ser utilizado en modo recursivo, utilizando varias “s” en diferentes etapas. Lo importante es que sean separados de 2 niveles.

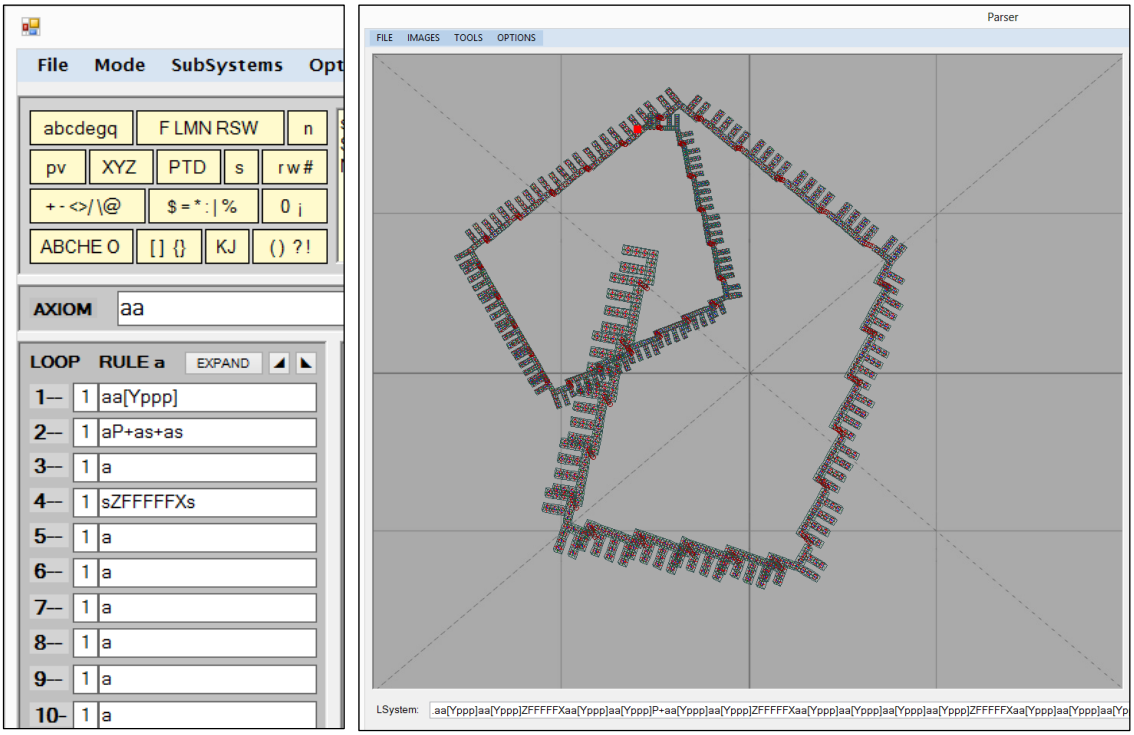


Fig. 87. El sistema al paso 5. “s” (regla 4) ha tomado el valor del sistema al paso 1.

Programación del símbolo “s”

El nivel del sistema que usa “s” se establece en Options y es fijo. Es decir, si se establece que “s” toma el sistema al nivel 1, cualquier “s” después tomará siempre el sistema a nivel 1.

Pero se puede fijar el nivel de “s” por ejemplo:

Sistema de “s”= num regla en que está “s” -1. Entonces, si “s” aparece en el nivel 4 y 7, en el 4 toma el sistema a nivel 3, y en el 7 el sistema a nivel 6.

“s” puede ser utilizado con los símbolos especiales “u” y “t” los híbridos que producen solo al final.

Cap. 4. Estudio sobre Moray

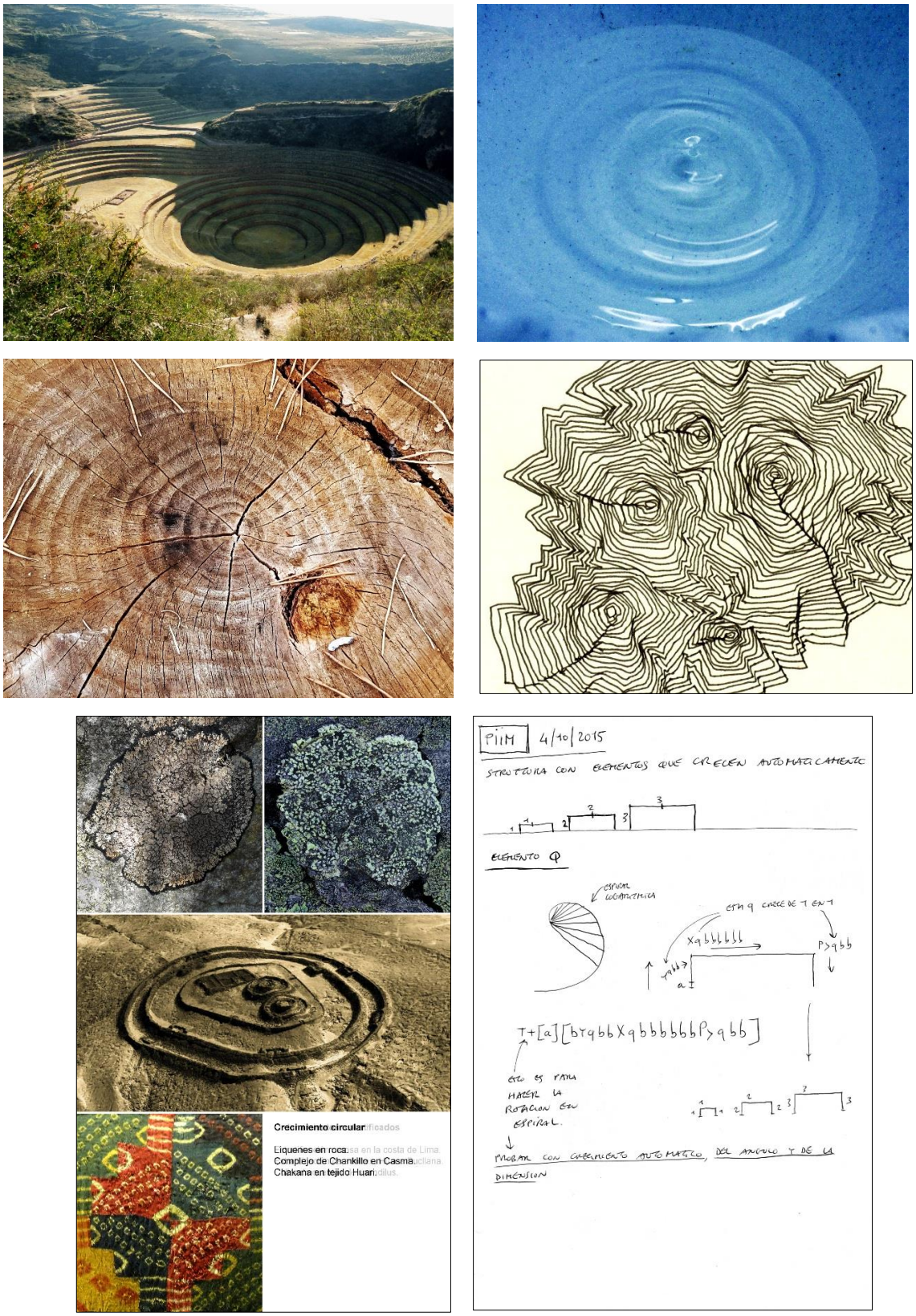
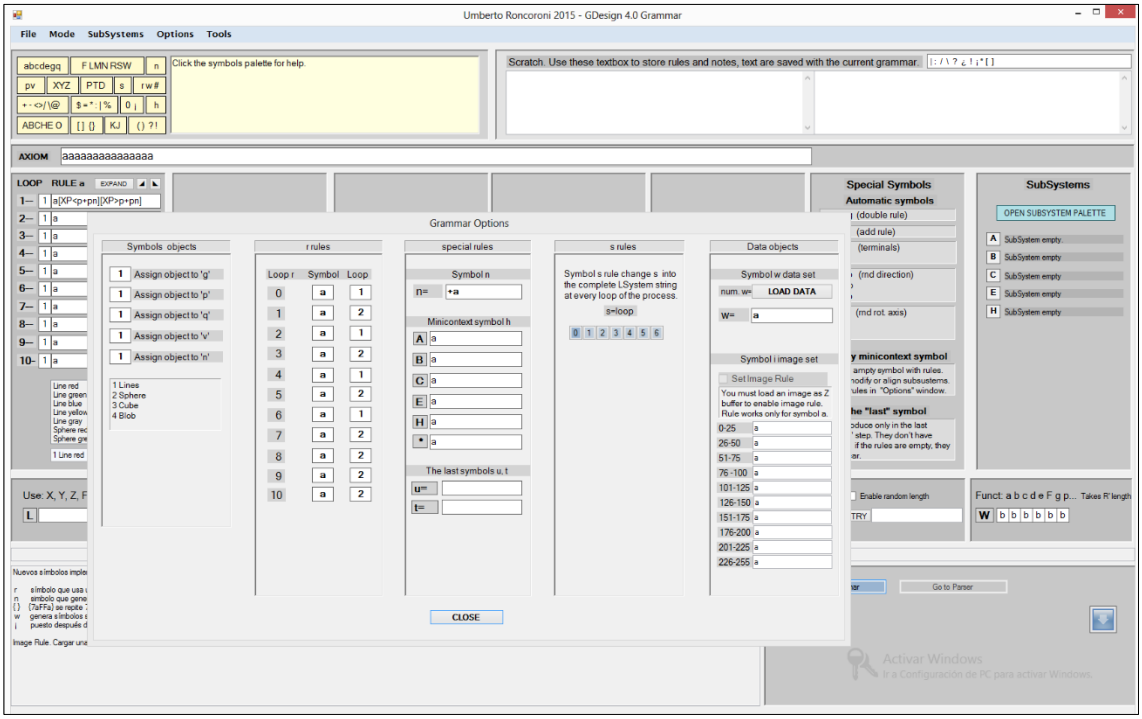


Fig. 88-89. Documentación fotográfica, dibujo a mano alzada, ficha comparativa y diseño de reglas



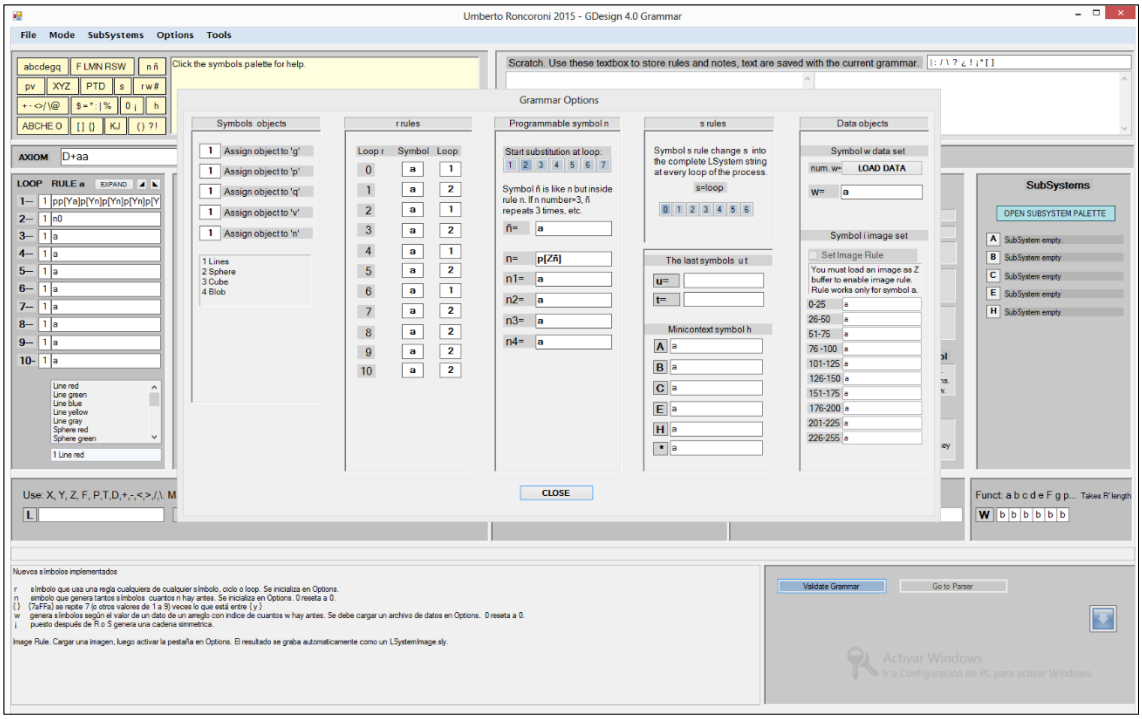


Fig. 90-91. Implementación en *shape grammars*.

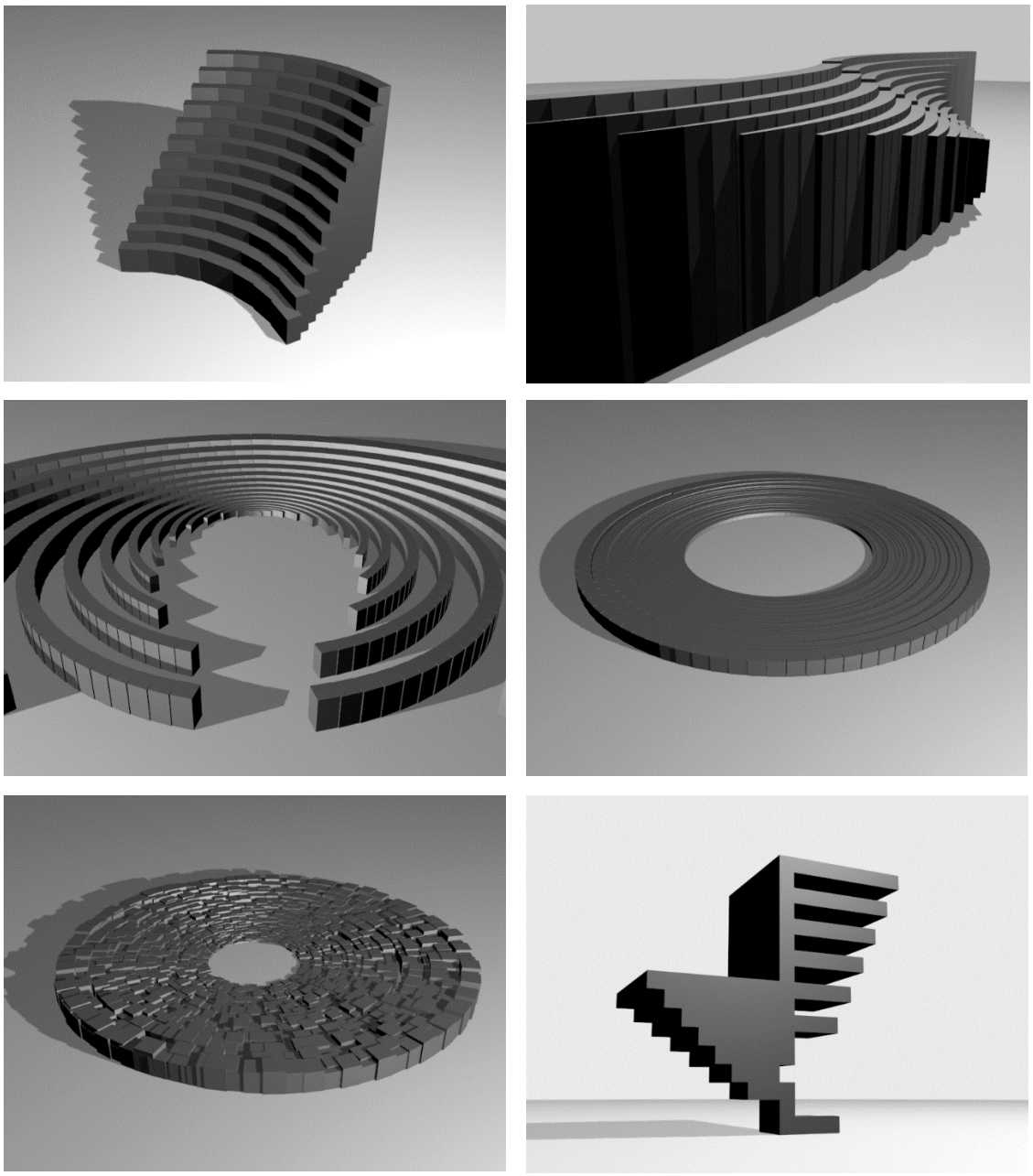
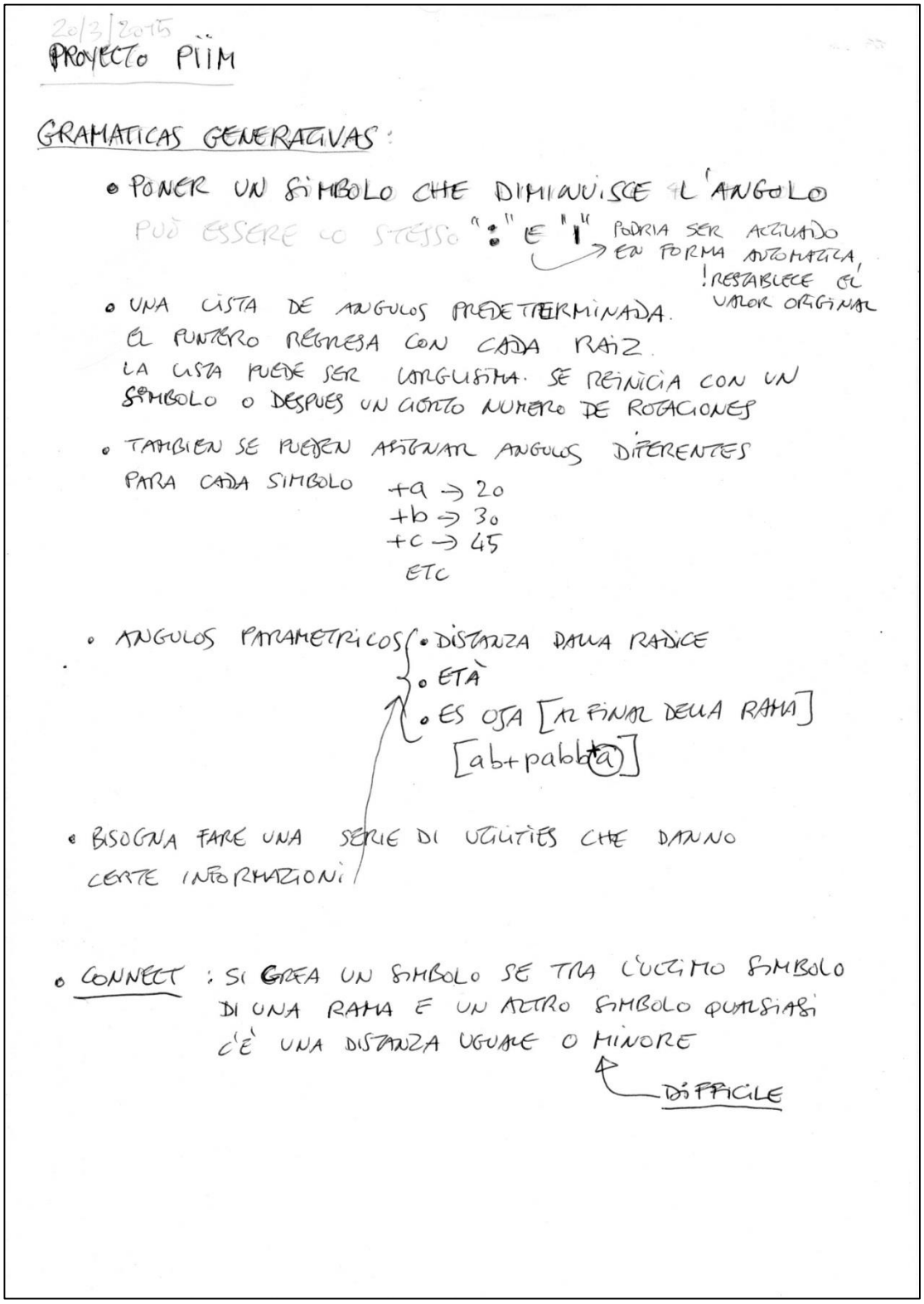


Fig. 92. Pruebas de las diferentes gramáticas.

Las gramáticas han ido evolucionando y la experimentación ha ofrecido indicaciones para perfeccionar las reglas de los símbolos programables. La última imagen a la derecha resulta de una variación de la gramática base. Lo importante de las *shape grammars* es precisamente la posibilidad de aplicar las reglas en modo creativo, obteniendo resultados diferentes del modelo original (Moray), sin embargo conservando algunos aspectos formales que le dan un estilo similar.

Anexo 1



20/3/2015

PROYECTO PIIM

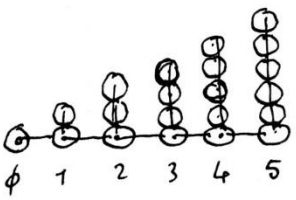
GRAMATICAS GENERATIVAS:

- PONER UN SIMBOLO CHE DIMINUISCE L'ANGOLO
  PUÒ ESSERE LO STESSO "+" E "-" → PODRIA SER ACCIUATO EN FORMA AUTOMATICA, !RESTABLECE EL VALOR ORIGINAL
- UNA LISTA DE ANGULOS PREDETERMINADA.
  EL PUNTERO REGRESA CON CADA RAIZ.
  LA LISTA PUEDE SER LONGUISIMA. SE REINICIA CON UN SIMBOLO O DESPUES UN CIERTO NUMERO DE ROTACIONES
- TAMBIEN SE PUEDEN ASIGNAR ANGULOS DIFERENTES PARA CADA SIMBOLO
  +a → 20
  +b → 30
  +c → 45
  ETC
- ANGULOS PARAMETRICOS
  - DISTANZA DALLA RADICE
  - ETÀ
  - ES OSA [AL FINAL DELLA RAMA]
    [ab+pabb@]
- BISOGNA FARE UNA SERIE DI UTILITIES CHE DANNO CERTE INFORMAZIONI
- CONNECT: SI CREA UN SIMBOLO SE TRA L'ULTIMO SIMBOLO DI UNA RAMA E UN ALTRO SIMBOLO QUALSIASI C'È UNA DISTANZA UGUALE O MINORE
  ← DIFFICILE

7 MARZO 2015 (7)

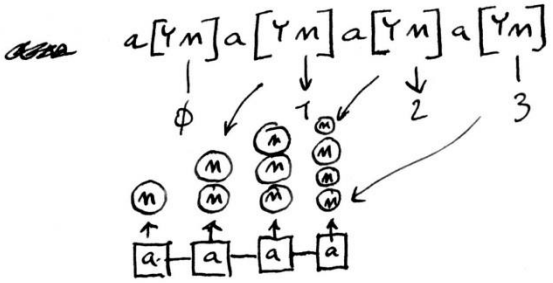
IDEAS PARA LSYSTEMS

COME FARE UN DISEGNO COSÌ:



UN SIMBOLO ϕm → ←
1m → mm
2m → mmm
3m → mmmm
4m ETC.
5m

FUNZIONA COSÌ: LA REGOLA DI m È QUE PRODUCE UNA SERIE DI SIMBOLI LUNGA IL NUMERO DI m, CIOÈ QUANTI m CI SONO PRIMA PER ESEMPLO:



LA GRAMMATICA SAREBBE COSÌ:

b = a[Ym]b

b
a[Ym]b
a[Ym]a[Ym]b
a[Ym]a[Ym]a[Ym]b

TERMINAL

a = p[Ym]a[Ym]

1) a
2) p[Ym]a[Ym]
3) p[Ym]p[Ym]a[Ym]
4) p[Ym]p[Ym]p[Ym]a[Ym]

ETC.

7 MARZO 2015

IDEAS X LSYSTEMS

SIMBOLO DESTRUCTOR "~"

$$\sim \begin{cases} a\sim\cancel{a} = a \\ \cancel{a}\sim b = b \end{cases}$$

OPPURE ELIMINA TUTTE E DUE

EJEMPLO:

$abbpb\sim aaa \rightarrow abbp\sim\cancel{a}aa \rightarrow abbpaa$

$abbpb\sim aaa \rightarrow abb\cancel{p}\sim aaa \rightarrow abbaaa$

$abbpb\sim aaa \rightarrow abb\cancel{p}\sim\cancel{a}aa \rightarrow abbaa$

STUDIARE CASI STRANI QUANDO SI COMBINANO GRUPPI DI QUESTI SIMBOLI

- EL DESTRUCTOR SI ADELANTE TIENE UNA RAMA DESTRUYE TODA LA RAMA

EJEMPLO $a\sim[ppppa]b \rightarrow a\sim\cancel{[ppppa]}b \rightarrow ab$

OPPURE PUÒ FONDERE 2 RAMIFICAZIONI

$pa[bbb]\sim[ppp] \rightarrow pa[bbbppp]$ SI DANNO VARIE COMBINAZIONI

BORRA][

$pa[bbb][\sim[ppp]]$ BORRA [E [ppp]

CIOÈ BORRA LA PRIMA [O] QUE TROVA

O

LA PRIMA] O [CHE TROVA

7 MARZO 2015

IDEE X L SYSTEMS

UN SIMBOLO " " CHE USA LE REGOLE DI ALTRI SIMBOLI IN QUALSIASI ORDINE. FA UN PÒ LE VECI DEL GOTO

	LIVELLO	SIMBOLO	REGOLA DEL SIMBOLO
" "	1	a	2
	2	a	1
	3	b	6
	4	b	1
	5	a	3
	6	b	4

	SIMBOLO	REGOLA
1	☐	☐
2	☐	☐
3	☐	☐
4	☐	☐

TEXT BOX TEXT BOX

↳ POWER EN OPCIONES

CHE PERÒ SI PUÒ METTERE NEI SIMBOLI, PER ESEMPIO UN SIMBOLO CHE FA RETROCEDERE IL LIVELLO O AVANZARE IL LIVELLO, CIOÈ L'ORDINE DELLE REGOLE.
SI PUÒ FARE ANCHE IN MODO INDIPENDENTE PER OGNI SIMBOLO.
CIOÈ FORZARE UNA REGOLA, AFFINCHÉ QUEL LIVELLO 1 PUÒ USARE LA 9 O NEL 9 LA 1

a(9) → SIGNIFICA CHE USA LA REGOLA 9, SIA DOVE SIA.

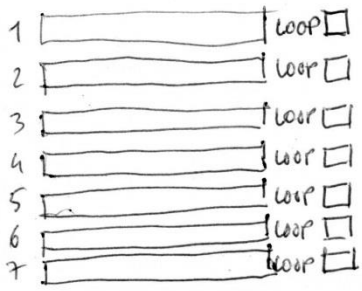
SE NON C'È NUMERO USA LA REGOLA CORRENTE, CIOÈ SE È NEL LIVELLO 3 USA LA 3, ETC.

PROYECTO PiiM 5/4/2015

REGLAS GENERATIVAS

⊙ COMO IMPLEMENTAR BUCLES EN LOS LSYSTEMS.

SE PUEDE HACER UN NUMERO AL COSTADO DE LA REGLA QUE INDICA CUANTA VECES SE REPITE



POR EJEMPLO 3, APLICA LA REGLA OBVIAMENTE SOBRE SI MISMA - PRE CALCULA TODO.

EJ: a = apb

1 apb

2 apbpb

3 apbpbpb ETC.

LUEGO

a = apbpbpb

LA VENTAJA DE ESTE SISTEMA ES QUE EL VALOR DEL LOOP SE CONVIERTE EN UN PARAMETRO.

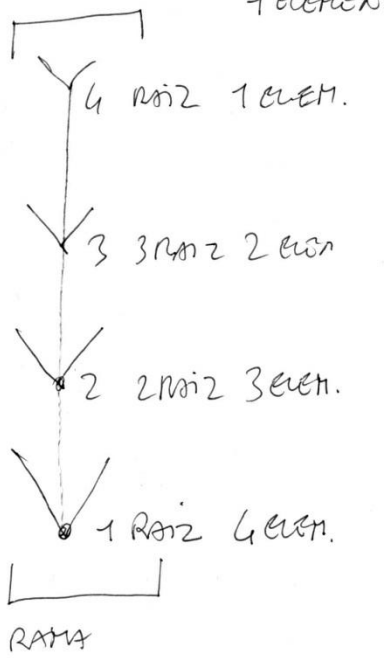
ESTE VALOR DE BUCLE PUEDE SER FIJO O DEPENDER DE CIERTAS CONDICIONES.

⊙ COMO IMPLEMENTAR IF EN LAS REGLAS.

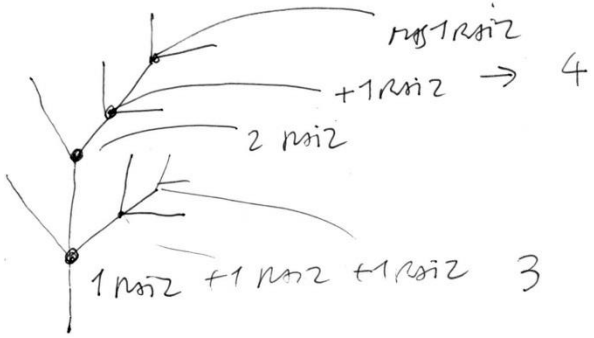
- UN MODO ES LA REGLA DE CONTEXTO, QUE DEPENDE DEL ANTECESOR Y DEL SUCESOR
- OTRO MODO ES EL ~~LOOP~~ VALOR DEL LOOP, QUE PUEDE VARIAR DE ACUERDO A CIERTOS FACTORES, COMO LA CANTIDAD DE UN SIMBOLO, SU EDAD
  - LA EDAD SE DETERMINA EN BASE A LA POSICION EN LA CADENA. SI ES LA MISMA LA EDAD AUMENTA SIMBOLOS VIEJOS YA NO PRODUCEN.
  - PODRIAMOS LLAMAR ESTOS PARAMETROS FILTROS O DETERMINANTES
- TAMBIEN SE PUEDE HACER CONFRONTANDO LA REGLA CON LA REGLA DE OTROS SIMBOLOS.
-

GRAMÁTICA-1 13/4/2015

- COMO DETERMINAR UN ARBOL LA LONGITUD DE UNA RAMA DE ACUERDO A LA RAIZ, ES DECIR, CUANTAS RAICES

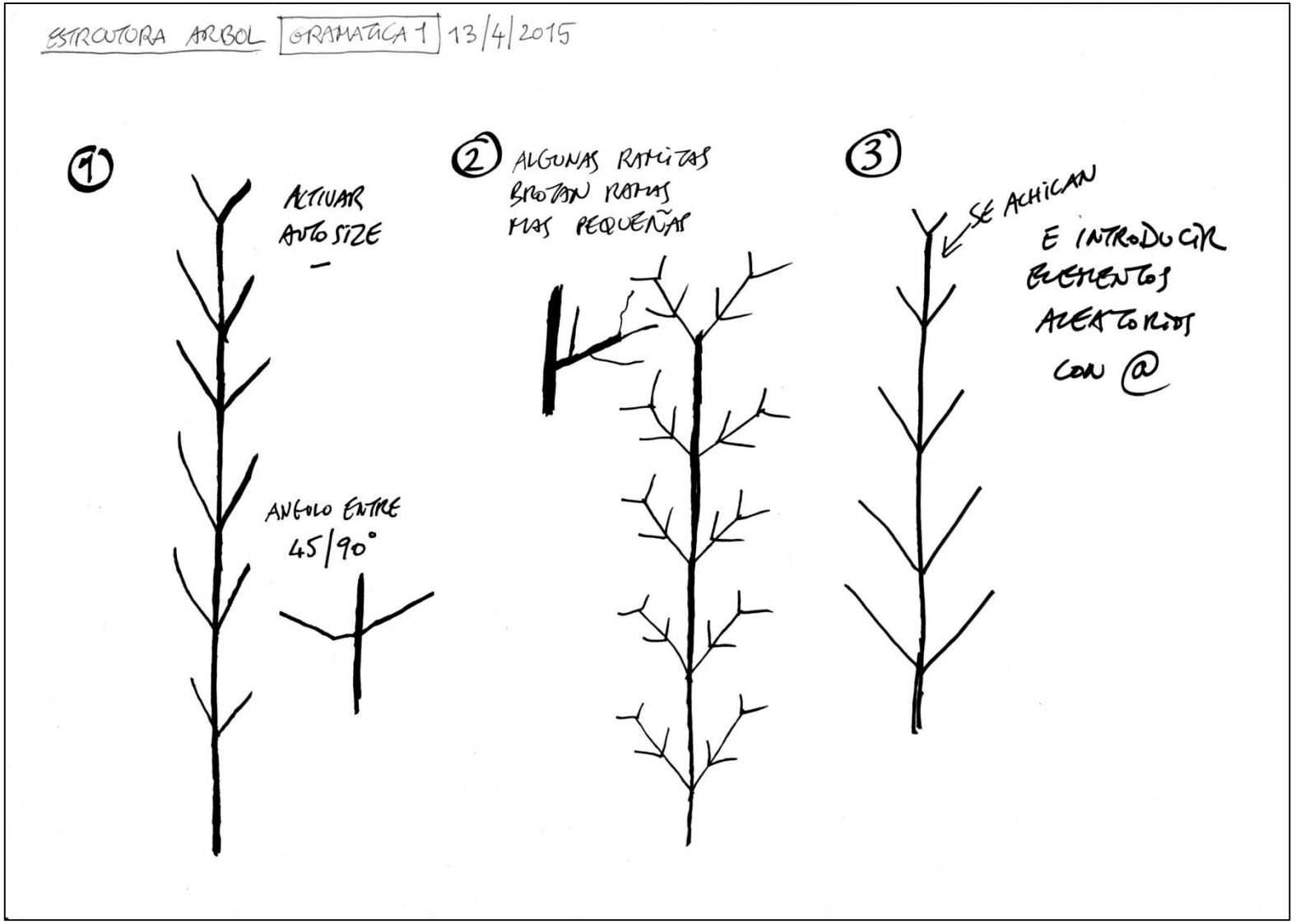


1 ELEMENTO Z QUE PRODUCE CONTANDO LAS RAICES (CUALQUIER SIMBOLO ANTES DE UN [SE VAN SUMANDO.



DESPUES DE CADA RAMA SE REGRESA A LAS RAICES ORIGINALES.

ESTRUCTURA ARBOL GRAMATICA 1 13/4/2015



(4)

RANDOM.

CON @

Y REGLAS

STOCASTICAS

(5) PASO

PARA OBTENER

MAYOR VARIEDAD,

UTILIZAR

3 SIMBOLOS,

P PARA EL TRONCO

a, b PARA RAMIFICAR

ASÍ HAY MAS OPCIONES

ALEATORIAS

PiiM INVESTIGACION IDEAS LSYSTEMS

1 DE MAYO 2015

① GENERAR CADENAS SIMMETRICAS PARA LAS FUNCIONES ALEATORIAS.

R → GENERA UNA CADENA ALEATORIA DE M CHARS

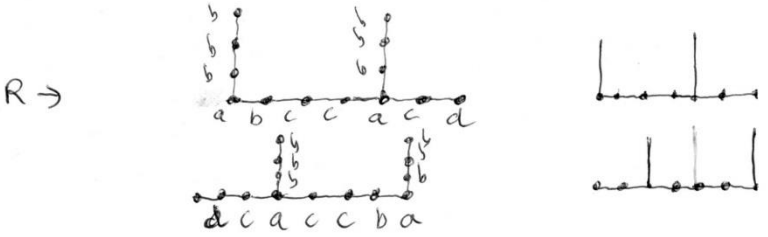
R=→ " " " " IGUAL AL R PREVIO

Rj→ GENERA UNA CADENA SIMMETRICA AL R PREVIO.

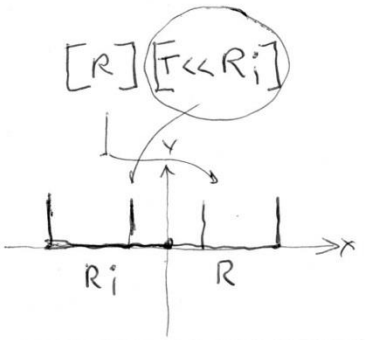
EJEMPLO: R=abccacd

Rj=dcaccba

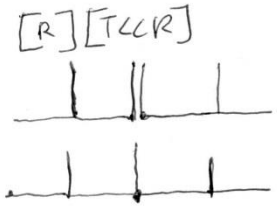
SI LUEGO PONEMOS UNA REGLA a=a[Yb b b]



X COMPLETAR LOS EFECTOS SE PUEDEN USAR LAS ROTACIONES



OJO!



ALTRA IDEA: UN SIMBOLO CHE RICEVE I SIMBOLI PRIMA DI CUI O DOPO DI CUI, INCLUSO IN UN ALTRO PUNTO DEL SISTEMA (PER ESEMPIO POSTO TRA DUE [ ] o { }

ALTRA IDEA:
UN SISTEMA SIMILE AI TAG-SISTEMS OUN SIMBOLO CHE FUNZIONA COME UN TAG-SYSTEM E

SIMBOLO m!

DETALLES SOBRE EL SIMBOLO m

PIIM
11/5/2015

am am am
1 2 3

PARA PODER MANEJAR LAS RAMAS CON m
OPPURE USARE IL SIMBOLO ϕ

a[m]a[m]a[m]
1 2 3

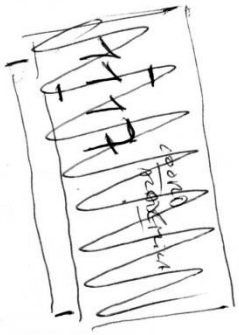
MEJOR UTILIZAR EL SIMBOLO
! QUE PONE A ϕ TODO:
- ANGULOS, SIMBOLO T, ETC.
- SIZE

a[bm!bm bm]a[bm!bm bm]
1 2 3 1 2 3

b!mmmmm bmmmm
a aa aaa aaaa

EL SIMBOLO !
DEBE BLOQUEAR LA
CUENTA HACIA
ATRAS

bm bm bm bm! bm bm
1 2 3 1 2 3



PiiM

1 DE MAYO 2015

COMO HACER EL LOOP DEL SIMBOLO

SUPPONIAMO LA REGLA 1 DI a

AXIOMA = a

2 1a = aYbbXab

MULTIPLIER ↑

STRING INT

SE CREA UNA FUNCION LOOPa(RULE, NUM)
QUE DEVUELVE UNA CADENA.

```
STRING LOOPa(STRING RULE, INT NUM)
{
    FOR L=0 TO MULTIa[0]
    {
    }
}
```

IN UPDATE RULE SI CREA UN ARRAY

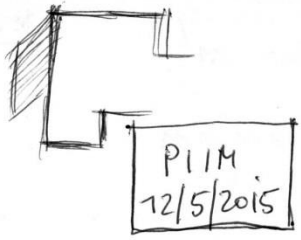
```
INT MULTI_a[10];
MULTI_a[0] = MULTI_a1.TEXT.PARSEINT();
```

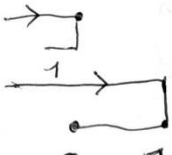
•NB: PARA QUE FUNCIONE EN LA REGLA DE a
DEBE APARECER 1a POR LO MENOS SI NO NO
HACE NADA.

a aF[Ybbb]F

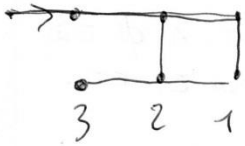
aF[Ybbb]FF[Ybbb]F

bm bm bm bm! bm bm





bm bm bm bm bm!
1 2 3 4 5

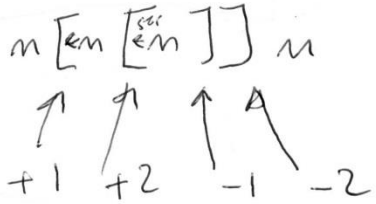


EL SIMBOLO ! YA ELIMINADO

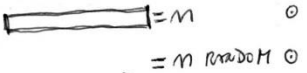
DEBE SALTARLE EL M [ ] M [ ] M [ ]

SE PUEDE HACER OTRO QUE ES (M,) CON EL MISMO CRITERIO, PARA CONTAR EN RAMAS. O UTILIZARLAS EN REGLAS DIFERENTES

CUENTA LAS [



M_STRING: CADENA DE LA CLASE LSYSTEM O REGULADOR QUE GUARDA QUE SE USA COMO M



INTERPRETE

NEWN += "a" ← ESTO SE ESTABLECE EN LAS OPCIONES, PUEDEN SER a, b, c, d, e, F Y TODAS LOS DEMAS

HASTA UNA M PUEDE PRODUCIR OTRAS M

NEWN += "F[+M!bmbm]"

SALE COMO UNA ESPECIE DE AUTOMATAS CELULAR. ESTUDIAR LAS POSIBILIDADES

PIIM 11 JUNIO

IDEAS X SYSTEMS

TAMAÑO ALEATORIO: FUNCIONA CON = TAMBIEN.

% = (SIZE/2 - RND(0, SIZE)) / 100 * RNDSIZE; ✓

PONER UN FLOAT ESTATICO DISEÑADOR.RNDSIZE;

! RESETEA EL VALOR DE M, PERO TAMBIEN SE PUEDE USAR X RESETEAR EL SIZE, EL ANGULO, ETC.

COSAS X ARREGLAR 13/6/2015

1) POSICIONAR CURSOR CON EL MOUSE X COMENZAR EN CUALQUIER LUGAR

2) FUNCION DRAW()

HAY UN ERROR EN) Y C NO CEE EL ANGULO.

- DISEÑADOR.CENTERX
- DISEÑADOR.CENTERY
- ORIGIN

FALTA IMPLEMENTAR DECAY

3) PONER UN VALOR QUE RESETEA EL ANGULO AL ORIGINAL, EL SIZE TAMBIEN

CENTER.X = ORIGIN/2 490

CENTER.Y = ORIGIN/2 415

PARSERMESSAGE

AXISON AXIS OFF

NOTA: "0" RESETEA LOS CONTADORES DE M

"." SIMMETRICO, SE AUTOBORRA SI ANTES TIENE R O S

"!" RESETEA ANGULOS Y SIZE

12 JUNIO

TAMAÑO SIZE CON FORMULA #

0% EL RANDOM NON SEMBRE A MOLTO.

DISEÑADOR.SIZEMODE = RND, FORMULA

DISEÑADOR.SIZE_FORMULA = 1, 2, 3 ....

DISEÑADOR.SIZEFORMULA1

SIZEFORMULA2

L SYSTEMS OTRAS REGLAS 12/5/2015 P1IM

(1) IMAGE RULE: REGLA QUE DEPENDE DE UNA IMAGEN (i)

(2) DATA RULE: REGLA QUE DEPENDE DE UNA BASE DE DATOS:

POSIBILIDADES: (SIMBOLO W)

a- LA REGLA CORRESPONDE A UNA TABLA DE DATOS DEPENDIENDO DE SU POSICION EN LA CADENA. ! RESETEA, COMO M.

b- LA REGLA DEPENDE DE UN SENSOR EXTERNO, QUE MANDA UNA SEÑAL, ESTA SE CONVIERTE EN DATOS

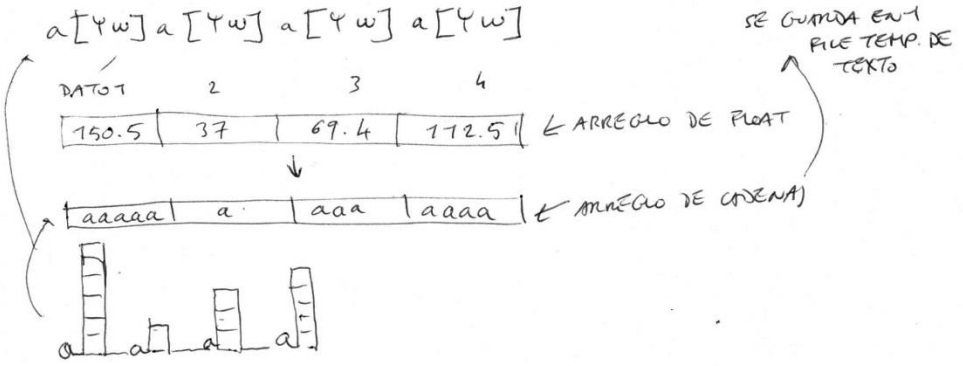
SE PUEDE HACER CON LEGO MINDSTORM

TIEMPO EN MIN.			
1	DATO 1 →	REGLA D1 →	LOOP 1
2	DATO 2 →	REGLA D2 →	LOOP 2
3	DATO 3 →	REGLA D3 →	LOOP 3
4	DATO 4 →	REGLA D4 →	LOOP 4
	ETC	ETC	ETC

C- LA REGLA CORRESPONDE A UNA BASE DE DATOS EN RELACION AL CICLO (LOOP) Y LA POS. EN LA CADENA.

EN TODOS LOS CASOS HAY QUE HACER UN ALGORITMO DE CONVERSION DE LOS DATOS EN REGLAS.

SERIA COMO HACER UN GRAFICO DE BARRAS.



PIIM 23/5/2015

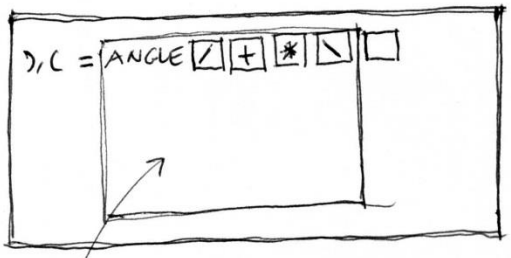
(1) UN SIMBOLO PARA LA ROTACION UTILIZANDO UNA FORMULA EN BASE AL VIEJO ANGULO:

P+a P)a P(a

ANGULO NORMAL — ANGULO CON FORMULA HORARIO — ANGULO CON FORMULA ANTIHORARIO

EN OPCIONES, PONER LA FORMULA

"),(" = ANGLE / 2 + 0.5



O UNA LISTA DESPLEGABLE CON VARIAS FORMULAS

(2) : Y | SE APLICAN TAMBIEN A LOS ANGULOS

! PONE EL VALOR ORIGINAL

(3) UNA SECUENCIA DE ROTACIONES DE ACUERDO A UNA TABLA DE DATOS. (EN GDESIGN 3.3).

(4) UN SIMBOLO PARA CONTROLAR EL TAMAÑO "t"

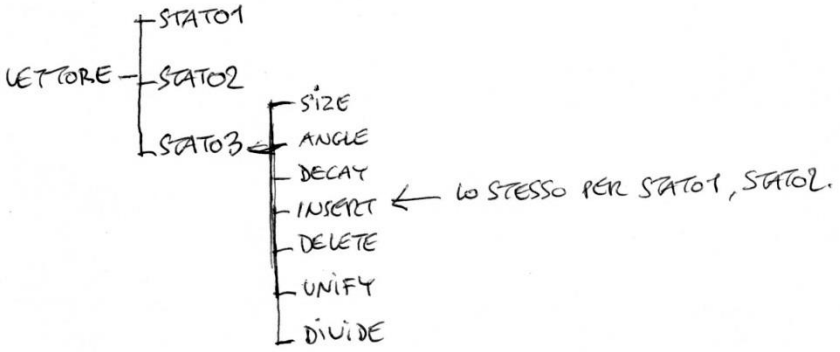
ta= ↙ CON FORMULAS O VALORES FIJOS
tb=
tc=
td=
te=
tf=

PTM 25/5/2015

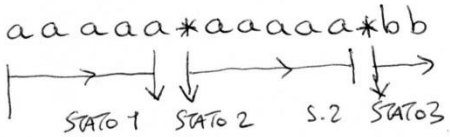
LSTSTEMS E MACCHINE DI TURING

1 CABEZAL, 1 LETTORE COME IN UNA TM.

IL LETTORE HA DIVERSI STATI E PARAMETRI



UN SIMBOLO PER CAMBIARE LO STATO: *

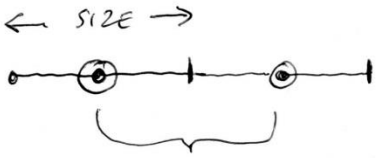


PUEDE SER UTILIZADO EN EL PROCESO NORMAL, TUTTO IL SWITCH/CASE DEI SIMBOLI CAMBIERIEBBE, BASTA METTERLO IN UN METODO.

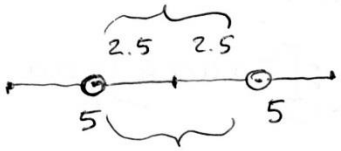
MSALAZAR@CONSORCIO.EDU.PE

CALCOLO DELL'OFFSET

PiiM 21/6/2015

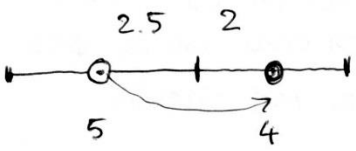


OFFSET: (OLDSIZE+NEWSIZE)/2 =

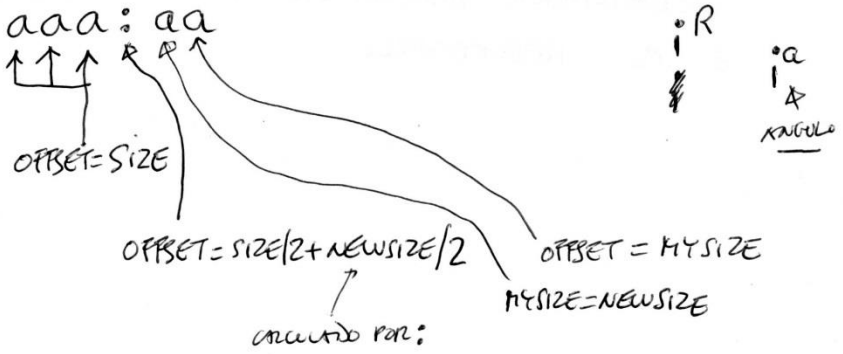


(5+5)/2=5

IL PUNTO ES CUANDO HAY UN : o |



(5+4)/2 = 4.5



BISOGNA FARE UN SIMBOLO CHE RESETTA EL SIZE ALL'ORIGINE !

PiiM 20 DE JONIO

TRABAJANDO EL SIMBOLO W

COME FUNZIONA:

SI LEGGE UN FILE .TXT CHE CONTIENE DATI.

IL FORMATO È:
1- NUMERO DEI DATI N
2- VACIO
3 DATO 1
DATO 2
DATO 3
...
DATO N

BISOGNA SISTEMARE QUESTA MECCANICA

MEMORIZZARE UN NUM_W

PONER UN SIMBOLO M MINUSCOLO QUE TOMA EL NUMERO DELL'ULTIMO W, UN POCO COME UN W=

↑ FORSE È MEGLIO FARE COSÌ

- I DATI SI CARICANO IN OPTIONS.

(OJO: BISOGNA CONTROLLARE CHE I DATI SI LEGGANO, SENO L'ARREGLO È VUOTO E DA ERRORE. SI PUÒ RIEMPIRE L'ARRAY DI 1 COSÌ NON DA ERRORE)

- I DATI SI CARICANO IN UN ARREGLO:

DATA[0]=7
DATA[1]=5
DATA[2]=12 ETC.

IL DATO SI ASSEGNA A UNA W A SECONDA DI CHE NUMERO DI W È:

a[1w]a[1w]a[1w]
↑0 ↑1 ↑2

DATA[0]=7 DATA[1]=5 DATA[2]=12 ETC.

- SE IL NUMERO DI W PASSA IL NUMERO DI DATI W=1 COSÌ NON DA OVERFLOW.

- ! RESETEA IL CONTATORE DI W.

PIIM 3 JULIO 2015

GDESIGN4.0

USO DE SIMBOLOS PARA INVERTIR ANGULOS Y CONTEOS, PARA CADENAS SIMMETRICAS, ETC.

¡ INVIERTE R, S R= abcd R¡ = dcba

∅ DESPUES DE n, w Y ? RESETEA EL CONTADOR

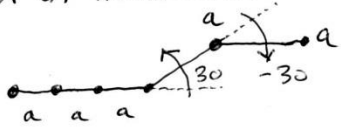
! INVIERTE EL ANGULO. POSIBILIDAD: !2 !3 !4

DUPLICA O TRIPLICA LA INVERSION.

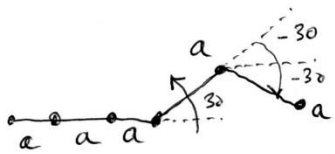
FALTA

PARA ESTO HAY QUE MODIFICAR LSYSTEM PARA QUE NO SE BORRE

aaaP+a!a



aaaP+a!2a



SIMBOLO PARA RESETEAR ANGULOS Y SIZE

OK HECHO 9/7/2015

#1 OPCION 1 SIZE

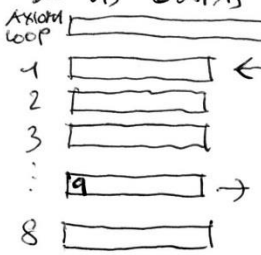
#2 OPCION 2 ANGULO

#1 #2 RESETEA PRIMERO EL SIZE Y LUEGO EL ANGULO.

PiiM 30/AGO/2015

AVANCES SIMBOLOS X GRAMATICAS GENERATIVAS

SIMBOLO 9. COMO ?2, USA LA CADENA DESARROLLADA EN UNA DE LAS ETAPAS ANTERIORES A CUANDO APARECE:



aqui el SIMBOLO "a" SE TRANSFORMA EN 9, O UN 19
ESTE 9 ES IGUAL A

LSYSTEM 1 HASTA LA REGLA 1
2 HASTA LA REGLA 2
3 HASTA LA REGLA 3
ETC.

- NOTA: COPIANDO LA CADENA TENEMOS QUE BORRAR EL PUNTO FINAL
- VA DESDE PUNTA ANA FINE

EN LA INTERFAZ, PUEDE SER ASI:

SIMBOLO 9. RECIBE LA CADENA FINAL EN LA ETAPA 1, 2, ETC.

SIMBOL 9 = LSYSTEM LOOP [1] [2] [3] [4] [5] [6] ... [9]

MEJOR CON BOTONES O CON UN SLIDER

- SE PONEN EN REGULADOR

S_rule =

LSYSTEM 1
LSYSTEM 2
LSYSTEM 3
LSYSTEM 4

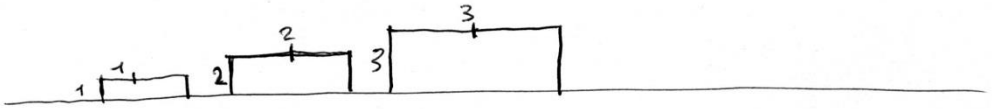
} UN ARREGLO LSSTRING_S

PONER PARA CADA SIMBOLO UNA SECUENCIA DE REGLAS EN LUGAR DE 1,2,3,4,5,6,7,8,9,10,

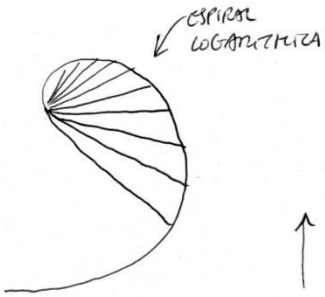
1,1,1,7,5,3,3,2,1,1 ETC

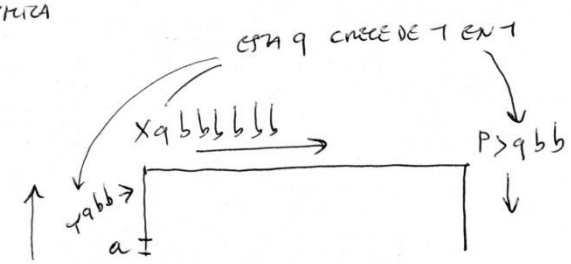
PiiM 4/10/2015

ESTRUTURA CON ELEMENTOS QUE CRECEN AUTOMATICAMENTE



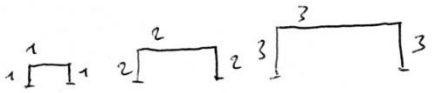
ELEMENTO Φ





T+[a][b+qbbXqbbbbbbP>qbb]

ESTO ES PARA HACER LA ROTACION EN ESPIRAL.



PROBAR CON CRECIMIENTO AUTOMATICO, DEL ANGULO + DE LA DIMENSION

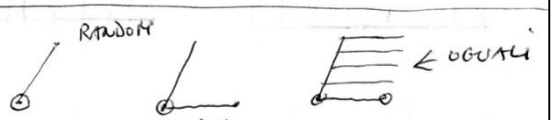
PIIM 9/10/2015

STRUTTURE CON SISTEMI L

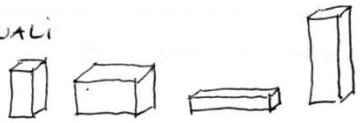
CATALOGO

PROGRAMMAZIONE CON SISTEMI L

① BOX ALEATORIO. R, S,), =



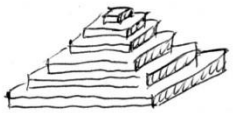
SERIE DI BOX CASUALI



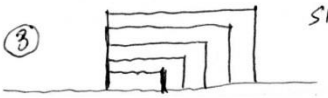
ETC.

② ZIGGURAT

9 SIMBOLO CHE CRESCE DI 1 IN 1

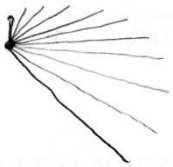


③



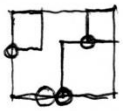
SIMBOLO 9

ANCHE CON ROTAZIONE SPIRALE LOGARITMICA



ETC.

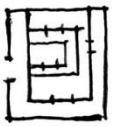
④ SUDDIVISIONE CON REGOLE DI CONTESTO USANDO R o S



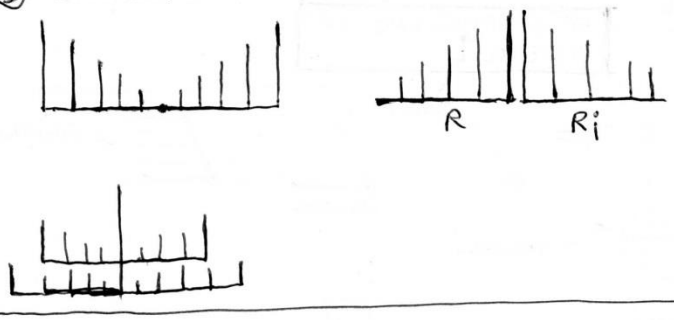
OPPURE UNA SEQUENZA COSÌ:

ETC.

LABIRINTO CON SISTEMI L



⑤ STRUTTURE SIMMETRICHE



⑥ REGOLE CON IMMAGINI E CON DATI

⑦ CELOSIAS

⑧ ESPERIMENTOS CON SIMBOLO M

anamamamam
1 2 3 4 5
↓ VECES LA REGLA.

Retro de la hoja anterior.

PARSER (1)

LMODEL → CLASE QUE SE ENCARGA DE DIBUJAR EL MODELO.

METODOS: PARSE LSYSTEM : DIBUJA SIMBOLO POR SIMBOLO

TIENE QUESTE PROPIEDADES:

POINT : IL PUNTO ACTUAL

NEWPOINT : IL NUOVO PUNTO CALCOLATO

LINE : LA LINEA TRA POINT E NEWPOINT

MYAXIS:

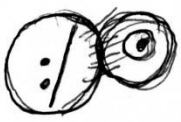
OBJAXIS:

POINT3D
AXIS
LINE 3D
CUBE

→ CADA UNO TIENE SUS METODOS
INIT(), MOVE(), ROTATE(), DRAW().
EXPORT(),

ESTO ESTÁ SOLO EN LA CLASE POINT3D.
EL CUBO ROTA CADA UNO DE SUS VERTICES CON POINT3D

DISEÑADOR → CLASE CON PROPIEDADES ESTATICAS CON TODOS LOS PARAMETROS DE DISEÑO



EL CENTRO

PIIM 2015

DATA RULE W

TERMINA CON ϕ EL CONTEO

EJEMPLO

abwabwabw

↑ / /

DATO ϕ 1 2 ...

EN UN ARCHIVO SE PONEN LOS DATOS

100
20
30
40
70
80
5
10
⋮

EN LA VENTANA OPCIONES
SE CARGA EL FILE MENÚ DATA

EL LSYSTEM SE CALCULA CADA W
(COMO EN EL CASO DE M) Y ESTO HACE
AVANZAR EL ARREGLO DE DATOS
COMIENZA CON ϕ

SIMBOLO R

R_ref → ARRAY DE CADENAS QUE DICEN QUE SIMBOLO ES EL REFERENTE,

R-LOOP → ARRAY DE INT QUE DICEN LA REGLA DE ESTE SIMBOLO

EJEMPLO:

EN EL LOOP 1 ES SIMBOLO ES "a" Y SE USA LA REGLA DE SU LOOP 3 ... ETC.

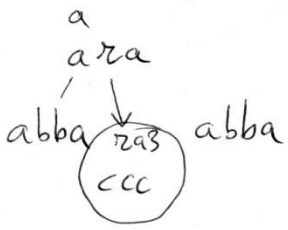
1 SIMBOLO DEBE TENER LA R EN SU REGLA, OPPURE SI IGNORA LA R QUANDO SI COPIA LA REGLA, PER EVITARE BUCLES INFINITOS

a

1 aRa

2 abba

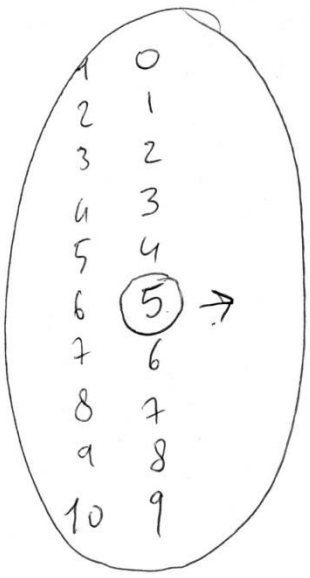
3 ccc



R = R_ref[CICLO]

1
2
3
4
5
6
7

REGLA LOOP 7



PiiM 8/10 2015

SIMBOLO S.

SE TRANSFORMA EN EL LSYSTEM DE UN CIERTO LOOP (OBVIAMENTE ANTES DE DONDE APARECE S)

AXIOMA = a

a1 a → a[P+aaaaaa]

a2 S → s[P+SSSSS]

LUEGO SE POSTPROCESA S Y DEBE SALIR

a[P+aaaaaa][P+a[P+aaaaaa] etc ...]

REGULADOR.STRING[n] CONTIENE EL LSYSTEM.

REGULADOR.NUM_LS CONTIENE EL LOOP QUE QUEREMOS EL LSYSTEM
↳ SE PONE EN OPTIONS

S[0] ~~CORRESPONDE~~ SE FIJA EN CICLO1 ES DECIR, S[n] ES

S[1] SE FIJA EN CICLO2 CORRESPONDE AL CICLO n+1

≠

SOLO SI NUM_S ES MENOR DE CICLO!

~~SOLO~~

S COMO R o S DEBEN SER PARSEADOS, SINO PRODUCEN AL CICLO SIGUIENTE!

---

ss[P+ss] a[P+a]a[P+a][P+a[P+a]a[P+a]]

PIIM 11-10-2015

LSYSTEMS Y MAQUINAS DE TURING

- LEGGE LA STRINGA COME SE FOSSE IL NASTRO DI UNA MACCHINA DI TURING.
- CAMBIA DI STADO A OGNI PARENTESI O A OGNI SIMBOLO, ETC.
- LA MACCHINA HA QUESTE AZIONI $\overset{-}{\rightarrow}$ $\leftarrow$

LEGGE
SCRIVE
TRASPORTA — CARICA UN SIMBOLO E LO INSERTA PRIMA O DOPO
NIENTE

IL "PACCO" CIOÈ IL SIMBOLO TRASPORTATO SI PASSA DA UNO STATO ALL'ALTRO, CIOÈ TRASPORTA ASSUME: PASSA o DEPOSITA

- CARICA
  ↳ CANCELLA IL SIMBOLO IN QUELLA CASELLA

PER ESEMPIO Z ϕ TRASPORTA
1 TRASPORTA
– DEPOSITA
C CARICA

EJEMPLO:

aabc[P+bbaa[ZcccP-b[ddd]bc]aP-aaa]bZc

ESTADO P1 [P2[→ ESTADO P2 ESTADO P3 P3]P2]→

BAJA A P2

P1, P2, P3 ETC POSSONO TRASPORTARE
CARICARE
DEPOSITARE

A COSA PUÒ SERVIRE QUESTO SISTEMA?
- A PARAMETRIZZARE CERTE FUNZIONI
- PER ESEMPIO A MODIFICARE DIMENSIONI E ANGOLI.

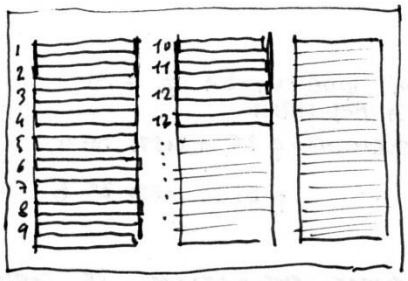
IDEE X LSYSTEMS **PiiM** 11-10-2015

FORMA DE PROGRAMAR CON PROCESOS RECURSIVOS.

① COMBINAR LIBREMENTE LAS REGLAS DE UN SIMBOLO DESDE UN REPOSITORIOS DE REGLAS EN UNA LISTA QUE DA LA SECUENCIA DE REGLAS, SI SON ALEATORIAS O ESTOCASTICAS

② COMPONENTES:

A - BASE DE REGLAS, X TODOS LOS SIMBOLOS, INCLUYENDO REGLAS ESPECIALES.



1-10 a
11-20 b
21-30 c
31-40 d
41-50 e

51-60 ESPECIALES
51 ⇒ a ⇒ a+a (ADD RULE)
52 ⇒ a ⇒ aa (DOUBLE RULE)
53 ⇒ a ⇒ COUNT RULE (COMO M)
54 ⇒ a ⇒ DELETE RULE 1

VALEN EL IGUAL. =

③ PARA CADA SIMBOLO SE HACE UNA LISTA DE LAS REGLAS QUE USA EN CADA CICLO.

$a = \{1, 1, 1, 5, 6, 3, 2, 1 \ldots\}$

LIST ITEM / CICLO: ϕ 1 2 3 4 5

EN CICLO ϕ LA REGLA 1
" " 1 " " 1
" " 2 " " 1
" " 3 " " 5 → ETC.

• SI HAY MAS CICLOS QUE REGLAS EN LA LISTA, EL SIMBOLO QUEDA IGUAL:

$a = \{3, 1, 1, 2, 5\}$

LIST ITEM / CICLO: 0 1 2 3 4

CICLO ϕ REGLA 3
⋮
CICLO 5 DEFAULT (a=a)

④ PARA LAS REGLAS ESTOCASTICAS SE PUEDE UTILIZAR UNA NOTACION ASI:

$a = \{3, 1, 1\text{-}2\text{-}4\text{-}6, 2, 5\}$

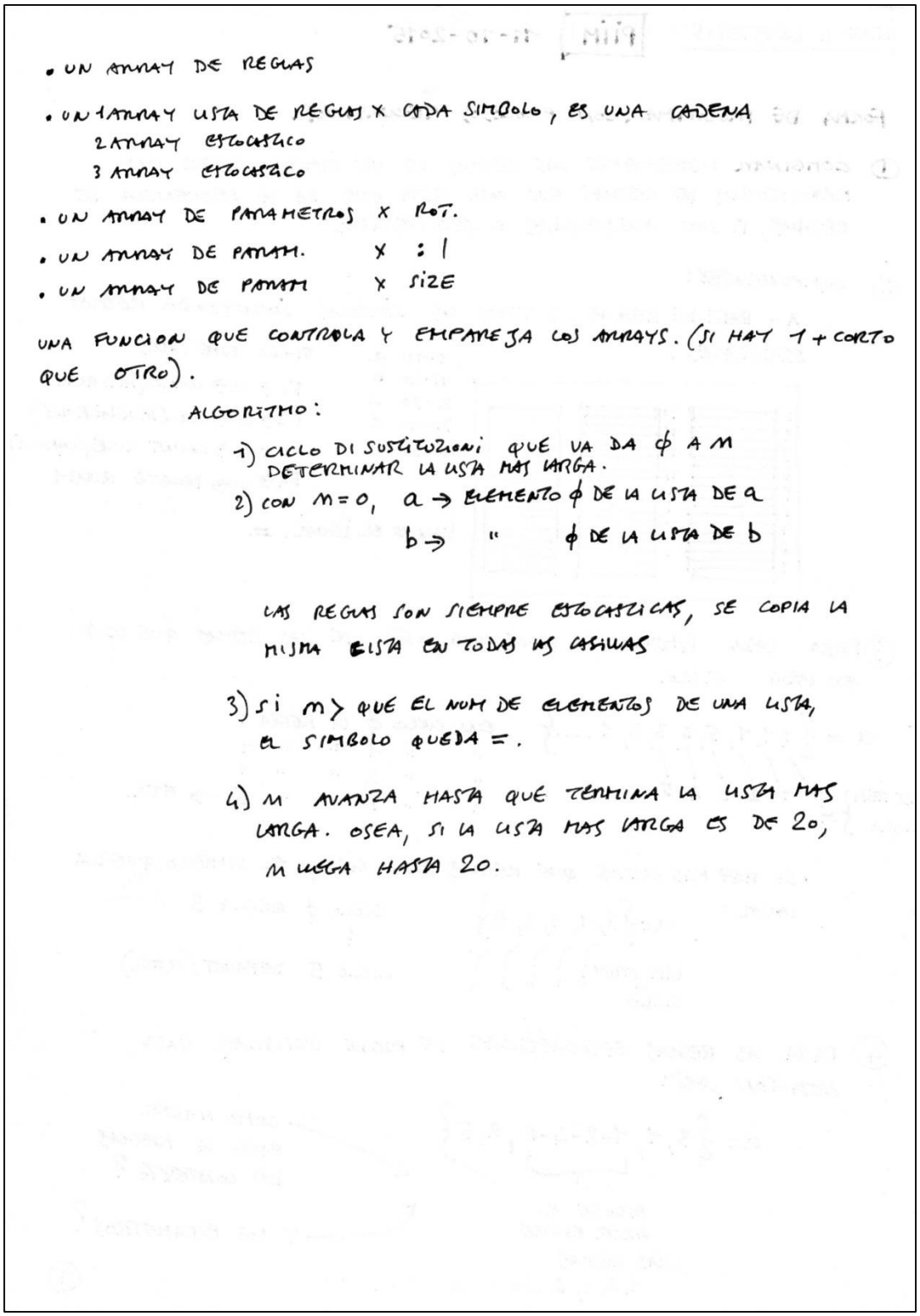
ESCOGE AL AZAR ENTRE LAS REGLAS 1, 2, 4, 6

COMO HACER PARA LA REGLAS DE CONTEXTO?

Y LOS PARAMETROS?

(1)

Retro de la imagen anterior



- UN ARRAY DE REGLAS
- UN 1ARRAY LISTA DE REGLAS X CADA SIMBOLO, ES UNA CADENA
  2 ARRAY ESTOCASTICO
  3 ARRAY ESTOCASTICO
- UN ARRAY DE PARAMETROS X ROT.
- UN ARRAY DE PARAM. X :|
- UN ARRAY DE PARAM X SIZE

UNA FUNCION QUE CONTROLA Y EMPAREJA LOS ARRAYS. (SI HAY 1 + CORTO QUE OTRO).

ALGORITMO:

1) CICLO DI SUSTITUZIONI QUE VA DA ϕ A M DETERMINAR LA LISTA MAS LARGA.
2) CON $m=0$, a → ELEMENTO ϕ DE LA LISTA DE a
   b → " ϕ DE LA LISTA DE b

   LAS REGLAS SON SIEMPRE ESTOCASTICAS, SE COPIA LA MISMA LISTA EN TODAS LAS CASILLAS

3) si m > QUE EL NUM DE ELEMENTOS DE UNA LISTA, EL SIMBOLO QUEDA =.

4) M AVANZA HASTA QUE TERMINA LA LISTA MAS LARGA. OSEA, SI LA LISTA MAS LARGA ES DE 20, M LLEGA HASTA 20.

IDEE X LSYSTEMS **PIIM** 17-10-2015

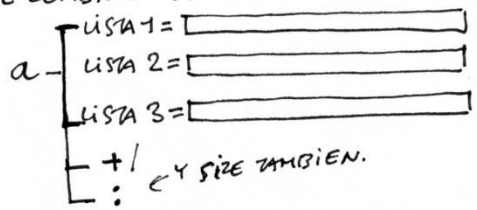
LUEGO SE VA AL MISMO PARSER

⑤ REGLAS DE CONTEXTO:

$a = \{1, a3b-[2]-+4a, \ldots\}$

a 3 b SI a ESTÁ ENTRE a Y b
[2] SI a ESTÁ ENTRE [ ]
+ a SI a ESTÁ ENTRE UNA ROTACION Y UNA a

ES POSIBLE TAMBIEN UNA NOTACION QUE COMBINE STOCASTICO CON ALEATORIO: LISTA 1 NORMAL
LISTA 2 ALEATORIA
LISTA 3 ALEATORIA
ETC.



⑥ PARAMETROS. ES FACIL, PONER UNA LETRA P, CADA SIMBOLO TIENE SU P. $a = \{1, P2, P4, 2, 1\}$

LAS REGLAS 2 Y 4 TIENEN UN PARAMETRO.

MEJOR: ES MAS FACIL CREAR REGLAS PARAMETRICAS CUYOS SIMBOLOS :, +, / ETC DEPENDEN DE UN PARAMETRO.

MENÚ PARA ASIGNAR PARAMETROS AGE, DISTANCE, POPULATION

CUANTOS SEGUIDOS HAY DE ESTE SIMBOLO

⑦ HASTE SE PUEDE HACER UNA LISTA DE VALORES X SIZE Y ANGLE

$a+/ = \{10, 15, 20, 45, 90 \text{ ETC}\}$

$a: = \{5\%, 3\%, \text{ETC}\}$

⑧ CIERTAS REGLAS PUEDEN SER SUBSISTEMAS.....

PiiM 11-10-2015

POST PROCESSING CON LSYSTEMS

ES UNA SUERTE DE TURING MACHINE.

UNA VEZ QUE TENEMOS UN SISTEMA, COMO ELABORADLO.
SE PUEDEN HACER VARIAS COSAS. ¡GENETICA!

(1) COMBINAR: LSYSTEM1 + LSYSTEM2. UN SIMBOLO DI 1 ~~CONECTO~~ Y UN SIMBOLO DI 2. LS1=aaaa, LS2=bbbb → LS3=abababab

VEREMOS EL PROGRAMA YA HECHO.

(2) FUSIONAR: UN SIMBOLO DI 1 + OTRO SIMBOLO DI 2 HACE UN NUEVO SIMBOLO DI 3

LS1=aaaa
LS2=bbbb
LS3=(a+b)c, (a+b)c, ETC.

(3) RESTAR: SI HAY UN SIMBOLO DI 1 = A UN SIMBOLO DI 2 SE BORRA

LS1=P+aaba
LS2=ccccbd
LS3=P+aaa ← SE BORRÓ d

(4) CONTRACT: DIMINUISCE LA CANTIDAD DE SIMBOLOS, CON VARIAS REGLAS aaa = aa ← FUSIONAR

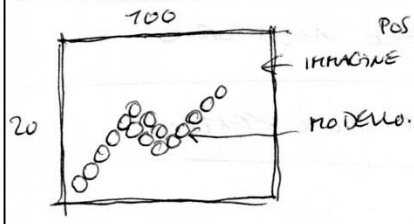
[P+bcdaaa]
↑ SE BORRA LA HOJA ETC.

(5) INSERT: INSERTA UNA RAMA O UNA SECUENCIA DE SIMBOLOS EN CIERTOS PUNTOS DETERMINADOS: RAICES, HOJAS, ANTES DE UNA ROTACION, ETC.

PIIM IMAGERULE 15/11/2015

COME CALCOLARE IMAGE RULE

CALCOLANDO LA POSIZIONE NELLA ~~DEL~~ STRINGA

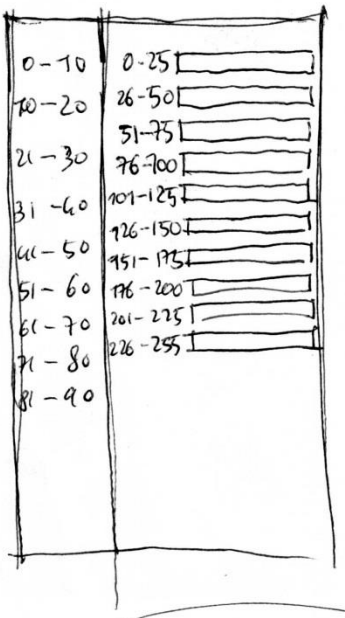


POS IN STRING = 200

1. METTERE L'IMMAGINE E IL MODELLO IN POSIZIONE
2. COMPARE IL PARSER COME SE FOSSE UN SISTEMA NORMALE.
3. NELLA FUNZIONE SETSPECIALSIZE O DISEGNANDO UN'ALTRA FUNZIONE SOSTITUIRE QUESTO ELEMENTO.
   - METTERE UN'OPZIONE GENERATE IMAGERULE.

   MYCSISTEM ⟶ MYCSISTEM_IMAGE (UNA NUOVA CARTELLA CHE PER SI PROCESSA NORMALE).

   SELECCIONAR SI ES X a, b, c, d, e,



OPTION: GENERATE IMAGERULE CSYSTEM

- CONTROLAR UN FLAG QUE ESTABLEZCA SI ES QUE SE ABRIÓ LA IMAGEN.

3908.16

823,685 CANVAS SIZE

Anexo 2

La virtualidad aumentada: procesos emergentes, arte y medios digitales

Sumilla

Últimamente se publican muchos ensayos críticos sobre internet, las redes sociales y los nuevos medios en general. Del arte digital se critica el lenguaje barroco y repetitivo de sus caprichos biogenéticos sin sustento: emergencia e interactividad, -aspectos clave de su estética- son simulacros, como evidenciaremos mediante la comparación con lo pintoresco romántico. Es la equivocada relación entre naturaleza y arte, pilar de este kitsch romántico e informático, que debe ser corregida en sentido estético y computacional. Por otro lado, es necesario llevar lo digital más allá de la abstracción de las simulaciones hacia lo que podríamos definir como la 'virtualidad aumentada'. Para esto, se discutirán las relaciones entre naturaleza, artista, obra y proceso; desde el punto de vista digital, se pondrá en tela de juicio la posibilidad misma de la complejidad y emergencia tecnológica, evidenciando tanto los problemas epistemológicos generados por las interfaces como el conflicto planteado por la contraposición entre la libertad indispensable en la interactividad y el determinismo impuesto por los algoritmos. Para enfrentar estos problemas, el volver la mirada al legado visual de las culturales ancestrales (en nuestro caso lo precolombino) puede ser un aporte esencial, como en cierto modo intuyeron las vanguardias artísticas. Estas consideraciones permiten implementar estrategias para poner los medios digitales realmente al servicio de la libertad y creatividad de los usuarios. En este sentido, concluiremos mostrando que los procesos de sustitución recursiva como los sistemas-L, basados teóricamente en las gramáticas generativas, permiten desarrollar herramientas digitales abiertas y creativas, donde la naturaleza, la tradición, el arte y la tecnología pueden trabajar en modo armónico y coherente.

Abstract

In the last couple of years many authors have written about the problems of internet, Facebook and new media as a whole. On digital art, the repetitive and baroque aesthetics of its biogenetics artworks is also widely criticized. Emergence and interactivity –both keywords of the digital artistic rhetoric- are just simulacra, as the comparison with the romantic picturesque will clearly show. The jeopardized nature and art relationship, column of the romantic and technological kitsch, must be corrected, regarding both its aesthetic contradictions and computational problems. By the way, it is paramount to pull the digital out of the abstractions of simulations, into a new field that could be defined as 'augmented virtuality'. To achieve this, we will discuss the relationships between nature, art, artworks and processes; from the digital media point of view, the very possibility of complexity and technical emergence will be taken into consideration, along with the fight between liberty as requested by interaction and the algorithmic determinism and the epistemological problems of interfaces. Here the deep meaning of traditional and ethnic visual culture (in our case, the pre-Columbian) will be harvested, following the statements of the historic avant-garde. All this brings the possibility to implement strategy to set digital media into a new social, creative and democratic mood. As a conclusion, we will show that string rewriting systems (such as L-Systems), grounded on generative grammars, can develop digital tools truly opened and creative, where nature, tradition, art and technology can work with sharpness and coherence.

Palabras claves
Complejidad, emergencia, pintoresco, investigación artística, arte generativo.

Keywords
Complexity, emergence, picturesque, artistic research, generative art.

Introducción

Desde hace unos años en la arquitectura, en el diseño industrial y en el arte se ha difundido el gusto por las formas generativas, algorítmicas o paramétricas: no hay *blog* de arquitectura, en vena de posmodernidad, que no esté saturado de diseños complejos, tanto más repetitivos cuanto más pretenden ser raros y originales (Salingaros, 2014; Silber, 2009). En efecto, después de la exaltación por los fractales o la vida artificial, las artes digitales parecen estancadas y pasadas de moda. Todo esto tiene principalmente dos causas: el vacío teórico en la metodología y en la práctica artística y una cierta pobreza cultural en el diseño de las herramientas digitales, que apuntan más a la productividad que al desarrollo de la creatividad y de la reflexión crítica de los usuarios. Entonces la estética y las ciencias de la computación deben desarrollar conceptos y mecanismos más adecuados para encontrar (o reencontrar) la energía cultural y social del arte y de los nuevos medios.

Una posibilidad es renovar el encuentro de los procesos digitales con los procesos naturales, cuya esencia se pierde en la abstracción matemática de las simulaciones. En este sentido, el arte, las culturas visuales y las técnicas tradicionales, herederas de antiguas sabidurías, pueden dar directivas interesantes para el potencial creativo y pedagógico de la computación. Aquí una cuestión esencial es evidentemente la difícil relación entre lo natural y lo artificial. La irregularidad y el orden, la espontaneidad de lo orgánico y el método de los algoritmos conforman dialécticas opuestas que generan muchas de las dificultades en las que se debate el arte digital, precisamente en lo que se refiere a la complejidad, a la emergencia y a la interactividad.

Nuestro principal objetivo[*] es explorar la relación entre naturaleza, arte y computación, apuntando específicamente a las interfaces y a las gramáticas generativas

[*] Este artículo se ha desarrollado en el marco del proyecto de investigación *Algoritmos y gramáticas generativas para el estudio de los patrones del diseño precolombino*, ganador del 1 Premio de Investigación Interdisciplinaria del Consorcio de Universidades de Lima, Perú en 2015.

de la forma, donde hay cuestiones estéticas, metodológicas y tecnológicas que todavía quedan desatendidas.

Desde la estética, trataremos la relación entre lo sublime, lo bello y lo pintoresco, contrastando el romanticismo con la posmodernidad, partiendo desde Burke, Kant y Addison para llegar a Derrida, Eagleton, Costa y Vattimo. Estos conceptos, conflictivos ya en la época romántica, hoy lo son mucho más porque sus contradicciones están amplificadas en los medios digitales; así que, siguiendo a De Prada (2009, págs. 63-65), se hará un examen de lo pintoresco en relación a la topología, a la matemática y al uso del ordenador.

Desde el punto de vista metodológico, se quiere demostrar que los medios digitales deben utilizarse escapando de la burbuja virtual que ellos mismos generan; en este sentido hablaremos de la virtualidad aumentada y del papel del arte y de las técnicas manuales tradicionales. Solo así es posible evitar la superficial fantasmagoría barroca que, como han anotado Salingaros y Silber, caracterizan el *kitsch* digital y la pobreza de sus aplicaciones en la arquitectura, en el arte y en la educación.

En cuanto a la estructura de las tecnologías y de las herramientas digitales, se mostrarán cuáles son sus límites y debilidades en referencia a la complejidad y a la emergencia. Por ejemplo, el hecho que éstos solo toman en cuenta los aspectos productivos, o el problema del conocimiento, que explica por qué artistas y educadores no consideran al *software* y a las interfaces en su complejidad epistemológica y semiótica. Buscando posibles soluciones a estos problemas, se hará una breve introducción a las gramáticas de la forma y a los sistemas-L, tecnologías que permiten integrar la sensibilidad artística, la sabiduría de la manualidad y la eficiencia creativa de las máquinas.

La emergencia en el arte y en la tecnología

Para comenzar, tenemos que entrar en el marco general de la complejidad y prioritariamente en lo emergente, que concentra sus problemas puntuales. Emergente designa algo que nace a través de la interacción sistémica, sin un diseño predeterminado, al modo de las formas naturales, como los panales de abejas o las nubes. Como ya estaba entendido en la estética de lo pintoresco y de las vanguardias históricas y como ahora pretende el arte digital, lo emergente (espontáneo y eventual) se contrapone a lo

consciente y racional y pone en discusión la tiranía de la obra de arte y del artista. Como ha escrito De Prada a propósito de la exposición *Parallel of life and art*[7]:

"*Los autores de la exposición sugerían que la forma artística, en lugar de ser el producto de un proceso consciente de diseño, debía ser (al igual que los montajes y objetos encontrados surrealistas) el producto de la intuición y la selección*". (De Prada, 2009, págs. 83-84)

Fig. 1. Imagen de la exposición *Parallel of Life and Art*, 1953.

Quizás han sido Deleuze y Guattari quienes mejor han explicado el sentido rizomático, fractal, híbrido y en permanente evolución de lo emergente:

"*Ya no se trata de imponer una forma a una materia, sino de elaborar un material cada vez más rico, más consistente, capaz por tanto de captar fuerzas cada vez más intensas."* (Deleuze y Guattari, 2004, pág. 333).

Concretamente, lo emergente implica tres contextos o dinámicas: el de la libre autoorganización de la autopoiesis, el del arte y el de la interacción tecnológica que hoy pretende darle instancia. Sin embargo, en seguida veremos que el arte -por su individualismo- y lo tecnológico -por su determinismo algorítmico y por la opacidad de sus procesos- pueden simular lo emergente solo si recurren al azar.

[7] *Parallel of Life and Art* fue una exposición organizada por artistas (Eduardo Paolozzi, Nigel Henderson), arquitectos y críticos (Peter y Alison Smithson, Reyner Banham) en 1953 en Londres, con el intento de relacionar la naturaleza, la ciencia, la fotografía y el arte de vanguardia.

El problema de lo pintoresco y del azar

El azar, en cuanto cuestionamiento de lo bello, de la ciencia y de la tecnología, de la autoría y de la originalidad, es un tópico importante en la estética de lo pintoresco y de las vanguardias históricas, especialmente Dada, el Surrealismo y el Futurismo[8], y que posteriormente se encuentra en el arte informal, en el arte conceptual, en el *Land Art* y en Fluxus[9]. Como Stubrin ha señalado:

"*Desde el punto de vista de la biología –que es la ciencia que más se ha abordado en este trabajo– podemos rastrear el reconocimiento del uso de un método basado en la aplicación continua de una técnica en la que el azar produce más de lo que evita; esta forma experimental de proceder es igual a la que utiliza el arte que viene sufriendo, desde hace más de medio siglo, transformaciones que dificultan su conceptualización*" Stubrin (2013)

Pero se abusa en el uso del azar como estrategia liberadora, pues en realidad no permite generar la dinámica diversidad cuya misteriosa coherencia caracteriza lo auténticamente emergente. Por esto Shaftesbury y Addison apostaron por la supremacía de la complejidad natural sobre el orden del arte y de los artistas; supuestamente, lo pintoresco resuelve la unión entre lo sublime (la naturaleza, la irregularidad) y lo bello (lo regular). Pero, como señala Prada:

"*... el nervio vital de todo efecto radica en la necesidad de la estructura, la imposibilidad absoluta de desplazamiento; aquí en cambio, el Arte* juega *con la apariencia de lo irregular*" (2009, pág. 25).

En efecto, la definición de arte como obra única y producto individual del artista no permite el orden emergente (sistémico y complejo), pues el artista está atrapado en su libertad particular. El libre juego de las facultades intelectuales del artista (el genio kantiano) es un simulacro de la complejidad emergente precisamente por esta libertad, ya que sin vínculos con lo real, sus facultades intelectuales se combinan por el arbitrio y el azar. Conscientes de esto, Dada, Surrealismo y Futurismo rechazaron lo bello, la mímesis,

[8] Véase *Cómo hacer un poema dadaísta* de Tzara, el cadáver exquisito surrealista, el *Intonarumori* del futurista Russolo y el *Ready Made* de Duchamp.
[9] Por ejemplo Pollock con el *dripping* o Dubuffet con el *art brut*.

el orden y el método a favor del azar, pero sin resolver el problema de fondo de lo emergente. En seguida hablaremos de dos aspectos puntuales de este problema: la relación entre lo bello y lo sublime y entre vanguardia y arte primitivo.

Lo emergente, lo sublime, lo primitivo y el arte como proceso

En efecto, la contradicción entre lo bello y lo sublime es insanable[10], porque lo bello banaliza lo sublime en tanto que recorta su radical inconmensurabilidad y apertura. El rechazo de las vanguardias por la *mimesis* y la belleza clásica, sin embargo, no logra eliminar los parámetros románticos, primero porque al solo negarlos se reafirma el poder de su existencia, y luego porque al mezclarlos con el azar se regresa al libre juego de las facultades intelectuales del genio, al grado cero de una estética mucho más estricta porque no se percibe como tal.

Así mismo, la apropiación del arte africano, precolombino (fig. 2) o tropical en el intento del arte occidental de recuperar la complejidad emergente a través de lo mítico, ancestral y subconsciente, no llega a incluir su principal valor: el mismo hecho de no auto considerarse arte, y sin entender que sólo lo puede ser en los términos occidentales[11]. En otras palabras, en esta apropiación, el misticismo, lo irracional y lo primitivo son inconsistentes porque no llegan en profundidad, son solamente un artificio del arte para salir del paradigma moderno y romántico.

Por otro lado, la emergencia no se expresa en una obra, pues es esencialmente comprensible dentro el proceso de un sistema dinámico. En este sentido las obras de arte digitales deben resolver la relación entre el artista y el espectador: el artista puede experimentar la emergencia en el proceso creativo, pero el espectador no, porque recibe la obra como producto acabado. Esto también sucede cuando se trata de obras de arte o de herramientas interactivas (piénsese a la lógica subyacente en programas aplicativos como Photoshop o 3DStudioMax), como explicaremos en seguida.

[10] La característica relevante de lo sublime (Kant, 2004) es que rebasa la escala humana, es inconmensurable, en el sentido dinámico (de la fuerza) y matemático (de la magnitud). En ambos casos, dimensiones que no pertenecen al arte y menos a la tecnología. Como bien explicó Derrida (2001), el problema de lo pintoresco es que, como lo colosal, se situaría en el borde, en el límite entre lo bello y lo sublime, entre la obra (*ergon*) y el marco (*parergon*). Pero el estar en el límite es algo dinámico que se puede dar solo en un proceso.

[11] La crítica poscolonial define esto como "eurocentrismo".

Fig. 2. La cultura precolombina nace de la relación entre la naturaleza y la cosmovisión, lo que construye un campo complejo definido por múltiples dimensiones: productiva, sagrada, utilitaria, ritual, espacial y climática (Crousse, 2011). Catarata y Líneas de Nazca, estructuras geológicas y huaca Pucllana de Lima. Fotos de la autora.

El problema del determinismo tecnológico

Visto desde el punto de vista computacional, el *kitsch* pseudo emergente se debe a tres clases de dificultades: una estructural, que se debe a la naturaleza de los procesos computacionales y a la ingeniería de *software* que determina su uso profesional, creativo y educativo, una propiamente informática que se manifiesta en el diseño de las interfaces, del *software* aplicativo, de los lenguajes de programación y de los sistemas operativos, y la estética, embebida en las herramientas informáticas, que ya hemos tratado más arriba.

El problema estructural de los algoritmos y de los procesos computacionales es que son lineales y determinístas. Fractales, vida artificial e inteligencia artificial son simulaciones de la complejidad natural, su irregularidad es impredecible solo en apariencia; es decir, ciertamente son complicados, pero complicado no equivale a complejo[12]. Entonces Sommerer y Mignonneau podrían equivocarse cuando, comentando su proyecto artístico con la vida artificial *Life Spacies* afirman que:

[12] Por ejemplo, la dificultad de visualizar un proceso recursivo.

"*Al interactuar con estos sistemas, los usuarios los vuelven cada vez más complejos y les otorgan rasgos propios de sistemas complejos como variedad y dependencia, irreductibilidad, ruptura de la simetría, adaptación y organización, mutación y evolución, expansión de la diversidad, reacción ante los vecinos y el control externo, exploración de sus opciones y reproducción*" Sommerer y Mignonneau (2009)

Así pues, la complejidad emergente natural se falsifica o bien utilizando funciones aleatorias -con lo cual se pierde su significado sistémico originario- o predisponiendo la irregularidad como variantes del algoritmo principal -donde lo emergente es diseñado de antemano, como la inteligencia artificial de la habitación china de Searle-. Bajo este punto de vista, lo pintoresco y el *kitsch* digital implican ciertamente engañar al espectador.

El problema propiamente informático –la interactividad- se debe a que todo proceso computacional es determinista, entonces un auténtico proceso emergente tiene solo dos posibilidades: la inclusión de la naturaleza con el *physical computing* y la inclusión de las irregularidades de los mundos personales de los usuarios (una auténtica multiplicidad que se auto organiza). En ambos casos el mecanismo subyacente es la interactividad, pero la computación es incompatible con la auténtica interactividad, porque los efectos eventuales de la libre interacción debilitan su exactitud y sobre todo porque la interactividad supone la igualdad entre los participantes de la acción comunicativa (Habermas, 1990). El concepto clave de esta igualdad es el de la "justicia epistémica"[13], pues sin ésta no hay posibilidad de retroalimentación entre los componentes de un sistema, condición necesaria de su complejidad emergente. Lo que sucede en el sistema de los medios digitales es que las interfaces impiden el acceso de los usuarios a las estructuras epistemológicas y a los conocimientos embebidos en el *software*, que así queda como una caja negra (Flusser, 2000). De esto y por los excesos de los hipermedia y de internet (Carr, 2008), por la lógica consumista y hedonista (Oppenheimer, 2004) y por la banalidad de la sociedad de la información (Kurz, 2003) se aprovechan los monopolios corporativos (Morozov, 2011). Si el arte pretende recuperar su significado a través de la ciencia y de la tecnología, de esta forma obtiene el efecto contrario.

[13] Un término de la crítica poscolonial, que significa igualdad, libertad y ecua distribución del conocimiento, trasparencia de los procesos tecnológicos y respeto de las identidades culturales por los lenguajes tecnológicos.

Lo virtual aumentado

Resumiendo, los fenómenos emergentes y complejos son incomputables por efecto de contradicciones estéticas y por el determinismo algorítmico. ¿Entonces, la naturaleza, la emergencia, el arte y la computación son incompatibles? Como ya pensaban Shaftesbury e Addison, la naturaleza y la realidad tienen prioridad sobre el arte (y la computación), así que cualquier solución debe respetar este principio general y considerar las cuestiones que hemos señalado anteriormente. Por lo tanto hay que integrar propuestas estéticas, metodológicas y tecnológicas.

La solución estética consiste en resolver las contradicciones entre lo bello y lo sublime, lo regular y lo irregular, el diseño y la emergencia, insolubles en la obra de arte, en la dinámica creativa y productiva (el proceso). Éste puede tener una esencia abierta e irregular, por darse en el tiempo y (considerando la justicia epistémica) en modo interactivo; por otro lado, lo bello y la regularidad se encuentran en las múltiples instancias en que el mismo proceso (que es abierto) puede finalizar. Hay que observar dos aspectos más: primero que el orden y la belleza son posibilidades o acontecimientos eventuales del proceso emergente y que cada obra es una instancia con orden propio, aunque su origen se encuentre en potencia en el proceso que la ha generado. Y segundo, en cuanto produce múltiples instancias, el proceso es una herramienta, un metalenguaje, una metaforma: obra para hacer obras, lenguaje para desarrollar lenguajes. En este sentido, desde el punto de vista de los medios digitales (pero no exclusivamente), la solución del problema estético va en el sentido de la autoría distribuida del arte, lo que le devuelve su función práctica y su relevancia social.

Desde el punto de vista metodológico, una posible solución, que explica el significado de virtualidad aumentada, es el uso de las herramientas digitales en un momento preciso de la práctica artística y pedagógica. En el proceso artístico, lo digital debería comenzar y desembocar en el mundo real, es decir, participando en este proceso como uno de los elementos de un sistema integrado que incluye varios aspectos[14], como la ingeniería de *software* y el dibujo a mano alzada. Pallasmaa ha escrito:

[14] La complejidad y lo emergente aparecen en el sistema de elementos que constituyen las etapas de la historia del proceso, una idea que fue anticipada por el concepto de «formatividad» de Pareyson (1998) y desarrollada por la teoría de la forma como *memory storage* de Leyghton.

"*A dibujar a mano con lápiz o bolígrafo, la mano sigue los contornos, las formas y los patrones del objeto, mientras que dibujar con el ratón y el ordenador, la mano normalmente seleccionas las líneas de un conjunto dado de símbolos que no tienen una relación analógica –y en consecuencia, tampoco háptica o emocional- con el objeto de dibujo*."(Pallasmaa, 2012, pág. 108).

Fig. 3. Según Pallasmaa (2012, págs. 99-100): Dibujar es observar, expresar, recibir y dar contemporáneamente, donde se condensa el espacio y la materia con la percepción, pensamiento e imaginería de quien dibuja. Fotografía y dibujos de la autora.

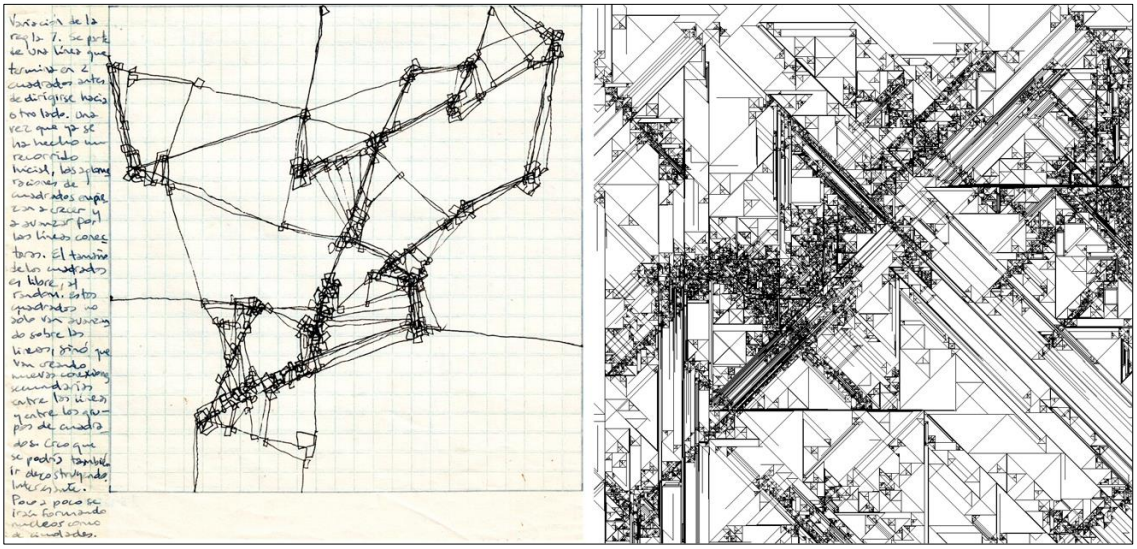


Fig. 4. Dibujo a mano alzada en base a reglas generadoras, de la autora (izquierda). Dibujo con algoritmos de vida artificial, del autor (derecha).

Desde el punto de vista del problema tecnológico del determinismo y del azar, dos posibles soluciones serían mejorar las funciones que ofrecen las herramientas digitales y el modo en cómo se ofrecen, y optimizando el diseño de la interactividad. En ambos sentidos se trata de la justicia epistémica, que finalmente se puede conseguir mediante una correcta filosofía tecnológica y de las interfaces. Una filosofía tecnológica dirigida a la libre y democrática colaboración entre usuarios, como la que caracteriza el *software*

libre, el *open source*, el *copyleft*, el *aesthetic computing* (la documentación, el código bien escrito) (Fishwick, 2006) y una suerte de *slow computation*. Con respecto a las interfaces, estas constituyen una barrera entre el usuario y los procesos computacionales subyacentes, y por lo tanto, deberían ser repensadas, hasta el extremo de eliminarlas, a pesar que esto pueda dificultar la productividad (pero ad *aspera astra*).

Estas filosofías tecnológicas conciben lo digital como herramienta para hacer herramientas, idea que empieza a germinar entre los usuarios de comunidades del arte, la arquitectura y el diseño digital gracias a lenguajes de programación innovadores como Processing o Grasshopper. Sin embargo, se necesitan también instrumentos intermedios: *middleware*, algo entre la aplicación (totalmente amigable, pero cerrada) y el lenguaje de programación (totalmente abierto, pero esotérico) y *middleart*: algo entre la obra de arte y la herramienta. Aquí vale la pena gastar algunas palabras acerca de las *shape grammars* y los sistemas-L.

Gramáticas generativas, gramáticas de la forma y sistemas-L

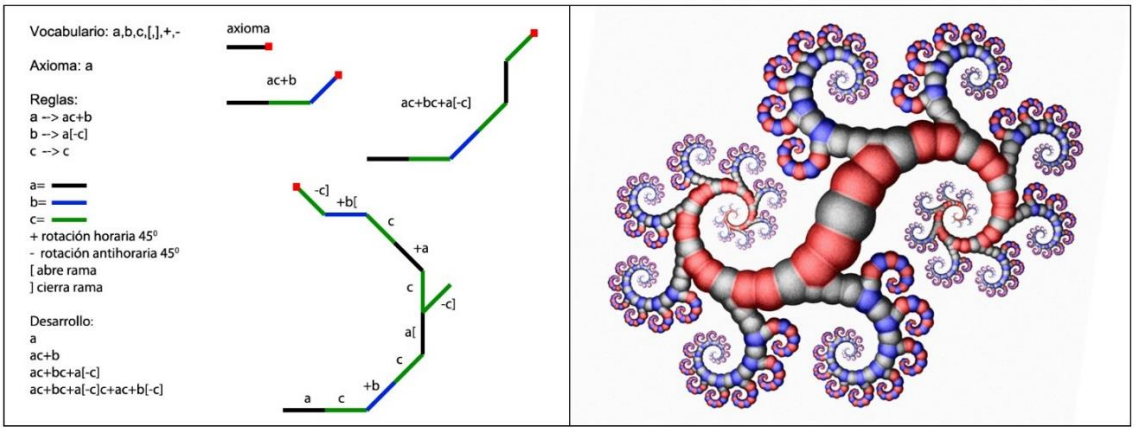


Fig. 5. El principio de las *shape grammars* y de los sistemas-L es simple, consiste en sustituir un conjunto de símbolos con otros, de acuerdo a una serie de reglas. Luego los símbolos finales son traducidos por un módulo llamado *parser* en elementos (hasta con miles, como en el *rendering* de la derecha), de un dibujo 2D o 3D o en notas musicales, o en elementos arquitectónicos. Ilustración del autor.

Las gramáticas de la forma (Stiny y Gips, 1971) son procesos basados en sustituciones recursivas de símbolos alfanuméricos (fig. 5) de acuerdo a un conjunto de reglas formales. Lo interesante es que pueden generar estructuras complejas y procesos tecnológicos emergentes porque el diseño de las reglas es interactivo y abierto, inclusive a múltiples usuarios y porque se pueden conectar con el mundo real a través del *physical computing* y porque pueden integrar funciones aleatorias (cuestionables pero no inútiles).

Pero la ventaja principal de las gramáticas de la forma y de los procesos de sustitución recursiva, en cuanto al diseño de las interfaces (fig: 6), es que sus principios, reglas y procedimientos pueden ser totalmente trasparentes: el funcionamiento del proceso y las relaciones entre sus componentes son editables por el usuario (que en caso del *software open source*, además, puede enriquecer el sistema con su propio código).

Sin embargo, el estado actual de desarrollo de estos sistemas delimita su uso a la generación de formas autosimilares, con reglas que determinan demasiado su resultado final. Hace falta entonces desarrollar métodos y procedimientos que enriquezcan su flexibilidad y sus posibilidades de programación[15].

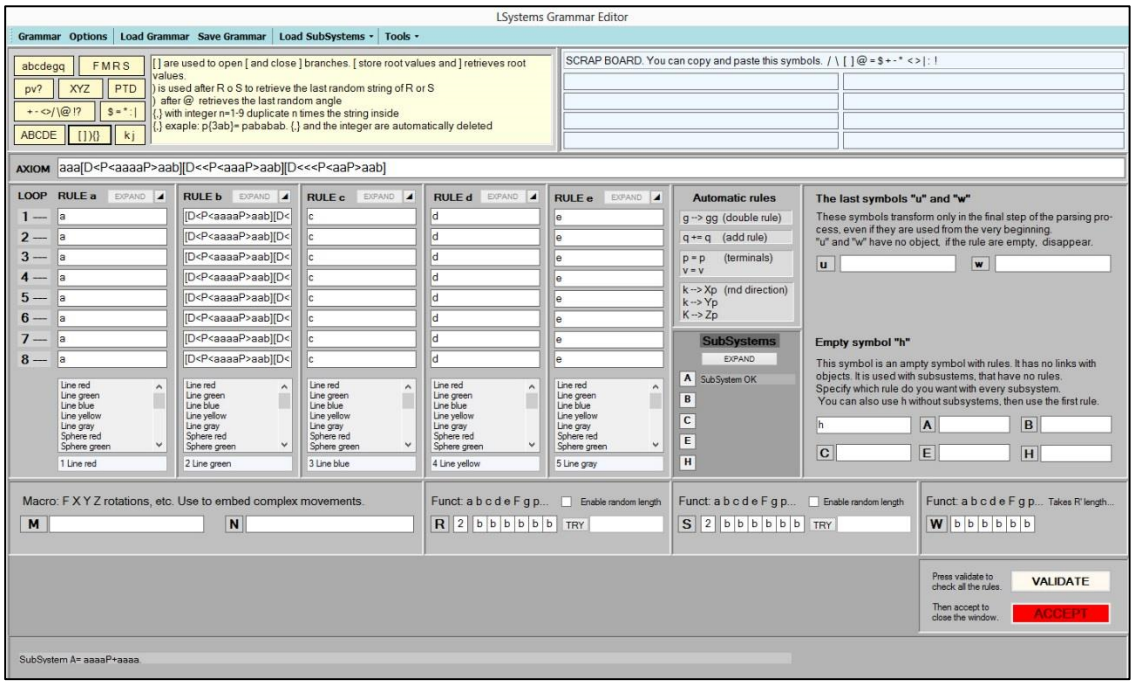


Fig. 6. Prueba de interfaz para un *software* de *shape grammars* del autor. En el diseño se ha buscado la completa visibilidad de los procesos, símbolos, parámetros y mecanismos del *software*, que pueden ser probados, revisados y documentados en tiempo real. Ninguna información queda oculta al usuario, que puede controlar los procesos al 100%, a pesar de su aparente dificultad. La transparencia del conocimiento permite una experiencia similar al dibujo manual, aunque no en sentido aptico, sino epistemológico.

Precisando el concepto de la virtualidad aumentada, la experimentación y el planteamiento de los problemas de diseño que se presentan por la relación con las estructuras naturales, lo precolombino y el dibujo a mano alzada (fig. 3,4) ha ayudado a poner a prueba la capacidad y flexibilidad de estos algoritmos. Además ha permitido implementar nuevas funciones que han acercado las gramáticas de la forma no solo a un

[15] También este trabajo de programación se ha desarrollado para el 1 Premio a la Investigación Interdisciplinaria del Consorcio de Universidades de Lima.

verdadero lenguaje de programación (fig. 6), sino a la creación de conocimiento como el que se genera en el momento mismo del dibujo a mano alzada, en ese proceso de observación simultánea del mundo observado y del mundo mental del creador (Pallasmaa, 2012, pág. 100).

Conclusiones

Proponemos acá ideas que valen como líneas de investigación artística con la tecnología, propuestas de trabajo. En primer lugar la relación naturaleza, arte y tecnología hoy cuenta con una base teórica más consistente que se alimenta de las ciencias de la complejidad, de los avances de las neurociencias, de la epistemología, del constructivismo o de la segunda cibernética.

Con respecto a la virtualidad aumentada, queremos subrayar otra vez, sobre todo para los educadores, que lo digital no debe invadir o sustituirse a la realidad, como en efecto sucede con la realidad aumentada o las redes sociales (Google Eyes; Facebook, etc.), sino que es lo virtual que se debe enriquecer con la realidad. Por esto es importante la inclusión del arte, de la mano, del cuerpo y de lo háptico. El modo para hacer esto es considerar lo virtual como la etapa de un proceso real, como su complemento o su síntesis. Así pues, la complejidad emergente de lo digital está solo en el proceso, no en el resultado: la complejidad la genera el usuario cuando puede producir con creatividad y libertad a través de un sistema informático cuyo patrimonio epistemológico es abierto y trasparente. Por último, desde el punto de vista de la estética, se trata de un arte que obviamente no se entiende en el sentido romántico y moderno (ni posmoderno), sino de una nueva manera que está todavía en elaboración. Es frente al solipsismo de las interfaces, a la justicia epistémica, a la idea de una nueva estética y a la identidad cultural dada por la tradición, que el arte digital debe decidir si permanecer en el *kitsch* o recuperar su importancia concreta.

Bibliografía

ADDISON, J. (1991). *Los placeres de la imaginación y otros ensayos de The Spectator*. Madrid: Visor.

CARR, N. (2008). «Is Google Making Us Stupid? What the Internet is doing to our brains». En *The Atlantic*, Julio/Agosto 2008. [en línea] <http://www.theatlantic.com/magazine/toc/2008/07/> [Fecha de consulta: 20/8/2015].

CROUSSE, V. (2011). *Reencontrado la espacialidad en el arte público del Perú*. Tesis doctoral en Espacio público y regeneración urbana: arte, teoría y conservación del patrimonio. Barcelona: Universidad de Barceloan, Facultad de Bellas Artes. [en línea]. <http://www.tdx.cat/handle/10803/1551> [Fecha de consulta: 20/8/2015].

DELANDA, M. (2009). «Emergencia, causalidad y realismo». En: «Arte, cultura y ciencias de la complejidad» [en línea]. *Artnodes*. N.º 9. UOC. [Fecha de consulta: 20/8/2015].

DELEUZE, G. y F. GUATTARI. (2004). *Mil mesetas*. Valencia: Pre-textos.

DE PRADA, M. (2009). *Arte y naturaleza. El sentido de la irregularidad en el arte y en la arquitectura*. Buenos Aires: Nobuko.

DERRIDA, J. (2001). *La verdad en pintura*. Buenos Aires: Paidós.

FISHWICK, P. (2006). *Aesthetic computing*. [en línea]. <http://www.cise.ufl.edu/~fishwick/aescomputing/> [Fecha de consulta: 20/8/2015].

FLUSSER, V. (2000). *Towards a philosophy of photography*. Londres: Reaction Books.

HABERMAS, J. (1990). *Moral consciousness and communicative action*. Cambridge: the MIT
Press.

KANT, I. (2004). *Lo bello y lo sublime. Metafísica de las costumbres*. Buenos Aires: Libertador.

KURZ, R. (2003). «La ignorancia de la Sociedad del Conocimiento», en *antroposmoderno*. antroposmoderno.com/antro-articulo.php?id_articulo=247.

LEYGHTON, M. (2001): *A Generative Theory of Shape.* Berlín: Springer-Verlag.

MOROZOV, E. (2011). *El desengaño de internet, Los mitos de la libertad en la red*. Barcelona: Ed. Destino.

OPPENHEIMER, T. *The flickering mind.* Nueva York: The Random House Publishing Group, 2004.

PALLASMAA, J. (2012). *La mano que piensa. Sabiduría existencial y corporal en la arquitectura*. Barcelona: Ed. Gustavo Gili.

PAREYSON, L. (1998). *Estetica*. Milán: Bompiani.

SALINGAROS, N. A. (2014). «Charles Jencks y el Nuevo Paradigma en Arquitectura». En *Arquitectura y deconstrucción*. Madrid: Mairea Libros.

SILBER, J. (2009). *Architetture dell'assurdo. Come il «genio» ha tradito un'arte al servizio della comunitá.* Turín: Lindau.

SOMMERER, C. y L. MIGNONNEAU. (2009). «Life Spacies y Life Spacies II: modelar sistemas complejos para arte interactivo». En: «Arte, cultura y ciencias de la complejidad» [en línea]. Artnodes. N.º 9. UOC. [Fecha de consulta: 20/8/2015].

STINY, G. y J. GIPS. (1971). *Shape grammars and the generative specification of painting and sculpture*. [en línea] <http://www.shapegrammar.org/ifip/SGBestPapers72.pdf> [Fecha de consulta: 10/8/2015].

STUBRIN, L. (2013). «Arte y ciencia: convergencias en el marco de la teoría de la complejidad». En: Pau Alsina (coord.). **«**Historia(s) del arte de los medios**»** [en línea]. Artnodes. N.º 13, pág. 80-87. UOC [Fecha de consulta: 20/8/2015].