Informe Final Proyecto de Investigación

Integración del diseño computacional con la materialización arquitectónica para la fabricación digital de pieles arquitectónicas

VIE-799-16

Escuela de Arquitectura y Urbanismo
Escuela de Ingeniería en Computación

Profesores investigadores:

MSc. Arq. Esteban Castro Chacón (coordinador)
MSc. Ing. Mauricio Avilés Cisneros
MSc. Arq. Sebastián Orozco Muñoz

Estudiantes:

Alejandro Mora Tenorio (computación)
Diego Méndez Arce (arquitectura)

Proyecto ejecutado de enero 2017 a diciembre 2017

Entregado: junio 2018

Tabla de contenido

1. Código y título del proyecto

VIE-799-16

Integración del diseño computacional con la materialización arquitectónica para la fabricación digital de pieles arquitectónicas

2. Autores y direcciones

MSc. Arq. Esteban Castro Chacón – *Arquitectura - Investigador coordinador.* ecastro@tec.ac.cr

MSc. Ing. Mauricio Avilés Cisneros – *Ingeniería en computación - Investigador asociado.* maviles@tec.ac.cr

MSc. Arq. Sebastián Orozco Muñoz – *Arquitectura - Investigador asociado.* sorozco@tec.ac.cr

Alejandro Mora Tenorio – *Ingeniería en computación - Estudiante asistente* moraalejandro94@gmail.com

Diego Méndez Arce – *Arquitectura - Estudiante con tesis vinculada.* diego@darkstudiocr.com

Julio Núñez – *Arquitectura - Estudiante asistente, 6 meses en el proyecto* julionunezcorrales@icloud.com

3. Resumen

El problema planteado por este proyecto es ¿Cómo aprovechar el manejo de la información de los sistemas informáticos para ser aplicado en el proceso de la toma de decisiones de diseño arquitectónico de pieles adaptadas a las condiciones tecnológicas y ambientales locales? El objetivo principal es el generar un caso práctico de aproximación proyectual que incorpore el diseño asistido por algoritmos al proceso de diseño arquitectónico, de forma que se logre una integración entre las oportunidades generadas por los avances informáticos en el manejo de datos y la toma de decisiones propias de la actividad creativa del diseño.

Para lograr esta integración, se exploraron diferentes técnicas de simulación y programación que permiten el diseño programado de sistemas de cerramiento donde la toma de decisiones con respecto a forma y configuración fueron dados por el programa de simulación generado como apoyo al arquitecto en busca de soluciones informadas y lógicas en relación a las cualidades morfológicas de un material flexible.

De este proceso se lograron generar propuestas de diseño que surgieron como resultado directo de las condiciones contextuales, específicamente como respuesta a la aceleración del viento y de la elección de los materiales a utilizar, desde una optimización para la fabricación digital y como resultado de un proceso programado mediante algoritmos.

El proyecto se concluye que con las herramientas de fabricación digital disponibles y el conocimiento del cómo aplicar la programación orientada a objetos, es viable una exploración del diseño computacional al incluir las restricciones propias del material, con una finalidad clara.

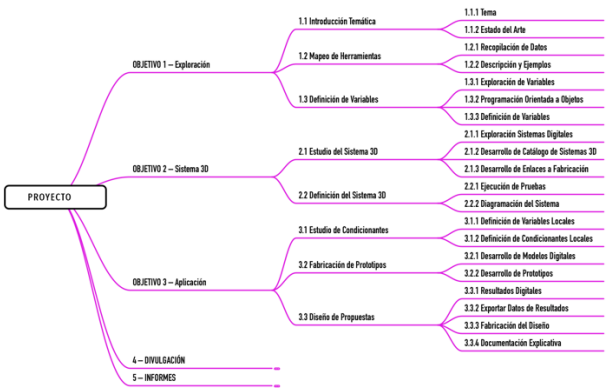

Imagen 1 Diagrama de objetivos y actividades del proyecto de investigación. Fuente: Autores.

4. Palabras clave

Fabricación digital, optimización geométrica, diseño asistido por algoritmos, computación material, pieles arquitectónicas, simulación digital

5. Introducción

En el ámbito de investigación de este proyecto confluyen campos que van desde las matemáticas, ciencias computacionales, biología, filosofía, ingeniería y arquitectura. En relación con los dominios inexplorados de esta convergencia y su potencial para la arquitectura, nace una disciplina emergente denominada ***Computational Design***, que significa literalmente Diseño Computacional. Actualmente, no existe una traducción consensuada del concepto al español, por esta razón y para efectos de este proyecto, se le llamará Diseño Computacional a la disciplina que emerge de la confluencia de los campos anteriormente mencionados, cuya particularidad está en digitalizar problemas de diseño y construcción del objeto arquitectónico. Esta disciplina aborda partes del proceso proyectual desde una perspectiva computacional donde la computación básicamente se encarga de procesar información e interacciones entre los elementos de un sistema específico para generar resultados con mayores niveles de complejidad.

> *"El procesamiento de información e interacciones entre los elementos que constituyen un ambiente específico... En una mayor relevancia con el diseño y el diseñador, la aplicación más general de la computación está en la generación de resultados producto del procesamiento de propiedades externas e internas. La computación provee una base para negociar e influenciar la inter relación de estos grupos de datos de información, con la capacidad de generar orden, formas y estructuras complejas.*" (Menges y Ahlquist, 2011, p. 13).

El abordaje al problema de diseño desde la perspectiva del Diseño Computacional habilita el empleo de comportamientos complejos en lugar de simplemente modelar una forma o figura compleja. La respuesta formal de un objeto arquitectónico bajo este paradigma no es resultado de una creación 'manual' del objeto 3D en un paquete de software (i.e. ArchiCAD, Revit, etc.), la representación del objeto 3D ya sea en una maqueta de cartón o en la computadora no está definida por una secuencia de dibujos y procedimientos de modelado como se hace convencionalmente, más bien, es un resultado directo de estructuras de datos procesadas por algoritmos y funciones de optimización y búsqueda, donde el diseñador es un disparador, controlador y filtro durante diferentes momentos del proceso, mas no el ejecutor absoluto de la creación del objeto arquitectónico.

El Diseño Computacional también tiene una fuerte influencia de la biología, toma conceptos evolutivos para incorporar nuevos paradigmas en el proceso de diseño arquitectónico. En estos casos se suele utilizar el término Diseño Generativo o Evolutivo.

Debe señalarse que este proyecto es interdisciplinario y se sitúa donde convergen principalmente la Arquitectura y la Ingeniería en Computación, además de otras disciplinas. Este campo es una nueva disciplina de origen reciente en la arquitectura y la computación a nivel mundial. Es un tema de vanguardia en un grupo selecto de universidades y pocas entidades privadas, que gozan de un gran prestigio académico y profesional a nivel mundial, dentro de las cuales se destacan: Massachusetts Institute of Technology, Stuttgart Computational Design Institute, Carnagie Mellon School of Architecture, Institute for Advanced Architecture of Catalonia, Architectural Asociation School of Architecture - London, Harvard University Graduate School of Design y Bartlett School of Architecture, de Londres. De ahí el hecho de que el tema central de la investigación no ha sido documentado de manera extensiva por sí mismo. En relación con la escasez bibliográfica Menges & Ahlquist mencionan:

> *"Encontramos, sin embargo, que ningún libro cubría este espectro de temas en este contexto específico. Esto se hizo incluso más evidente cuando nos encontramos en la situación*

afortunada de adquirir y ver cientos de títulos relacionados para la biblioteca de nuestro recién fundado Instituto para el Diseño Computacional en la Universidad de Stuttgart." (2011, p. 8-9)

Las investigaciones en el campo del Diseño Computacional tienen en su mayoría un carácter exploratorio debido a que el tema es emergente y a expensas de la innovación tecnológica, cuyo principal desarrollo es relativamente reciente. El origen del Diseño Computacional se dio en los 60's con la aparición del primer CAD original, SKETCHPAD. A pesar de esto, el progreso más importante se ha dado en los últimos años, tiempo durante el cual se han consolidado algunas líneas de investigación claras que implican una fuerte relación entre el proceso de diseño con la instrumentalización del comportamiento complejo de los materiales y la aparición de nuevos paradigmas arquitectónicos. A continuación, se presenta un breve resumen de los antecedentes y el estado actual de las investigaciones en este campo.

Los orígenes más primitivos e incipientes del Diseño Computacional pueden ser rastreados a un trabajo en específico: SKETCHPAD, considerado uno de los programas de cómputo más influyentes jamás escritos (Tedeschi, 2016). Desarrollado por Ivan Sutherland en el MIT en 1963. Las características principales fueron las siguientes: una pantalla de gráficos en tiempo real interactiva, un dispositivo de entrada interactivo (un lápiz digital) y un modelo geométrico subyacente. Las geometrías podían ser dibujadas como instancias paramétricas representando diferentes aspectos de la forma, el espacio y su estructura. A partir de esto, se derivó GRASP, un sistema generativo que utilizaba procedimientos aleatorios para la creación de formas y restricciones basado en la interrelación especifica de la estructura, la exposición solar y la organización programática. LOKAT también aportó un componente evaluativo del desempeño de acuerdo con reglas de asociación y proximidad de los componentes del programa.

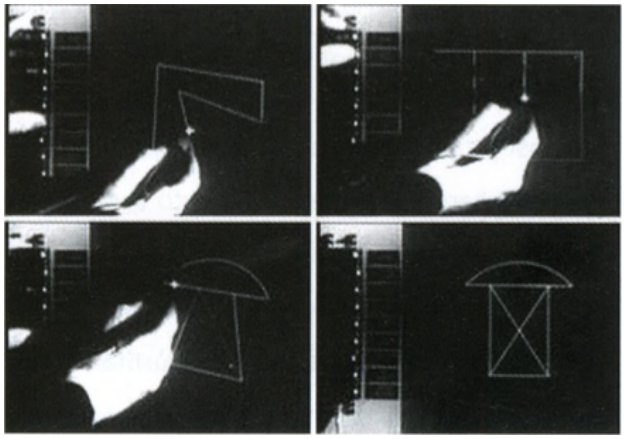

Imagen 2 Ivan Sutherland on MIT Lincoln Lab's TX-2 computer (1963). The Scketchpad interface.
Fuente: http://aasarchitecture.com/2014/07/research-pavilion-2013-14-icd-itke.html/research-pavilion-2013-14-by-icd-itke-15

Los primeros experimentos en Diseño Computacional revolucionaron la industria de la arquitectura, la construcción y la ingeniería en computación, SKETCHPAD, GRASP y LOKAT son el punto de partida del Diseño Computacional, las tecnologías CAD y BIM. Las raíces del abordaje computacional se pueden ver en estos primeros experimentos.

La red de contribuciones directas e indirectas a la investigación en el Diseño Computacional se extiende a diferentes ramas de las ciencias, pero existen varios autores e investigadores ya consolidados en este campo que trabajan para las instituciones que han hecho un aporte significativo en los últimos años.

El Instituto de Diseño Computacional de Stuttgart es una institución que ejemplifica muy bien la línea de investigación generalizada que sigue este grupo de investigadores e instituciones. Su trabajo académico y profesional se ha concentrado en el desarrollo de procesos de diseño integral en la intersección entre diseño computacional morfogenético, ingeniería biomimética y fabricación asistida por computadora, sus investigaciones posibilitan un desempeño del entorno construido altamente articulado.

Las investigaciones más importantes y contundentes son las que van dirigidas en generar un nuevo tipo de vínculo entre procesos alternativos de diseño generativo, simulación y optimización, con procesos de manufactura de alta precisión, que involucran en

muchos casos robótica y manufactura asistida por computadora, o ***Computer Aided Manufacturing*** (CAM), que en muchos casos presentan resultados de desempeño sostenible mejorado. Varias investigaciones con aplicaciones muy específicas en la optimización estructural son ejemplo de lo anterior, consisten en la optimización estructural empleando biónica y nuevas configuraciones geométrico/matemáticas como punto de partida y procesos que simulan comportamientos evolutivos presentes en la biología para optimizar el diseño las estructuras con superficies curvas complejas.

Uno de los comunes denominadores en este tipo de investigaciones es que intentan instrumentalizar comportamientos materiales complejos generando así vínculos estrechos con las tecnologías de manufactura computarizada.

Un ejemplo de las nuevas herramientas digitales que se han desarrollado producto de estas investigaciones es la Tesis Doctoral de Miloš Dimčić (2011) en El Stuttgart Computational Design Institute cuyo Fundador y Decano actual es Achim Menges. En la investigación, el diseño y el análisis estático son combinados para llevar a cabo la optimización de mallas envolventes generadas sobre una superficie de forma libre. Se desarrolla un plug-in para Rhinoceros 3D (software basado en la representación geométrica de NURBS), que utiliza algoritmos genéticos como un método de optimización que implementa llamadas iterativas automáticas a Oasys GSA (software de análisis estático comercial FEEM) para generar una malla envolvente estáticamente óptima. Para hacer esto posible, en la investigación se desarrollaron algunos tipos nuevos de generación automática de mallas. Diagramas Voronoi fueron utilizados de manera conjunta con el método de Fuerza-Densidad adaptado para desarrollar un nuevo tipo de malla estructural que llamamos Voronax.

La otra línea de investigación importante es la de base teórica, donde se hacen análisis profundos del progreso del Diseño Computacional y otras donde se plantean nuevos paradigmas en la arquitectura, sus métodos y procesos de materialización producto del progreso antes mencionado.

Luciana Parisi (2013) ofrece una inquisición filosófica en el estado del algoritmo en el diseño y la interacción arquitectónica. Su tesis plantea que la computación algorítmica no es simplemente un método matemático abstracto, sino que más bien constituye en su derecho propio un modo de pensamiento, en que su operación se extiende a formas de abstracción que están más allá del control y la cognición human directa. Esto incluye modos de infinidad, contingencias e indeterminación, así como cantidades incalculables que subyacen en el proceso iterativo del procesamiento algorítmico.

Imagen 3. Pabellón del ICD en colaboración con ITKE, desarrollado en el año 2014.
Fuente: http://aasarchitecture.com/2014/07/research-pavilion-2013-14-icd-itke.html/research-pavilion-2013-14-by-icd-itke-15

Dentro de este campo existe un nicho con potencial de investigación. La instrumentalización de comportamientos materiales complejos tiene como limitante en primera instancia el material. Las instituciones que actualmente llevan a cabo investigación en este campo cuentan con presupuestos robustos y tecnologías de fabricación robotizada avanzadas en comparación con lo que se dispone en nuestro país. Respecto al material, la investigación presente se diferencia de las

investigaciones actuales en el estado del arte porque incorpora dos grupos de grupos de variables principales que contextualizan el proceso algorítmico. El primer grupo de variables está asociado a la disponibilidad de la tecnología material y constructiva local y el segundo a las condiciones ambientales locales.

5.1. El problema

El problema consiste en ¿Cómo establecer un sistema material que pueda ser programable por medio de herramientas de diseño computacional? El problema es, en sí mismo, un vacío entre la arquitectura y la computación, que se manifiesta como una brecha entre la simulación computacional y los procesos creativos a nivel cognitivo dentro del diseño arquitectónico. En palabras más simples, las computadoras no están siendo usadas para pensar mejor o al menos para pensar de formas diferentes, sino para procesar con mayor rapidez de lo que hacemos los humanos.

Las computadoras, el software y el Internet, alteraron el mundo en el que vivimos, y la arquitectura no fue una excepción. A pesar de que la industria ha sacado gran provecho de la computarización en muchos aspectos, existe mucho por investigar. Los procesos de diseño no han cambiado, la computación no ha alterado su esencia. El problema está en que actualmente la computación funge como una prótesis virtual de nuestras manos, similar a las herramientas de dibujo convencionales, con algunas extensiones mejoradas, y no parecen haber expandido nuestras capacidades cognitivas y procesos de diseño de forma sustancial, en algunos casos ha sucedido lo contrario y la herramienta, más bien, ha prejuiciado y asfixiado procesos creativos, generando que en la práctica contemporánea, la computación del diseño sirve simplemente al propósito de extender o acelerar procesos de diseño ya bien consolidados." (Menges y Ahlquist, 2011, p. 8).

En general la industria de la arquitectura y la construcción no ha aprovechado el potencial que subyace en la computación, el uso más generalizado de las computadoras por el contrario se ha limitado a manipular estructuras de datos que ya fueron conceptualizadas a priori. La computación como una verdadera herramienta de diseño capaz de simular procesos creativos es ausente en términos generales.

> *"Los arquitectos no han sacado provecho del poder computacional de las computadoras, el modo predominante de utilizar computadoras actualmente es uno en el que entidades y procesos previamente conceptualizados en la mente del diseñador son introducidos, manipulados y almacenados en el sistema de la computadora. En contraste, la computación como una herramienta de diseño es empleada de forma muy limitada."* (Terzidis, 2006, p. xi).

A pesar de que existe una vasta diversidad de herramientas en el mercado tipo CAD (Computer Aided Design), estas han sido concebidas para un uso específico que no satisface las necesidades de diseño del usuario. Incluso, cuando el diseñador demuestra experiencia especializada en el manejo de uno o varios de los paquetes CAD disponibles en la industria de la arquitectura y disciplinas afines, encuentra serias limitantes en el uso del software como herramienta de diseño, en cambio, el software modificado por el diseñador a través del scripting provee un rango de posibilidades para la especulación creativa que simplemente no es posible usando el software original tal como fue intencionado. (Burry, 2011, p. 9)

Respecto a estas limitantes e interrogantes antes mencionadas, esta investigación profundizará en la instrumentalización material para la generación de pieles arquitectónicas a través de procesos de diseño computacional, donde la participación de la computación como herramienta de diseño toma mayor relevancia.

5.2. Justificación

Este proyecto surge de la necesidad de expandir el potencial de mejora en los procesos exploratorios de diseño arquitectónico y abre la posibilidad de visualizar nuevos paradigmas de diseño previamente ocultos, abriendo así el aspecto innovador y la posibilidad de posicionar a la institución a la vanguardia de la investigación en el campo de tecnologías emergentes en la arquitectura a nivel mundial.

Esta línea de investigación ha sido tradicionalmente abordada en países con un desarrollo tecnológico de punta y abiertos a la especulación sobre el futuro al aplicar dichas tecnologías. Es así que la posibilidad de explorar este nuevo paradigma de diseño y generación de la forma desde contextos de ***low tech*** y desde latitudes tropicales, representa una oportunidad para introducir un nuevo enfoque que puede aportar al desarrollo del campo específico.

Este proyecto enriquece el carácter creativo del diseño arquitectónico y expande las posibilidades arquitecturales del Diseño Computacional. Incrementa de forma sustancial la capacidad exploratoria de posibles soluciones geométricas en el diseño de las envolventes arquitectónicas contextualizadas a las necesidades de contextos tropicales como el de Costa Rica.

Dentro del ámbito del Diseño Computacional, la propuesta de investigación tiene novedad y un nivel inventivo sustancial, en el estado de la cuestión esta investigación no se encuentra anticipada, debido a su particular orientación específica, producto de los dos grupos de variables locales mencionadas en los párrafos anteriores, esta investigación da un aporte significativo, lo que implica un avance tecnológico considerable. El carácter vanguardista de la investigación posiciona la investigación y al TEC en un alto nivel en lo que respecta a las tecnologías emergentes en la arquitectura y la computación.

6. Marco teórico

¿Qué es Diseño Computacional?

Entender el Diseño Computacional es mucho más fácil si se visualiza como un proceso a través de una simple comparación con su referente más cercano (CAD). Si los procesos de Diseño Asistido por Computadora (CAD) comienzan con lo especifico y terminan con el objeto, los procesos computacionales comienzan aún más antes con propiedades elementales y reglas generativas que finalizan con la información que deriva la forma como un sistema dinámico (Menges & Ahlquist, 2011, p.13).

Para hablar de Diseño Computacional conviene aclarar la definición de Computación. Computación se confunde comúnmente con computarización. La Computación es el procedimiento de cálculo, i.e. determinar algo por métodos matemáticos y lógicos, mientras que la computarización es el acto de ingresar, procesar y guardar información en una computadora.

En esencia la computarización involucra la digitalización de entidades o procesos que son preconcebidos, predeterminados y bien definidos como la automatización, mecanización, digitalización y conversión. Mientras que la computación pretende emular o extender el intelecto humano y "En sus múltiples implicaciones, involucra resolución de problemas, estructuras mentales, simulación, cognición e inteligencia a base de reglas, para mencionar algunas." (Terzidis, 2006, p. xi). Debido a esta naturaleza la computación explora lo indeterminado, lo vago, lo confuso, y comúnmente procesos indefinidos, involucra racionalización, razonamiento, lógica, algoritmos, deducción, extrapolación, exploración y estimación.

6.1. La Forma Algorítmica

Un algoritmo, en su definición más general es una secuencia de pasos finitos, para efectos del proceso computacional es empleado para atender problemas en un número finito de pasos, Involucra deducción, inducción, abstracción, generalización y estructura lógica. Es la extracción sistemática de principios lógicos y el desarrollo de un plan de solución genérica. "Las estrategias algorítmicas utilizan la búsqueda de patrones repetitivos, principios universales, módulos intercambiables y enlaces inductivos. El poder intelectual de un algoritmo descansa en su habilidad para inferir conocimiento nuevo y extender ciertos límites del intelecto humano." (Menges & Ahlquist, 2011, p. 94).

La forma resultante (modelo geométrico) de un proceso algorítmico es el resultado de un proceso mucho más complejo y no convencional debido a que es el resultado de la interacción computacional entre reglas internas y presiones externas (Morfogenética), que por sí mismas originan en otras formas adyacentes (ecología). "Las reglas internas, pre concretas comprenden, en su actividad una forma embebida, lo que es hoy en día claramente entendido y descrito por el término algoritmo." (Stanford, 2008, p. 147)

6.2. Las pieles en arquitectura

Convencionalmente la envolvente de un objeto arquitectónico se entiende como la separación física entre un ambiente acondicionado y uno no acondicionado. La envolvente del edificio se compone de todos los elementos de la capa externa que sirven para mantener un ambiente interno seco, con calefacción o enfriamiento y que facilitan la climatización de dicho espacio. En general, la gran cantidad de funciones que cumple una envolvente se pueden clasificar en tres grandes categorías: de soporte, para resistir y transferir cargas estructurales y dinámicas; de control, para el flujo de materia y energía de cualquier tipo; y de acabado, para lograr un efecto estético deseado. A pesar de no ser la única, esta definición genérica es ampliamente aceptada en el gremio de la arquitectura a nivel mundial. Este perfil aborda la generación computacional de la piel arquitectónica en función de dos grupos de variables

primarias: la tecnología constructiva local y las condiciones ambientales locales.

La investigación en pieles arquitectónicas genera un interés en el ámbito de la arquitectura, la construcción y la ingeniería, debido a la importancia que generan las superficies de borde de los edificios en la climatización y control de los espacios internos, traduciéndose en una necesidad de control energético. Este campo está liderado por el evento anual llamado "Advanced Building Skins Conference" o Conferencia sobre las Superficies Avanzadas para Edificaciones. El objetivo de esta conferencia es el de contribuir a un enfoque planificado, multidisciplinario e integrado por arquitectos, ingenieros, científicos, fabricantes y la industria de la construcción para reducir el consumo de energía de las edificaciones. La conferencia internacional anual atrae 600 participantes de más de cincuenta países, y se realiza en sedes itinerantes en las ciudades con mayor innovación en el campo,[1] donde para el año 2018 se tiene programada la edición número 13 a realizarse en Bern, Suiza.

6.3. El aspecto material

La arquitectura es una práctica material predominantemente basada en una aproximación al diseño caracterizada por anteponer la forma sobre su subsecuente materialización. Desde el renacimiento, la creciente división entre procesos de diseño y las formas de construcción ha llevado a un período de creciente dependencia en herramientas de representación que pretenden ser descripciones geométricas y explícitas al tiempo de ser instrucciones para el traslado del dibujo a la construcción. Esto ha hecho que los arquitectos, con pocas excepciones como Gaudí, Frei Otto, Heinz Isler y otros, inevitablemente acojan métodos de diseño que personifican la separación jerárquica entre definición de la forma y de su materialización.

Imagen 4. Antoni Gaudí. Modelo de estudio para el diseño de la Sagrada Familia.
Fuente: http://dataphys.org/list/gaudis-hanging-chain-models/

El diseño computacional por su parte permite que por medio de algoritmos y sistemas generativos se pueda manipular la información existente en un sistema para generar respuestas informadas en relación a los datos incorporados. El diseñador es por tanto un agente que coordina un sistema en busca de estrategias programables que sean capaces de desarrollar el diseño formal. Este método de abordaje al proyecto ha sido utilizado en primera instancia por artistas gráficos y diseñadores conceptuales.

Existen muchas variantes de procesos de diseño computacional, no todos toman en cuenta el material necesariamente. En este proceso en particular, el material es un punto de partida para la búsqueda de la forma. De hecho, uno de los referentes más importantes en el campo habla de que el material es capaz de "computar la forma" es decir el resultado formal, es producto de lo que el material es capaz de generar. Sería fácil pensar que el proceso de diseño convencional ya toma en cuenta esto, pero no lo es así. En el diseño computacional el material produce formas desde lo específico hacia lo general mientras que en el diseño convencional sucede lo contrario. La instrumentalización de las propiedades materiales y sus capacidades de desempeño son puntos de partida esenciales y constituyen un nuevo paradigma de diseño.

[1] https://abs.green/es/homs-es/

Consecuencia de esto es que existen investigaciones en diseño computacional que no integran la variable material dentro de su concepción procedimental, aunque también existen aquellas donde la generación digital de la forma es un fin en sí misma, incluso cuando el objetivo del diseño sea su materialización. Un ejemplo del primer caso es el trabajo desarrollado por William Latham, que mediante el uso de algoritmos genéticos podía "criar" modelos tridimensionales que evolucionaban en la pantalla, los cuales eran muy eficientes en la generación de diseños estéticamente novedosos al dejar que el programa tomase decisiones de diseño, aunque sin ninguna funcionalidad o resultado práctico.

En el caso del segundo tipo de investigación, resalta el trabajo realizado por Michael Hansmeyer, quien utiliza un proceso generativo abstracto con resultados muy complejos para después buscar los materiales y técnicas de fabricación que puedan materializar sus diseños, es decir, el proceso digital y el proceso de fabricación no están relacionados integralmente, ya que la generación digital, al ser abstracta y no gobernada por leyes físicas, permite cualquier resultado posible, mientras que los procesos de fabricación están restringidos por los materiales que se utilizan y las cualidades propias de las máquinas.

Imagen 5 Parte de una columna de Michael Hansmeyer en proceso de fabricación.
Fuente: https://www.espazium.ch/eine-archologische-ausgrabung

6.4. La fabricación en el proceso de diseño computacional

Por su parte, en el ejercicio y prácticas actuales de la arquitectura, las herramientas digitales son todavía utilizadas para crear esquemas de diseño a través de un rango de criterios que dejan de lado las capacidades morfológicas y de desempeño inherentes de los sistemas materiales utilizados. Los procesos de materialización, construcción y producción son considerados sólo después de que la forma se ha elaborado, llevando a procesos de ingeniería "top-down" que usualmente se yuxtaponen con lógicas inapropiadas de elaboración.

La fabricación digital es uno de los recursos de avance tecnológico que posibilita el desarrollo del diseño computacional. La complejidad formal y de componentes variados producto del manejo de datos en los procesos digitales, condiciona la factibilidad de ejecución según técnicas convencionales de manufactura. En otras palabras, no tiene sentido diseñar objetos de cientos o miles de partes distintas, cada una con un acople particular o diferente, para fabricarlo manualmente siguiendo planos impresos. Las herramientas asociadas a la fabricación digital permiten brincarse la parte analógica del proceso para trasladar los datos del diseño directamente a la manipulación del material, según las restricciones propias de cada equipo especializado.

El proceso idealmente busca la automatización de todas las partes del proceso, sin embargo, aún en los centros de investigación más avanzados, el componente humano siempre resulta indispensable, ya sea para preparar el material, clasificarlo, llevarlo a su sitio, embalarlo, y finalmente, ensamblarlo. Pero aún con estas limitaciones, la digitalización del proceso permite facilitar estas tareas mediante la visualización de los datos y organización de la información, ya que la posibilidad de etiquetar y nominar cada pieza única del diseño, se facilita al tener los datos embebidos en el sistema.

7. Metodología

Este proyecto se caracteriza por estar identificado como de investigación aplicada, con un importante componente práctico y experimental que en sus diferentes etapas se debía poner a prueba, y que, según la experiencia de los investigadores, constantemente se debían tomar decisiones sobre el rumbo a seguir. Esto se consolidó mediante reuniones semanales en equipo para el trabajo conjunto mediante la dinámica de taller. Estos talleres se dividían por un lado en sesiones de generación y evaluación de ideas, y por el otro, de ejecución y elaboración de prototipos físicos y digitales.

La estructura de ese proceso investigativo estuvo ordenada mediante tres objetivos específicos, relativamente secuenciales cronológicamente, y que requirieron actividades y metodologías específicas para cada uno de ellos.

El Objetivo 1 consistió en identificar la factibilidad de uso de diversas técnicas de programación y simulación vinculadas al software Processing, para aplicar en la materialización arquitectónica. Para ello se planteó una primera actividad de mapeo y uso de las herramientas de programación digital aplicables al diseño tridimensional, que consistió en dos fases principales;

1) Una introducción conceptual que amplió el estado del arte con el fin de lograr un mejor entendimiento de todos los integrantes de la investigación. Esto incluyó una revisión bibliográfica más amplia a la planteada en el perfil del proyecto, y una búsqueda de proyectos similares realizados en diferentes partes del mundo.

2) La identificación de las herramientas principales, para con ello analizar ejemplos de uso en proyectos similares y un primer uso de cada una de ellas. El total de herramientas analizadas fue de 13, las cuales se detallan en el capítulo de resultados. Adicionalmente, se contó con la retroalimentación de la Fundación Costa Rica para la Innovación en relación a las experiencias realizadas por ellos en torno a la fabricación digital, por medio de una visita a sus instalaciones en la sede de San Pedro. De esta experiencia se obtienen referencias del tipo de materiales usados y detalles sobre las prácticas fundamentales en la realización de prototipos mediante fabricación digital. Del ICAT, Virginia Tech, por su parte, conoció el avance generado, pero sin mayor aporte al proyecto. Como parte de esta actividad, se incluyó la visita de campo al Centro de Innovación INTEL, ubicado en Belén, para un acercamiento a las tecnologías de fabricación digital e inteligencia artificial impulsadas por la compañía.

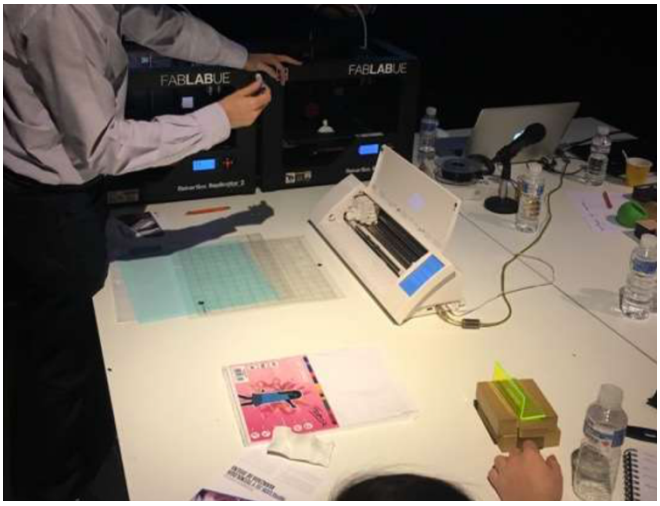

Imagen 6. Visita a las instalaciones la Fundación Costa Rica para la Innovación. Fuente: Autores.

Para este primer objetivo, también fue necesaria una segunda actividad que consistió en la definición de variables a programar. Para lograr determinar estas variables, se analizaron las restricciones más importantes en relación al contexto local, considerando así parámetros locales de clima, tecnología y materiales. Estos parámetros debían a su vez, valorarse según las herramientas técnicas de simulación y programación estudiadas en el objetivo anterior, para tener la seguridad de que las restricciones encontradas serían factibles de digitalizar mediante la exploración algorítmica.

Imagen 7. Visita de campo a las instalaciones del Centro de Innovación INTEL. Fuente: Autores.

El objetivo 2 consistió en proponer un sistema material que pueda ser programable por medio de las herramientas del diseño computacional. Para ello se realizaron dos tereas principales:

1) Estudio de sistemas tridimensionales a utilizar en relación a un comportamiento material. En este caso, producto de las actividades anteriores, se deciden valorar dos comportamientos materiales que configuran dos sistemas tridimensionales diferentes: El '*tensegrity*' y la flexión de barras. Esto se realizó para determinar cuál de los dos era más factible de programar y cuál generaba mayores posibilidades de diseño.
   Paralelamente, se realizó una exploración por medio de modelos físicos para probar las posibilidades de uso y manejo de los diferentes materiales planteados en el objetivo uno, pero en relación con los sistemas tridimensionales mencionados anteriormente. Esto permitiría seleccionar el material y el comportamiento material a programar de manera racional a raíz de lo experimentado.

2) Definición del sistema material y su programación. Una vez valorados los resultados de la tarea anterior, se procedió a seleccionar el material a trabajar para extraer los datos relevantes de su comportamiento, a través de mediciones estadísticas y pruebas físicas, documentadas y medidas. Esta selección se realizó a partir de los datos obtenidos y según sus cualidades estructurales y de adaptabilidad al uso en pieles arquitectónicas. Esta parte del trabajo se realizó directamente por medio de la tesis de grado del estudiante Diego Méndez.

   Posteriormente, se debía realizar la identificación de los parámetros a incluir o excluir en el sistema de modelado, incluyendo los contextuales, ambientales y tecnológicos, para determinar un único uso que se le asignaría a la envolvente.

Finalmente, el objetivo 3 consistió en la aplicación de diseño de envolvente arquitectónica aplicado a un contexto local específico. Para esto se utilizó lo analizado en el estudio de condicionantes locales para la definición de variables, determinándose una función específica para la piel arquitectónica. Esta utilidad sería evaluada según simulaciones virtuales especializadas, efectuadas sobre las opciones surgidas durante el proceso.

La configuración geométrica que resultase con mejor desempeño, con una mejor respuesta a los esfuerzos estructurales de la piel, pasaría a la fase de prototipado, la cual debía incluir la mayor cantidad de recursos de fabricación digital posibles, entre los que destacan el corte láser y la impresión 3D. Para ello se propuso analizar y describir los aciertos encontrados en el proceso de investigación.

Finalmente, la programación algorítmica generada, permitiría el diseño variado de propuestas que incluyeran las variables antes descritas, que, al incluir las bibliotecas de programación estudiadas, permitiría la interacción en tiempo real del sistema diseñado.

8. Resultados

Este capítulo se estructura según la secuencia de actividades que se plantearon por objetivo al inicio de la investigación, los cuales sirvieron de base para la toma de decisiones que se debían realizar durante el proyecto y se detallaron en la metodología. Estas actividades se resumen en:

- Estudio de las técnicas de programación.
- Valoración de variables contextuales: clima, tecnología, materiales.
- Valoración de las técnicas de simulación en relación al desempeño material.
- Sistemas digitales a utilizar según el comportamiento elegido.
- Sistemas tridimensionales para prototipar.
- Extracción de datos del material y definición del sistema.
- Ejecución de pruebas.
- Fabricación Digital.
- Fabricación del prototipo.

8.1. Estudio de las técnicas de programación

a. PROCESSING

Imagen 8 Logo de Processing, versión 3.0.
Fuente: https://processing.org/

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Fue iniciado por Ben Fry y Casey Reas a partir de reflexiones en el Aesthetics and Computation Group del MIT Media Lab dirigido por John Maeda.

Processing es utilizado por artistas y diseñadores como una herramienta alternativa al software propietario. Puede ser utilizado tanto para aplicaciones locales, así como aplicaciones para la web (Applets). Se distribuye bajo la licencia GNU GPL. Existen varias razones por las que esta aplicación ha sido utilizada en buena medida por artistas y diseñadores para generar productos visuales a partir de código computacional, pero entre ellas destacan el hecho de que es un buen punto de partida para los no expertos en programación, así como la versatilidad del mismo en generar resultados visuales y de simulación, ya que el programa se basa en animaciones visuales como resultado de técnicas de programación y algoritmos. (Shiffman, 2012).

Si bien es cierto, no es el único lenguaje o ambiente computacional que permite generar este tipo de resultados gráficos de simulación, (lo mismo se puede lograr con ActionScript, Java Script, Java, Cinder entre otros), Processing presenta un ambiente tipo “creative coding” que ha generado una comunidad activa y colaborativa a nivel mundial que lo utiliza de forma sistemática y práctica. Algunos de los trabajos realizados mediante Processing se pueden ver en la página “exhibition” del portal web de la herramienta: https://processing.org/exhibition/


```
import toxi.physics.*;
import toxi.physics.behaviors.*;
import toxi.geom.*;
import peasy.*;
import controlP5.*;

PeasyCam cam;

VerletPhysics physics;
ArrayList<Particle> particles;
Particle [][] mesh;
ArrayList<Spring> springs;
ArrayList<ConstrainedSpring> cSprings;
ControlP5 controller;
PMatrix3D currCameraMatrix;
PGraphics3D g3;

boolean applyPhysics = false;
float ParticleWeight = 0.1;
float gravity = 0.5;
float frictionCoefficient = -0.001;
float staticFrictionCoefficient = 0.0172;


float MeshParticleWeight = 0.05;


int camx = 20;
int camy = 30;
int camz = 30;


void setup() {
  controller = new ControlP5(this);
  g3 = (PGraphics3D)g;

  size(2000,1400,OPENGL);
  smooth();
  frameRate(60);


  physics=new VerletPhysics();
  physics.addBehavior(new GravityBehavior(new Vec3D(0, gravity, 0)));

  cam = new PeasyCam(this, 250);
  scale(10, 10, 10);

  particles = new ArrayList();
  springs = new ArrayList();
  cSprings = new ArrayList();
  cam.lookAt(camx,camy,camz);
  cam.setMinimumDistance(1);

  controller.addSlider("Gravity",0.001,-0.001,0.005,0,0,1,1);
  controller.setAutoDraw(false);

  physics.setWorldBounds(new AABB(new Vec3D(0,0,0),new Vec3D(30,9.3,20)));


  draw();

  //*****************************************************************PRUEBA DE TENSEGRITY*****************************************
  Particle p4 = new Particle(5, 0, 0, MeshParticleWeight);
  Particle p2 = new Particle(15, 9, 0, MeshParticleWeight);
  Particle p3 = new Particle(15, -10, 0, MeshParticleWeight);
  //p2.lock();
  p3.lock();
  particles.add(p4);
  particles.add(p2);
  particles.add(p3);
  for (Particle k : particles){
      physics.addParticle(k);
```

Imagen 9 Visualización de un código en la interface usual de Processing. Fuente: Autores.

b. CONTROL P5

ControP5 es una biblioteca o '***plug-in***' para el ambiente de programación Processing, escrita por Andreas Achlegel. Su última actualización es del año 2015.

Por medio de controlP5, se pueden incluir interfaces gráficas de usuario, o como se conoce por sus siglas en inglés, GUI, en los programas desarrollados en Processing. Esto significa la posibilidad de incluir controles interactivos en el sketch de Processing para modificar en tiempo real los resultados de la simulación. Esto bajo el formato de botones, 'sliders' campos de texto, interruptores entre otros.

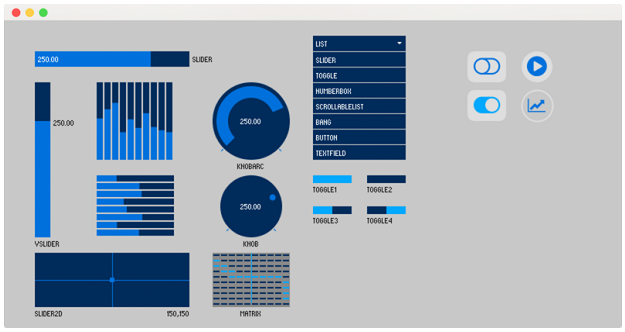

Imagen 10. Ejemplos de controles de interface.
Fuente: http://www.sojamo.de/libraries/controlP5/#features

c. CULEBRA

Imagen 11. Logo de culebra

Culebra es una biblioteca para la visualización de agentes autónomos en el lenguaje Processing. Incluye abstracciones para la creación de los agentes con comportamientos predefinidos y configurables. La biblioteca puede utilizarse de tres maneras diferentes: utilizando los objetos y controlador definidos en la biblioteca, creando objetos propios que implementen los comportamientos de la biblioteca, o creando objetos derivados de los objetos de la biblioteca.

La biblioteca provee diferentes implementaciones de agentes que tienen dos capacidades generales: la de moverse y la de perseguir otros objetos.

Los comportamientos definen la forma en que se desplazan los agentes en el espacio, los que implementa la biblioteca son: manada, deambular, ruido Perlin, rastreo, creador de hijos, arrastre sobre una malla y fuerzas (atracción, repulsión y otras).

d. HEMESH

Imagen 12. Logo HMESH.
Fuente: https://github.com/wblut/HE_Mesh

Hemesh es una biblioteca para Processing creada por Frederik Vanhoutte (Bélgica), de acceso gratuito y abierto mediante la licencia de "***Public Domain***[2]", que permite la generación de mallas tridimensionales, o como es descrito en la página oficial, mediante esta biblioteca de Java, se pueden crear y manipular mallas poligonales en Processing.

Toda la documentación se encuentra en GitHub en su versión 6.1 para Processing 3, actualizada en el 2017. De su utilización, práctica y ejemplos vistos, se desprende que es una herramienta apropiada para la manipulación de sólidos en configuraciones complejas, lo cual la hace una opción viable a utilizar si la investigación se desarrollase en esa línea.

e. IGEO

Esta es una biblioteca libre y de código abierto para el modelado 3D. Está orientada al diseño en arquitectura, diseño de productos, entre otros. Incluye abstracciones para el manejo de vectores, ***B-splines*** racionales no uniformes, panelización, manejo de multiagentes y simulación de físicas. Además, cuenta con la posibilidad de exportar los modelos en formato Rhinoceros 3dm.

El autor es el diseñador Satoru Sugihara del Instituto de Arquitectura de California del Sur y fundador y director de la firma ATLV.

Imagen 13. Pantallazo de la página web de la biblioteca iGeo, a junio 2017: http://igeo.jp/p/

La biblioteca está escrita en el lenguaje de programación Java y puede ser utilizada desde el ambiente de programación Processing en su versión 2.

Lo que se busca con la utilización de la biblioteca es la generación de estructuras tridimensionales mediante procesos algorítmicos. En una exploración inicial se observa que puede ser utilizada con diferentes propósitos como los enumerados a continuación.

a) Matemática de vectores: se cuenta con abstracciones básicas para el manejo de vectores tridimensionales y operaciones sobre los mismos como suma, resta, multiplicación, división, inversión, distancia, producto punto, producto cruz, reflexión, bisección.
b) *B-splines* racionales no uniformes: es uno de los modelos matemáticos utilizado para describir geometrías en espacios y es de los más utilizados en software de tipo CAD. Las geometrías se describen por medio de arreglos

[2] http://creativecommons.org/publicdomain/zero/1.0/

de coordenadas tridimensionales los cuales definen puntos de control para generar superficies continuas.

c) Panelización: permite la utilización de puntos discretos extraídos de una figura o superficie continua y utilizarlos para la creación de nuevas estructuras rectas como líneas o paneles. Esto es relevante ya que los materiales que se puedan utilizar en construcción usualmente no pueden seguir la continuidad curva de un modelo y es necesario adaptar el mismo para reflejar las posibilidades del material.
d) Agentes autónomos: estos son entidades que deciden independientemente cómo interactuar con su entorno sin seguir un plan global. Esta interacción incluye generalmente algún tipo de movimiento y la creación de nuevas relaciones entre los elementos del espacio. La biblioteca cuenta con abstracciones para modelar este tipo de interacciones, entre ellas, partículas y agentes. Estas características son básicas y necesarias para la creación de comportamientos autónomos, aunque no son suficientes para definirlos en su totalidad.
e) Simulación de físicas: se incluyen abstracciones para la simulación de las leyes newtonianas, permitiendo el manejo transparente de conceptos como posición, velocidad, aceleración, fuerza. Se incluyen abstracciones como líneas de tensión, resortes, redes tensoras, gravedad.

Documentación

La biblioteca cuenta con documentación en línea bastante extensa. Por cada una de las clases se cuenta con una descripción de atributos y métodos. Para la mayoría de los métodos se cuenta con descripciones de sus valores de entrada y salidas. Está disponible en línea en el sitio del proyecto.

La documentación fue generada automáticamente a partir de la documentación interna del código, por lo que en ocasiones el nivel de detalle de la misma no es el deseado. El sitio del proyecto cuenta con una serie de tutoriales que ilustran la utilización de la biblioteca, pero aparentemente fueron creados con una versión anterior de la misma, por lo que mucho de lo que se construye manualmente en los ejemplos ya cuenta con abstracciones programadas que pueden ser utilizadas para evitar hacerlas de nuevo. Sin embargo, esto no se encuentra documentado apropiadamente y es posible que al explorar la documentación se encuentre con que existen abstracciones que solucionan directamente algún problema que se esté solucionando.

Acceso a ejemplos y código fuente

Se ha encontrado que el acceso a ejemplos en línea de la utilización de la biblioteca es limitado. El principal punto donde se encuentran ejemplos es la página oficial de la biblioteca, pero como se mencionó anteriormente, estos ejemplos no aprovechan al máximo las características disponibles. El sitio no cuenta con algún foro donde se pueda contactar a otros desarrolladores que utilicen la biblioteca. En búsquedas en la red no se encuentran ejemplos de utilización de la biblioteca diferentes a los ofrecidos por la página oficial.

Una ventaja es que la biblioteca es de código abierto, por lo que se encuentra disponible el código fuente completo en la plataforma GitHub. Esto ha sido de gran utilidad ya que, al encontrarse con limitaciones de documentación o ejemplos, el estudio directo del código fuente ayuda a entender y aprovechar las capacidades de la biblioteca.

f. PEASYCAM

PeasyCam es una biblioteca independiente de código abierto desarrollada por Jonathan Feinberg, originalmente para Processing 2 pero también compatible con la tercera versión. La biblioteca provee una herramienta de cámara la cual es controlada a través del mouse, en la cual se puede manejar la rotación y el acercamiento con los cuales se observa la simulación, así como fijar un punto determinado para la posición de la misma.

La instalación de esta biblioteca es muy sencilla, se instala de manera estándar en Processing 3, simplemente se hace ***click*** en Sketch, luego en ***import library*** y finalmente en ***add library*** se ingresa el nombre en el buscador y se procede a instalar.

La documentación[3] es bastante completa, en ella se incluyen los métodos disponibles en la biblioteca y los tipos de variables que reciben, para la mayoría de métodos no se incluye una descripción detallada de los mismos.

Acceso a ejemplos y código fuente:

Dado que la funcionalidad de esta biblioteca es bastante específica y de sencilla utilización, para los intereses del proyecto no es necesario tomar muchos ejemplos como referencia, sin embargo, dentro de los foros de Processing se pueden encontrar varios ejemplos de su utilización.

El código fuente de la biblioteca está disponible para ser descargado a través de github[4]. Sin embargo, el acceso al mismo no es de mayor relevancia dado que las necesidades que se tienen para la biblioteca son muy específicas.

Experiencia de utilización

La experiencia al utilizar esta biblioteca para la visualización de los ejemplos creados en Processing ha sido muy positiva, la misma es de muy sencilla utilización y no se ha encontrado ningún inconveniente hasta el momento.

g. PLETHORA

Plethora es una biblioteca de código abierto para el lenguaje Processing que sirve para la simulación de comportamientos de manejo o conducción. Se basa en el estudio de comportamientos en sistemas adaptables complejos realizado por Craig Reynolds. La biblioteca es parte de un extenso proyecto que busca mejorar la utilización de tecnologías de programación en el diseño algorítmico. Aunque se tuvo acceso a versiones anteriores de la biblioteca, en la actualidad el proyecto se encuentra suspendido y no es posible descargar la biblioteca directamente del sitio.

plethora-project

Imagen 14. Logo de Plethora.
Fuente: https://www.plethora-project.com

h. TOXICLIBS

ToxicLibs es una biblioteca independiente, de código abierto elaborada para tareas de diseño computacional, escrita en java para ser utilizada en el entorno de programación Processing (versión 3). La biblioteca ha sido hasta el momento desarrollada por Karsten Schmidt. La biblioteca puede ser utilizada para diferentes áreas de trabajo, como diseño generativo, animación, visualización de datos, arquitectura, fabricación digital, como medio educativo, entre otros.

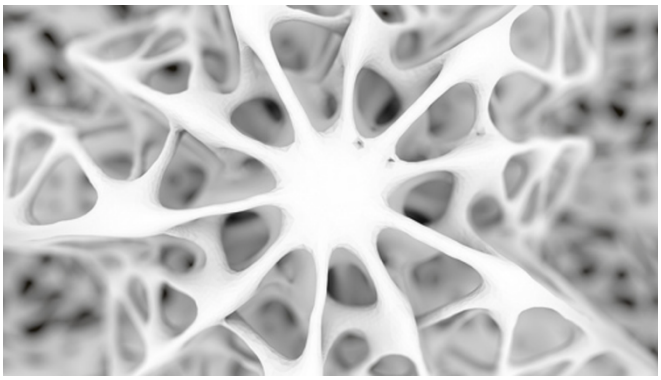

Imagen 15. Estructura tipo ósea generada con Toxiclibs.
Fuente: https://www.flickr.com/photos/toxi/5330031660/in/pool-toxiclibs/

El objetivo de la utilización de la biblioteca es la simulación y generación de estructuras tridimensionales mediante el uso de algoritmos, especialmente al implementar las siguientes técnicas:

- Manejo de vectores: Se pueden utilizar múltiples métodos para la construcción y manejo de vectores tridimensionales o en dos

[3] http://mrfeinberg.com/peasycam/reference/index.html

[4] http://github.com/jdf/peasycam/

dimensiones, para los mismos se pueden hacer sumas, calcular el ángulo entre dos vectores, calcular la distancia, calcular producto cruz y producto punto, obtener los componentes individuales del vector entre otros. Estos vectores son utilizados como parámetros de ubicación, de fuerza, entre otros para las clases de simulación de física.

- Simulación de comportamientos físicos: Se cuenta con abstracciones para simular diferentes comportamientos de física tanto en un ambiente de dos dimensiones como en uno tridimensional. Dentro de los comportamientos que se pueden simular está, la gravedad, fuerzas de atracción, partículas, resortes, entre otros.
- Agentes autónomos: Los agentes autónomos son aquellos que tienen un comportamiento que puede ser determinado por una serie de guías u objetivos los cuales son programados, pero no está determinado de manera directa o explicita por el programador, es decir actúan de manera autónoma. Para esto se puede utilizar los métodos de la biblioteca de atracción y repulsión bajo ciertos parámetros y sobre ciertos objetos, también se pueden utilizar algoritmos genéticos, para modelar un comportamiento o resultado deseado, a partir de posibilidades aleatorias que, dependiendo de un criterio de aptitud, se reproducen con ciertas modificaciones también aleatorias.

Documentación

La documentación[5] es bastante completa ya que se pueden encontrar todas las clases y métodos de las mismas, sin embargo, para la mayoría de los métodos no se da una explicación de su funcionamiento si no simplemente el tipo de parámetros que recibe, existen algunas excepciones donde se da una breve descripción del funcionamiento del método, pero ninguno entra en detalle.

Acceso a ejemplos y código fuente

Una fuente de ejemplos de la utilización de esta biblioteca, fueron los realizados por Daniel Shiffman, un programador y profesor en la escuela de arte de NYU (New York University), los ejemplos están en forma de video, por lo que son de gran utilidad para el entendimiento no solo de la biblioteca sino también de los conceptos y abstracciones que se están programando. En estos ejemplos se exploran las diferentes herramientas para la simulación de comportamientos físicos, así como de agentes autónomos, también se ejemplifica el uso de algoritmos genéticos. Adicionalmente para cada video se incluye el código escrito durante el ejemplo, lo cual es muy útil como referencia.

Otra fuente de ejemplos se puede encontrar en el sitio web de la biblioteca, consiste de una página con varios tutoriales de distintos temas. Los mismos pueden ser videos o enlaces hacia el foro de Processing donde se incluye una breve explicación y el código del tutorial. Sin embargo, la cantidad de tutoriales y la información contenida en los mismos es un poco limitada.

El código fuente de la biblioteca se encuentra disponible y es descargable desde ***Bitbucket***, esto presenta la ventaja de que se puede mantener los últimos cambios al día, también es posible obtener información más detallada de la que está disponible a través de la documentación de la biblioteca.

Conectividad con otras herramientas

Esta biblioteca tiene la posibilidad de exportar archivos en formato STL, el cual es compatible con Rhinoceros, esto para los objetivos del proyecto es de gran utilidad ya que permite una comunicación eficiente entre las dos herramientas. También existe la posibilidad de

[5] Se encuentra en el siguiente link: http://toxiclibs.org/docs/verletphysics/overview-summary.html

exportarse utilizando cualquier biblioteca para exportar datos geométricos de Java.

i. RHINOCEROS + GRASSHOPPER

Grasshopper es un 'Plug-in' para el programa de modelado tridimensional Rhinoceros desarrollado por David Rutten y Robert McNeel & Associates. Fue creado en el 2007 para Rhino 4.0 originalmente con el nombre de "*Explicit History*" y rebautizado en el 2008 como Grasshopper. En pocos años logró ganar vasta popularidad entre usuarios y desarrolladores, incluyendo estudiantes, académicos y profesionales, para ser hoy considerado como una de las herramientas de modelado tridimensional avanzado más populares.

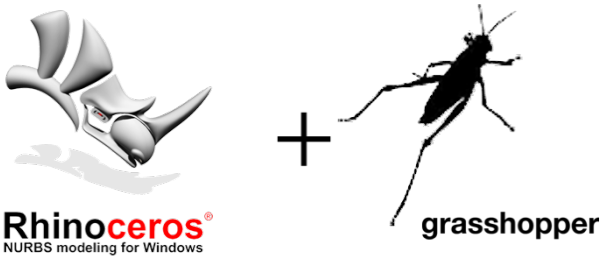

Imagen 16. Logos de las herramientas Rhino y Grasshopper. Fuente: http://www.grasshopper3d.com

Entre las ventajas que ofrece la herramienta para la exploración formal, se encuentran:

- Una comunidad de usuarios muy amplia y creciente, que generan una red para compartir trabajos y conocimientos, que posibilita la interacción para realizar preguntas y discutir problemas, esto en www.grasshopper3d.com
- Es una herramienta en constante actualización y mejoras, muchas de ellas a partir de la retroalimentación de los usuarios.
- Existe una amplia oferta de *plug-ins* desarrollados para Grasshopper de forma independiente, que permiten extender las capacidades del programa.
- Tiene el potencial de interactuar con otros programas, no solo en compatibilidad, sino que además mediante la interacción en tiempo real.
- Permite la interacción de datos entre el modelado y el hardware (Arduino, Kinect, etc.)

j. KANGAROO

Kangaroo es una biblioteca o *plug-in* para Grasshopper, que funciona como una plataforma de simulación de sistemas físicos basados en partículas y resortes. Fue desarrollado por Daniel Piker con el fin de generar un sistema de fácil uso enfocado al diseño.

Imagen 17 Logo Kangaroo

Kangaroo permite que los diseñadores interactúen con la forma a partir de simulaciones en tiempo real de sistemas de partículas y resortes. Existen dos formas de interacción:

- Directa: Incluye la manipulación de puntos de anclaje, fuerzas y propiedades elásticas.
- Paramétrica o Interacción asociativa: Los puntos de anclaje, fuerzas y propiedades elásticas pueden ser encadenadas paramétricamente con otras partes del modelo 3D.

Si bien es cierto, Kangaroo presenta menor dificultad para lograr resultados físicos producto de la interacción de fuerzas, de forma similar a como se hace en Processing, tiene la limitación de que para sistemas complejos resulta muy difícil de utilizar y de controlar.

k. DYNAMO

La tendencia de incorporación de herramientas de programación computacional dentro de las diversas ramas de diseño ha traído consigo la necesidad de una mayor accesibilidad de sus herramientas específicas, para un público de usuarios no capacitado en el desarrollo de código fuente ni lenguajes de programación tradicionales.

El paradigma de programación visual (PV) surge como una respuesta a esta necesidad, y busca como objetivo primordial brindar una interfaz gráfica de usuario, más intuitiva, y menos dependiente del dominio de la sintaxis de los diversos códigos y sus lenguajes. "***La programación visual permite a los seres humanos describir procesos usando ilustraciones, mientras que, por su parte los lenguajes típicos de programación fuerzan al programador a pensar como una computadora. Así se permite al programador describir los procesos en términos más humanamente lógicos***"[6].

A efectos de diseño arquitectónico, una de las plataformas de programación visual más recientes e interesantes es la denominada Dynamo, desarrollada inicialmente por el programador Ian Keough a partir del año 2012, la cual se basa fundamentalmente en el uso de diagramas de flujo, para la representación gráfica de la interacción entre objetos, nodos y comandos de lenguajes IronPython, y Helix3D mayoritariamente.

Junto con esto, una de las principales características de esta plataforma, radica en el involucramiento de la empresa Autodesk para su desarrollo a partir del año 2016, con lo cual se ha estrechado significativamente la interoperabilidad con herramientas BIM.

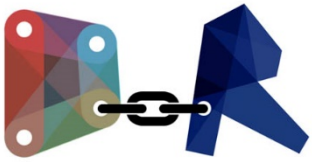

Imagen 18 Logo de DynamoBIM de Autodesk - Revit. Fuente: http://dynamobim.org

Hoy en día, Dynamo es la herramienta de PV oficial de esta empresa, lo que ha permitido a su vez, una integración directa con la plataforma BIM Autodesk Revit. Esta conexión representa una ventaja competitiva con respecto de otras herramientas similares.

Entre otras ventajas de esta herramienta específica se encuentran su naturaleza de código abierto, la cual permite ampliar su espectro de colaboración, difusión, desarrollo y adopción práctica. Finalmente cabe rescatar que se encuentran en desarrollo módulos específicos, orientados al paradigma del diseño computacional por medio de algoritmos generativos parametrizables, los cuales tienen una cercana relación con la temática y objetivos de la presente investigación.

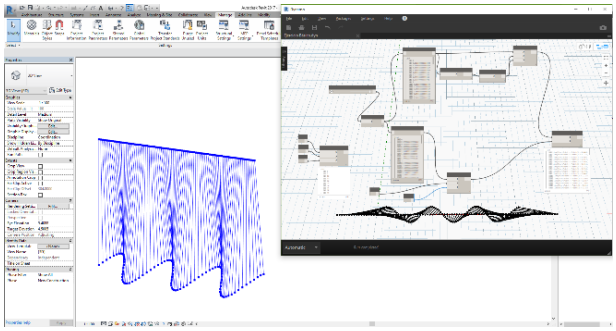

Imagen 19. Pruebas de parametrización de superficies en Dynamo. Fuente: Autores.

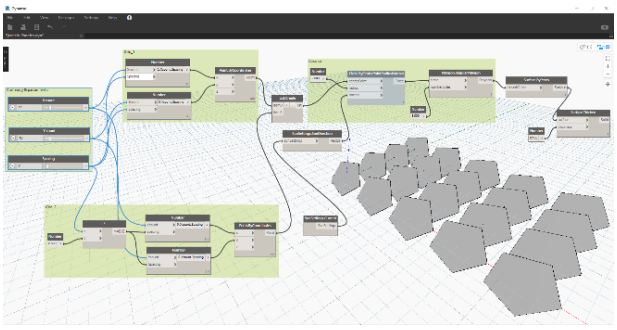

Imagen 20. Pruebas iniciales de panelización en Dynamo. Fuente: Autores.

I. LADYBUG

Una de las capacidades de ampliación funcional de la plataforma Dynamo, es la incorporación de "Paquetes" de código (script), dirigidos a la solución de problemas específicos, desde el comportamiento estructural de

[6] Revell Matthew, https://www.outsystems.com/blog/what-is-visual-programming.html, Agosto 2017.

los materiales, hasta el rendimiento geométrico-climático de un diseño. Como parte de esta línea, los programadores Mostapha Sadeghipour Roudsari y Chris Mackey, junto con un equipo de colaboradores especializados, iniciaron desde el año 2012, el desarrollo de una serie de paquetes de simulación bioclimática para diseño arquitectónico.

El principal y más desarrollado de estos paquetes se denomina Ladybug, y se define como "***una colección de aplicaciones para el diseño medioambiental y su enseñanza***"[7].

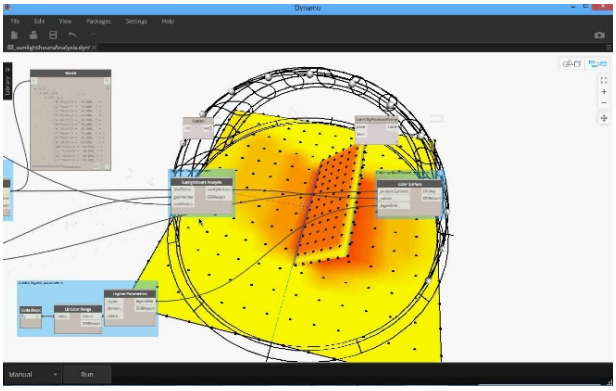

Imagen 21. Pruebas de simulación de radiación solar. Fuente:https://www.youtube.com/watch?v=6aD9U9Een3c

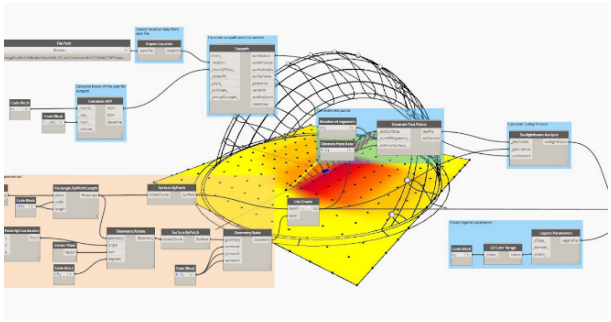

Imagen 22. Pruebas de simulación de radiación solar. Fuente:https://www.youtube.com/watch?v=6aD9U9Een3c

En suma, se trata de un entorno de simulación y análisis de factores de diseño asociados a la realidad climática de un emplazamiento específico, los cuales incluyen: Iluminación natural, radiación solar, confort higrotérmico, ventilación natural, gestión energética, micro-generación eléctrica, y otros afines.

Entre las principales ventajas de esta familia de aplicaciones se encuentran:

- Software validado, basado en motores de cálculo consolidados.
- Adquisición gratuita y naturaleza de código abierto.
- Basado en comunidad de colaboración.
- Integrado con múltiples interfaces de programación visual y modelado 3D.
- Flexible, basado en componentes modulares.
- Paramétrico y basado en simplificación iterativa.
- Visualmente descriptivo e interactivo.
- Multi-plataforma, escrito sobre lenguaje Python capaz de correr en cualquier sistema operativo.

m. HONEYBEE, BUTTERFLY Y DRAGONFLY.

El paquete Ladybug, cobija a toda la familia de aplicaciones para el análisis de diversas variables ambientales, sin embargo, es también el nombre de la aplicación específica para análisis y visualización de data climática. Junto con esta aplicación se encuentran otras tres, destinadas al análisis de las variables higrotérmica, solar, y eólica, estas son denominadas respectivamente: Honeybee, Dragonfly, y Butterfly**.**

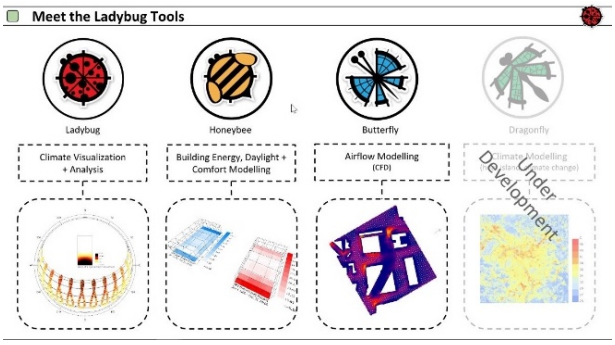

Imagen 23. Familia de herramientas del paquete Ladybug. Fuente: https://pdug.co/september-12-honeybee-ladybug/

Estas aplicaciones tienen la tarea de articular los modelos tridimensionales con los diversos motores de

[7] Roudsari M., Mackey C., https://www.ladybug.tools/about.html#team, Octubre 2017

cálculo validados científicamente, de entre los cuales se destacan: Radiance, Therm, OpenStudio, EnergyPlus y otros similares.

El siguiente gráfico describe la interrelación entre estas aplicaciones y motores de cálculo.

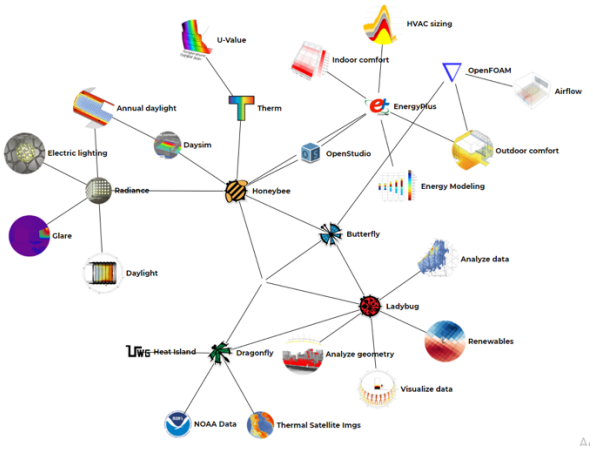

Imagen 24. interrelación entre aplicaciones y motores de cálculo vinculadas a Dynamo, incluyendo Radiance, Therm, OpenStudio y otras.
Fuente: https://www.ladybug.tools/

De esta forma, se manifiesta visualmente la enorme flexibilidad y rango de utilización del paquete ***Ladybug***, el cual se visualiza como el futuro entorno de simulación computacional para entornos de programación visual.

n. VVVV

VVVV es un ambiente de programación visual interactiva diseñado para prototipado y desarrollo ágil. Está específicamente diseñado para facilitar el manejo de medios con interfaces físicas, gráficos controlados por movimiento en tiempo real, interacción de usuarios simultáneos con gráficos, audio y video. El ambiente incluye un entorno de desarrollo, lenguaje de programación visual, biblioteca de funciones extensa y entorno de ejecución. Además del lenguaje de programación visual, permite la programación de lenguajes como hlsl y C#.

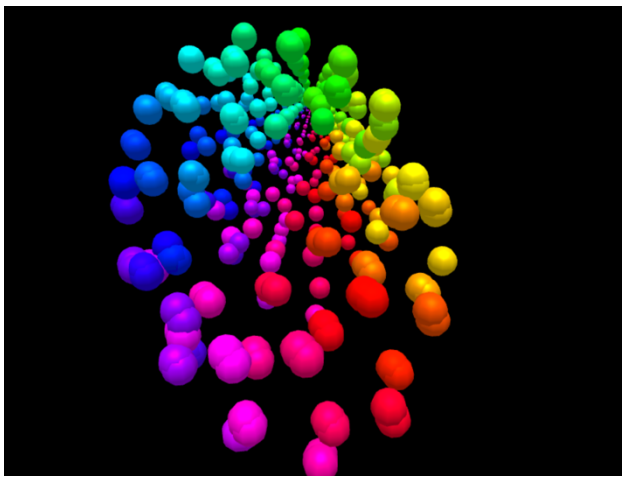

Imagen 25. Renderizado de esferas generadas por medio de vvvv durante la exploración de la herramienta. Fuente: Autores.

Algunas de las posibles aplicaciones y usos de vvvv son: animación 2D y 3D, manejo masivo de múltiples pantallas o proyectores, efectos visuales para imágenes y video, ***video mapping***, visualización de datos, computación física (Arduino, Kinect, Oculus Rift, etc.), manipulación de sonido y visión computarizada (detección de movimiento, reconocimiento facial, etc.).

A pesar de que VVVV es una herramienta muy versátil y amplia, dada su marcada orientación de carácter artístico se considera como no apropiada para el proyecto de investigación.

o. SIMSCALE

SimScale es un programa ingeniería asistida por computadora (CAE, por sus siglas en inglés) de computación en la nube que permite el Análisis Computacional de Dinámica de Fluidos, Análisis de Elementos Finitos y Simulaciones Térmicas. La plataforma al estar en la nube permite al usuario correr más simulaciones e iterar más posibilidades de diseño en comparación a un sistema local. En nuestro caso se utilizó para simular el desplazamiento de las barras de fibra de vidrio al ser dobladas. Debido a que el análisis requiere de datos precisos del material, se descartó debido a la poca disponibilidad de especificaciones técnicas del material.

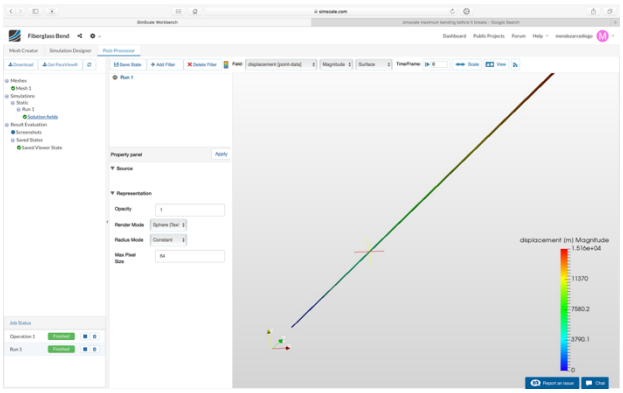

Imagen 26. Análisis de deformación de una barra, mediante el programa SimScale. Fuente: Autores.

8.2. Valoración de variables contextuales: clima, tecnología, materiales

Las investigaciones que están a la vanguardia en este campo cuentan con presupuestos elevados y una disponibilidad tecnológica de punta. Esta relación entre costo e innovación ha excluido a países en vías de desarrollo como el nuestro de la investigación en estos temas, donde las necesidades primarias siempre tienen prioridad sobre la búsqueda de la innovación. Sin embargo, el potencial del diseño computacional es tan amplio que vale la pena tratar de contextualizar la investigación para evidenciar la viabilidad del campo. Es por esto que la aproximación investigativa debe adaptarse a las condiciones locales y estar en relación con las necesidades contextuales, especialmente considerando que este proyecto es el primero que se realiza en Centroamérica y el Caribe y uno de los primeros en Latinoamérica.

Por esta razón es que la propuesta de investigación define tres restricciones fundamentales para la ejecución del proyecto:

- **Clima**: Responder a una necesidad climática en estrecha relación con la funcionabilidad de las pieles
- **Tecnología**: Utilizar la tecnología de vanguardia disponible en nuestro contexto, sin generar desperdicios o exceso en sobre costos
- **Materiales**: Utilizar materiales de bajo impacto y costo que sean apropiados para resolver la necesidad climática y para la utilización de las tecnologías disponibles.

Como se indicó en el capítulo anterior, las pieles pueden cumplir tres funciones básicas elementales, las cuales son:

- **Soporte**: para resistir y transferir cargas estructurales y dinámicas.
- **Control**: para el flujo de materia y energía de cualquier tipo.
- **Acabado**: para lograr un efecto estético deseado.

La función de soporte para esta investigación queda relegada a la capacidad auto-soportante de la piel misma, es decir, no cumplirá ninguna función estructural mayor más allá de soportarse a sí misma.

La función estética o de acabado, se considera una cualidad resultante no buscada en esta investigación, mientras que la capacidad de control es la que mayor reto representa, ya que son múltiples las posibilidades que se generan al respecto. Desde el punto de vista local, sin embargo, se identifican tres agentes de control que una piel en el trópico debe considerar: El agua, la luz y el viento.

Cada uno de estos agentes sujetos de control, tiene un comportamiento definido y específico que son en sí mismos amplios y complejos, además de apropiados a considerar en cualquier diseño de envolventes. Por esa razón fue fundamental seleccionar sólo uno de ellos para que, a partir de ahí, se tomaran las decisiones siguientes en relación a materiales y tecnologías a utilizar.

Si bien es cierto se valoró la posibilidad de generar pieles para la generación de sombra, finalmente se decidió por trabajar la variable de brisa.

Pieles para lograr sombra

Como parte del proceso de estudio de sistemas tridimensionales, las pruebas a escala 1:1 se llevaron a cabo mediante un proceso de diseño parametrizado. Diferentes segmentos circulares paralelos fueron unidos a través de trayectorias curvas pobladas a su vez por un número variable de puntos de anclaje. De estos puntos se trazaron diferentes tipos y ensayos de trayectorias, para definir las membranas de protección solar que buscaba proveer el ejercicio.

Imagen 27. Ejemplo de una necesidad típica de sombra en nuestros contextos, no solventada eficientemente por una piel auto-portante en San Pedro. Fuente: Autores.

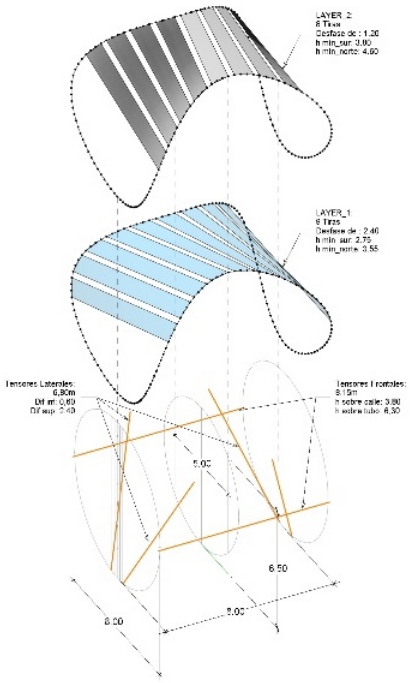

Imagen 28. Propuesta para la generación de una piel para proteger del sol para un evento exterior. Fuente: Autores.

Durante este proceso iterativo de análisis de opciones de diseño, el principal criterio de selección o descarte estuvo constituido por los resultados de rendimiento obtenidos en la plataforma de simulación computacional.

Se analizaron tres variables específicas para comparar posteriormente sus resultados: Radiación solar incidente, sombreamiento efectivo, e incidencia de viento sobre el sistema. De ellos, se desprendió la necesidad de fragmentar la membrana de protección, para disminuir la presión que se acumula sobre ellas y el efecto de vela (empuje lateral) a la que se somete el sistema.

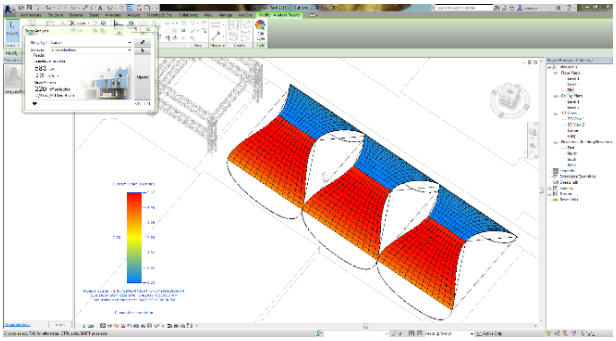

Imagen 29. Simulación de radiación solar incidente sobre membranas. Fuente: Autores.

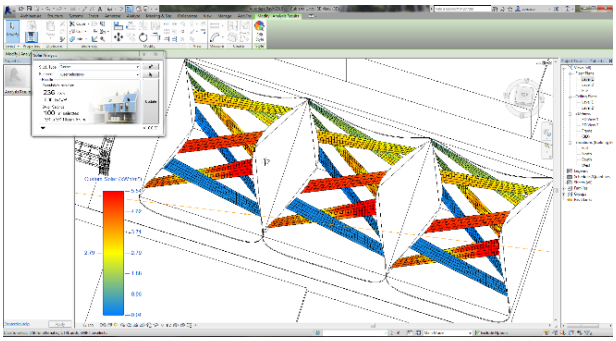

Imagen 30. Simulación de radiación solar incidente sobre membranas. Fuente: Autores.

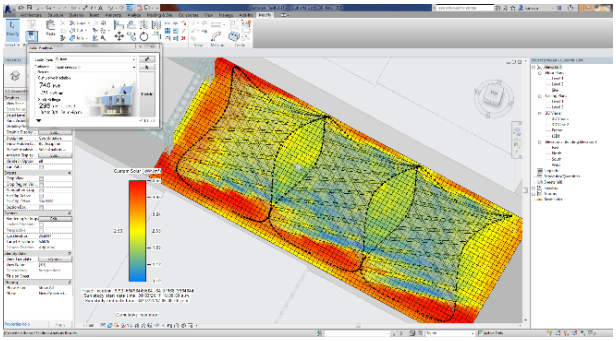

Imagen 31. Simulación de sombreamiento efectivo sobre área habitable. Fuente: Autores.

A pesar del esfuerzo por anticipar este factor de comportamiento estructural, el resultado final a escala natural, demostró una alta variabilidad entre las condiciones simuladas y las observadas en la realidad, por lo que uno de los principales hallazgos del ensayo es la necesidad de mayor rigurosidad en esta fase del proceso.

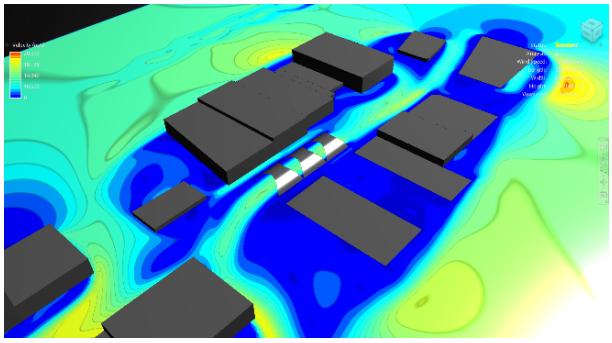

Imagen 32. Simulación de incidencia eólica sobre área habitable. Fuente: Autores.

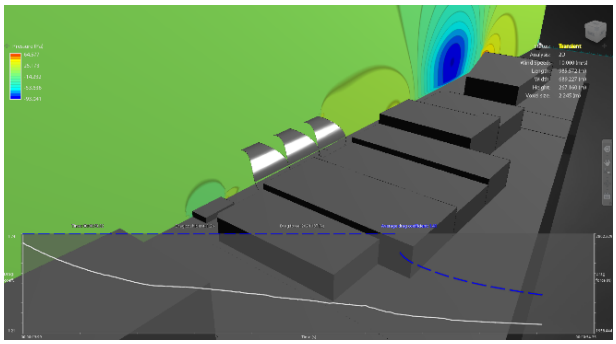

Imagen 33. Simulación de incidencia eólica y curvas de arrastre sobre área habitable. Fuente: Autores.

Otra de las conclusiones metodológicas sobre este tipo de procesos tiene que ver con la necesidad de visualización simultánea de resultados referentes a dos o más variables, es decir, la representación visual múltiples parámetros sobre el modelo de trabajo, en lugar de cada uno por aparte. Así, por ejemplo, sería necesaria la capacidad de apreciar la radiación solar incidente junto con la iluminación efectiva, o bien la aceleración del viento junto con la presión ejercida sobre las superficies circundantes.

Pieles para mejorar la ventilación

Como objetivo de rendimiento del sistema a diseñar, el equipo de trabajo se propuso lograr un conjunto de elementos con la capacidad geométrica de acelerar un flujo de aire incidente.

Este efecto de aceleración de un flujo de aire por angostamiento, se conoce como fenómeno Venturi, el cual es a su vez una interpretación del principio de

Bernoulli, desarrollado para el estudio de mecánica de fluidos.

En términos técnicos, comportamiento se describe como un aumento en la velocidad, y disminución en la presión que experimenta un fluido en movimiento dentro de un conducto cerrado, al ser canalizado a través de una sección menor[8].

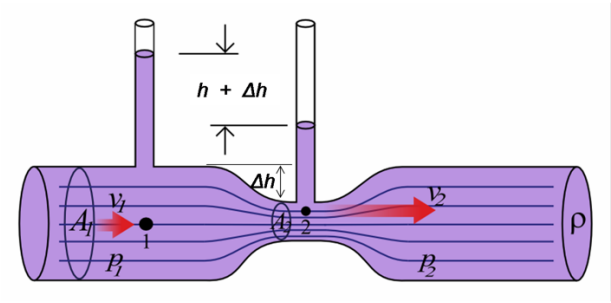

Imagen 34. Diagrama del efecto Venturi.
Fuente: http://slideplayer.com/slide/4180053/

La aplicación de este principio dentro de un proceso de diseño arquitectónico es de especial importancia durante la propuesta de envolventes de un espacio habitable.

Como abordaje general, un sistema de redireccionamiento y aceleración de viento se basa en la ruptura y fragmentación de una superficie originalmente plana, para la creación de aperturas y canalizaciones que logren angostar y acelerar los flujos de aire incidente.

Imagen 35. Prototipo de ventana optimizada para el aprovechamiento del efecto Venturi. Fuente: Orozco, Gamboa, Cuevas. 2016

Idealmente, este tipo de envolvente, o dispositivos deben de colocarse sobre las caras de presión positiva (a barlovento) de un edificio, y a la mayor altura posible, para tratar de captar el máximo caudal de ventilación.

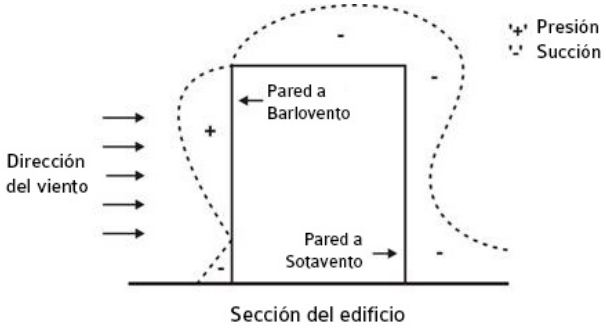

Imagen 36. Visualización de la diferencia de presión sobre las caras de un edificio.
Fuente: http://gbtech.emsd.gov.hk/english/utilize/natural.html

A manera de recomendación final, diversas investigaciones consideran más favorable el re direccionamiento angular del viento incidente, en lugar de una trayectoria oblicua. En general, una trayectoria diagonal dentro de un rango entre 30-60 grados con respecto de la dirección del plano de la envolvente es

[8] M. A. Gálvez Huerta; et alt. (2013). *Instalaciones y Servicios Técnicos*. Madrid: Sección de Instalaciones de Edificios. Escuela Técnica Superior de Arquitectura, U.P.M.

ideal, para una correcta distribución de las corrientes de ventilación en el espacio interior.

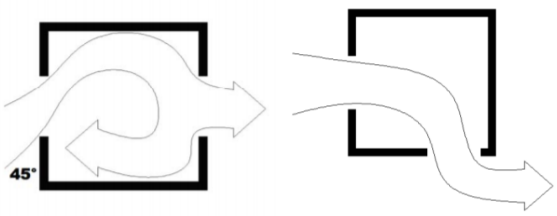

Imagen 37. Diferencia de distribución entre alineamientos oblicuo y diagonal en el diseño de aperturas arquitectónicas.
Fuente: http://vilssa.com/ventilacion-natural-ventilacion-cruzada

Consideraciones en relación a los materiales

El material debe tener disponibilidad. Debe ser un recurso de fácil obtención en el mercado local, en la medida de lo posible reutilizable, de bajo impacto ambiental y o reciclable y no puede tener un costo elevado en relación con los costos de los materiales convencionales disponibles a nivel local. Así que el material debe tener compatibilidad con las necesidades típicas de las inversiones que se realizan en el componente de superficies de pieles. Debe poder ser manipulado con facilidad por operarios no especializados y compatible con las dimensiones de los espacios de trabajo disponibles. Además, un aspecto fundamental es que la tecnología para su manipulación y fabricación debe ser compatible ("low-tech") para mantener esta viabilidad contextual.

A continuación, estos son los criterios de selección del material para su exploración y valoración:

- Disponibilidad
- Impacto ambiental
- Precio
- Compatibilidad
- Escala
- Tecnología requerida

El sistema material parte de una idealización relativamente simple, una barra flexible que puede formar un arco o un círculo. Es indispensable que el material seleccionado sea capaz de computar un arco o un círculo de forma natural bajo los esfuerzos generados por el sistema.

Como consecuencia se le practicó al material pruebas de doblado similares de manera empírica con el objetivo de explorar sus capacidades para reproducir un arco o un círculo.

Uno de los materiales más interesantes que se probó fue un tipo de bejuco conocido como putarrá, que crece en la zona de Los Santos y el Empalme, y se utiliza para fabricar los canastos del café. El tallo de la planta presenta comportamientos complejos que dependen del tiempo de secado, cuando está muy verde vuelve fácilmente a su posición original, pero si ya se ha secado lo suficiente aumenta resistencia a una deformación elástica. El tallo es de diámetro variable entre los nodos y los nodos a su vez son de diámetro variable, debido a que no es un cilindro perfecto cada sección de tallo y de nodo tiene un diámetro máximo y mínimo. Además, la forma del material no es perfectamente lineal, es sinuoso a lo largo de su eje.

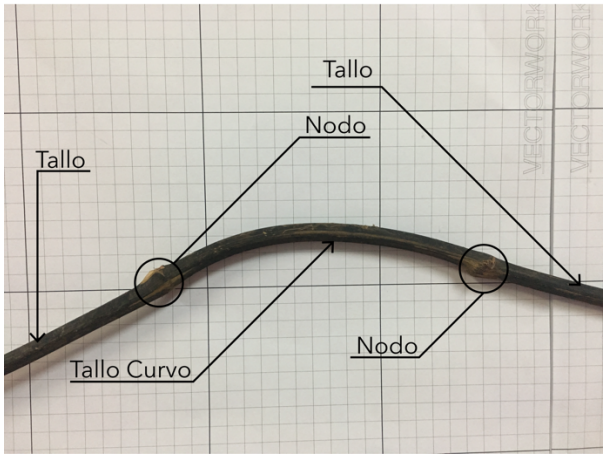

Imagen 38. Heterogeneidad del Putarrá en sus nodos, tallos y curvatura en los tallos. Fuente: Autores. Fuente: Autores

A pesar de lo anterior el material fue descartado debido al tiempo y los recursos que se requieren para trabajar con este tipo de materiales tan heterogéneos. Todo esto presentó dificultades para extraer datos en un tiempo razonable. Ejemplo de lo anterior es la siguiente

tabla en la que se muestran la gran cantidad de mediciones que requieren para hacer un muestreo de seis ejemplares. La predicción del radio mínimo de curvatura antes de que falle por estrés es sumamente variable debido a que su heterogeneidad produce esfuerzos locales en el material entre los nodos debido a las diferencias de dimensiones.

Tabla 1 Mediciones de diámetros máximos y mínimos para nodos y tallos de Putarrá.

		Tallos								Nodos							
Ejemplar	Diám. promedio	Mín./ Máx.	D1 (mm)	D2 (mm)	D3 (mm)	D4 (mm)	D5 (mm)	D6 (mm)	D7 (mm)	Nodos	Promedio de Nodos	Mín./Máx. Nodos	N1	N2	N3	N4	N5	N6
PU-001	9,10	8,44	7,93	8,32	8,82	8,68				3	13,82	11,76	11,20	11,96	12,13			
		9,77	9,30	9,69	10,20	9,88						15,87	15,28	16,08	16,25			
PU-002	9,91	9,26	9,10	8,90	8,75	9,47	9,46	9,85		6	15,64	13,73	12,50	12,65	12,69	13,24	13,60	17,67
		10,56	10,42	10,25	10,42	10,59	10,85	10,85				17,56	16,51	17,19	17,12	17,75	18,41	18,35
PU-003	8,27	7,61	7,15	7,39	7,55	7,85	7,73	8,00		6	13,56	11,25	10,57	11,49	11,03	10,74	11,28	12,36
		8,93	8,99	8,78	8,89	8,59	8,86	9,49				15,87	14,79	16,25	17,38	15,45	15,55	15,78
PU-004	9,05	8,42	7,93	8,60	8,71	8,44				3	13,48	11,97	11,67	12,21	12,03			
		9,68	9,55	9,95	9,65	9,56						14,98	15,14	15,39	14,41			
PU-005	10,28	9,26	8,93	9,37	9,64	9,39	9,20	9,03		5	16,03	13,23	12,95	13,16	13,50	13,31	13,25	
		11,30	10,52	11,32	11,48	11,60	11,40	11,45				18,82	18,09	18,45	19,35	18,78	19,44	
PU-006	9,98	9,26	8,79	9,80	9,28	9,22	9,04	9,45		6	15,56	13,13	12,59	13,21	14,05	13,16	12,07	13,67
		10,69	10,86	10,78	10,88	10,71	10,44	10,49				17,99	19,18	18,31	18,41	18,25	17,11	16,70
PU-007	10,41	9,38	9,24	9,25	9,72	9,55	9,46	9,20	9,24	7	16,91	14,30	12,49	15,08	16,15	13,88	13,90	14,83
		11,43	11,22	11,34	11,63	11,54	11,46	11,59	11,25			19,52	19,47	21,18	17,79	19,47	19,67	19,83

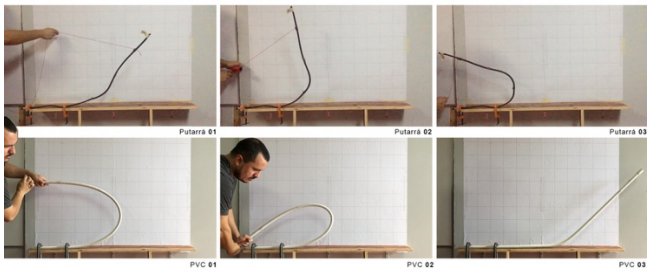

Imagen 39. Pruebas realizadas a la caña de putarrá (arriba) y PVC de ½" Ø (abajo). La caña de putarrá por ejemplo no computaba eficientemente un radio uniforme, mientras que la secuencia del PVC muestra que, a pesar de poder computar radios muy pequeños, el material llega a deformarse y a perder su tensión al ser llevado a esos niveles de estrés. Fuente: Autores

Imagen 40. Los nodos son zonas donde han crecido más ramas que fueron cortadas durante la recolección del material, este proceso puede producir fallas en el material desde antes de utilizarlo. Fuente: Autores

Cada material probado expresa mediante su comportamiento la posibilidad de uso y de escala. En algunos casos el material resultaba ser poco resistente para ser empleado en arquitectura, otros, por el contrario, mantenían una forma estable apropiada para sostener una superficie autoportante. Por ejemplo, en los tubos de PVC la mayoría presentaron deformación plástica con grandes radios para la utilización en superficies, sugiriendo la escala de la estructura primaria de espacios pequeños.

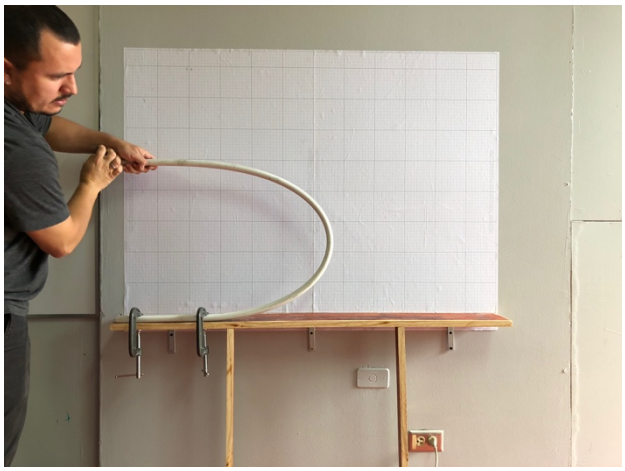

Imagen 41 Pruebas al Tubo de PVC. Fuente: Autores

Imagen 42. Tubo de conduit al fallar. Fuente: Autores

Con el objetivo de mantener el documento enfocado en lo esencial se ha mostrado tan solo una parte del proceso exploratorio de materiales en diferentes configuraciones geométricas.

Lista de Materiales

Para seleccionar el material adecuado se realizó una calificación cualitativa donde se hace una valoración numérica de las principales variables a considerar para

el material adecuado. La calificación 3 es positivo, 2 es neutro o aceptable y 3, negativo.

Tabla 2. Síntesis de la valoración cualitativa de los materiales a ser utilizados.

Material	Disponibilidad	Impacto Ambiental	Costo	Escala para implementación	Desempeño	Tecnología Requerida	Calificación
Barras de Aluminio 3mm Ø	2	2	0	2	1	1	8
Filamentos plásticos	3	0	1	0	2	3	9
Tubos Conduit diámetros variables	3	0	2	3	2	1	11
PVC, diámetros variables	3	0	3	1	3	1	11
Caña Putarrá	1	3	3	0	1	3	11
Caña bambú	2	3	3	1	2	2	13
Cables Plásticos de desecho	3	2	3	3	1	3	15
Cable de Fibra de Vidrio de desecho	3	2	3	3	3	3	17

El material seleccionado fue la fibra de vidrio, este es un material de desperdicio que sobra en cantidades considerables durante las instalaciones de fibra óptica. Debido a que la fibra óptica es sumamente delicada se le inserta dentro del cable un refuerzo de fibra de vidrio en forma de barra para evitar que se doble más de lo debido. Esto es importante ya que es un material diseñado específicamente para resistir esfuerzos de doblado y volver a su posición original. El cable empleado es el residuo de las instalaciones de la empresa JASEC, que utiliza un cable que cuenta con doble refuerzo, es decir contiene dos barras de fibra de vidrio a lo largo.

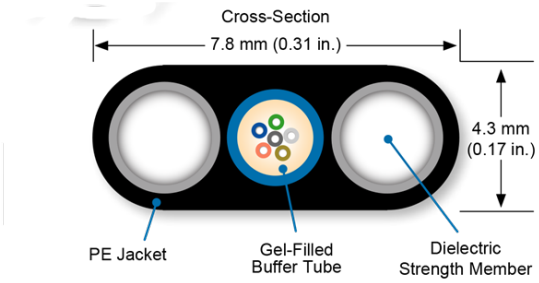

Imagen 43. Ejemplo de cable con barras de fibra de vidrio a ambos lados señalado como "Dielectric Strenght Member". OFS Furukawa Company. (2018). OFS 2 Count Mini LT Flat Drop Cable.

Imagen 44. Cable de fibra óptica mostrando los dos refuerzos de fibra de vidrio en los extremos en color blanco. Fuente: Autores.

El material tiene una alta disponibilidad debido a que se producen gran cantidad de residuos de cable durante las instalaciones de las empresas de telecomunicaciones.

A pesar de que la fabricación de la fibra de vidrio es altamente contaminante, su reutilización reduce su huella ambiental y contribuye a un uso más eficiente de las materias primas residuales.

La resistencia de la barra al doblado es alta y los radios generados son manejables para la escala de la estación de medición y los espacios de trabajo. Sin embargo, se debe manipular con guantes y anteojos ya que al deshilacharse durante la extracción del cable los residuos microscópicos producen comezón en la piel al entrar en contacto con la misma.

Las herramientas necesarias para extraer el material, manipularlo y cortarlo son de uso común. El material es un compuesto de fibras de vidrio y resina epóxica, es anisotrópico y sumamente homogéneo en sus dimensiones. Además, es resistente al agua y la corrosión, no conduce la electricidad, es de alta calidad, y se trasporta y almacena fácilmente.

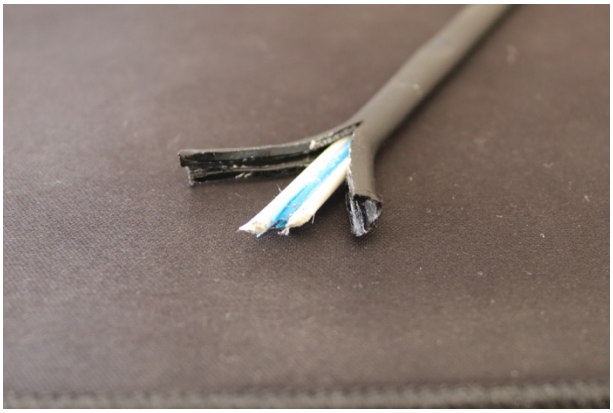

Imagen 45. Cable de fibra óptica mostrando los dos refuerzos de fibra de vidrio con la camisa del cable cortada y abierta. Fuente: Autores

En el mercado no existe una demanda considerable para los residuos de cable de fibra óptica con barras de fibra de vidrio por lo que no tiene un precio de venta definido y las empresas lo regalan.

El material se extrae con relativa facilidad después de hacer un corte preciso en el centro del cable, debido a que la camisa interna que recubre la fibra óptica es más suave que los refuerzos, el bisturí tiende a mantenerse en el centro del cable dañando la fibra óptica mas no los refuerzos.

8.3. Valoración de las técnicas de simulación en relación al desempeño material

Las técnicas de simulación descritas en el primer capítulo resaltaron las posibilidades de uso y características o potenciales de las mismas. Si bien es cierto, cualquiera de ellas puede tener una aplicación bastante buena, el desempeño del material seleccionado se utiliza como variable principal para seleccionar las herramientas a utilizar.

Para los intereses del proyecto, la comunicación con otras herramientas de modelado y manipulación de datos es importante. La posibilidad de la biblioteca iGeo de exportar los elementos del espacio tridimensional en el formato Rhinoceros resulta de gran utilidad ya que ésta es una herramienta para la cual la Universidad cuenta con licencias de instalación. Este es el medio de conectividad más evidente, sin embargo, dado que la biblioteca está programada en lenguaje Java, queda abierta la posibilidad de utilizar cualquier otra biblioteca para exportar datos de geometrías tridimensionales.

En general la experiencia con la biblioteca iGeo fue satisfactoria. La mayor utilización que se le ha dado hasta el momento es en la creación de una abstracción que permita simular materiales flexibles como tubos PVC, varillas de metal o incluso varillas de madera.

Se observa que la biblioteca tiene habilitado por defecto la interacción por medio de mouse y teclado para el manejo de perspectivas y movimiento de cámaras. Esto es diferente a otras bibliotecas del mismo tipo en donde debe hacerse manualmente.

Durante esta exploración se aprovecharon las diferentes abstracciones de la biblioteca para generar una abstracción nueva que permite la creación de una estructura alargada que responde a fuerzas que actúan sobre ella, como la gravedad. La estructura tiene un grado de tensión que hace que tienda a permanecer en línea recta.

Al experimentar con diferentes configuraciones de la estructura flexible, se nota que existen casos que aparentemente el motor de física de la biblioteca no puede manejar y los resuelve eliminando elementos del espacio tridimensional. Esto se hace evidente cuando la tensión de la estructura flexible se pone con un valor alto y la estructura vibra. Si la vibración es muy alta, algunas de las partículas utilizadas en la estructura son eliminadas automáticamente del espacio. Se estima que esto sucede cuando muchas fuerzas actúan sobre una misma partícula y el motor no logra resolver su comportamiento.

Siguiendo con la exploración del material flexible, se desarrolló una nueva abstracción basada en la anterior que consiste en un sistema de estructuras lineales flexibles que pueden unirse en ciertos puntos dadas algunas reglas de proximidad. La idea general es que, si dos estructuras se acercan hasta cierta distancia, las estructuras se unan en cierto punto y así se pueda

observar el comportamiento que surge a partir de esta unión.

Esta segunda abstracción aumenta el nivel de complejidad y expuso todavía más la limitación en cuanto al manejo de fuerzas "conflictivas" por parte del motor de física, ya que al unir y relacionar múltiples estructuras flexibles vuelve a ocurrir el problema mencionado anteriormente.

Tensegrity

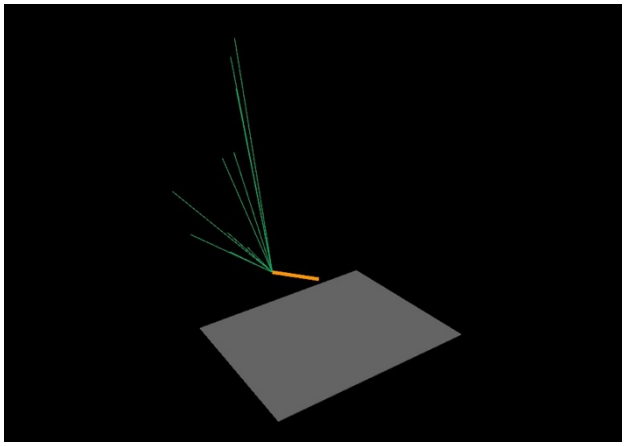

Imagen 46 Ejercicio simple de mantener en tensegity una barra. Fuente: Autores.

La experiencia al explorar ToxicLibs ha sido positiva en su mayor parte, aunque sí se han encontrado ciertos inconvenientes y limitaciones. Principalmente se ha utilizado para intentar simular el concepto de "tensegrity" y también para el modelado de una estructura formada por tubos de PVC y cuerdas.

Una de las limitaciones de la biblioteca es en el aspecto visual, ya que las clases no cuentan con una función para ser mostradas visualmente, sino que esto tiene que hacerse de manera manual para cada abstracción. La solución a esto durante la utilización fue la creación de clases que heredan de las clases de correspondientes de ToxicLibs, con un método de llamado "***display***" el cual no recibe parámetros y dibuja la partícula o resorte basado en su posición actual.

Otra limitación de la biblioteca es la falta de una herramienta de cámara para poder manejar el ángulo en el que se observa la simulación, así como el acercamiento y rotación con el que se quiere ver la misma. Para esto, se utilizó una biblioteca llamad Peasy Cam, la cual tiene todas estas funcionalidades y se puede integrar fácilmente en cualquier ambiente tridimensional de Processing, más adelante se hará una revisión más detallada de esta biblioteca.

Durante la exploración del concepto de tensegrity se llegaron a algunos casos en donde el motor de física remueve algunas de las partículas y resortes unidos a ellas del ambiente de la simulación cuando se llega a algún caso en el que no se pueda calcular su comportamiento, durante la exploración, sin embargo, solo se presentaron estos casos cuando se trabajó con las clases tridimensionales de la biblioteca.

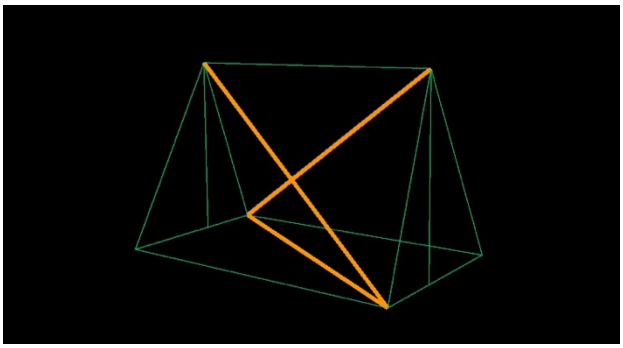

Imagen 47. Sistema de tres barras tensadas por sus extremos para lograr un estado estable. Fuente: Autores.

La mayor limitación encontrada durante la exploración de la biblioteca fue durante la exploración de tensegrity, fue al simular un resorte rígido en 3d, ya luego de su inicialización al aplicarle una fuerza hacia alguna dirección, por ejemplo, sostener otro resorte rígido que está moviéndose hacia abajo por la fuerza de gravedad, el mismo no se mantiene rígido, sino que se continúa estirando conforme el otro se mueve hacia abajo. Si bien es posible que esto pueda ser un error de programación y no de la biblioteca, se hicieron varias pruebas distintas con diferentes clases, incluso se utilizó una especialización de la clase Spring la cual tiene un valor máximo que no puede sobrepasar, pero el resultado fue el mismo. Al hacer las mismas pruebas en un ambiente de dos dimensiones no se tuvo este problema.

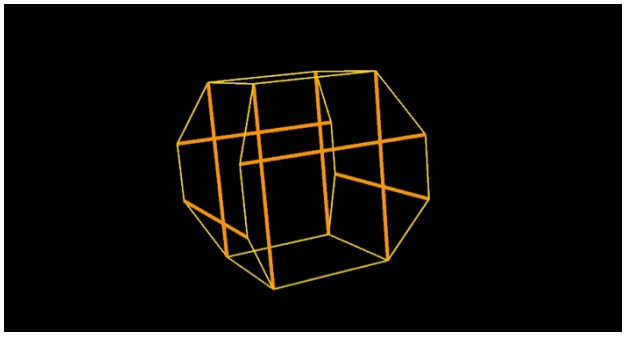

Imagen 48. Sistema de 8 barras en tensegrity. Fuente: Autores

Sistemas ténsiles

Para esta exploración inicial se propuso partir de una forma tridimensional definida y simular su comportamiento. La forma propuesta tiene como propósito servir de estructura principal de un toldo para tapar el sol a los asistentes de un evento al aire libre. La intención era generar experiencia en la herramienta de programación elegida y determinar su aptitud para el resto del proyecto mediante un ejemplo real.

La forma está pensada para ser construida a partir de un material flexible, más específicamente, a partir de tubos de PVC. Los tubos cuentan con cierto grado de flexibilidad que permite deformarlos de su forma original en línea recta. La manera de construir la forma sería iniciando con un círculo creado a partir de la concatenación de un número de par de tubos. Seguidamente, por medio de líneas de tensión situadas en el diámetro del círculo, se crea una forma similar a un ocho. Luego se agrega una línea de tensión adicional entre los dos puntos más distantes de la forma generada en el paso anterior, esta línea de tensión genera una concavidad hacia arriba que le da tridimensionalidad a la estructura.

Primeramente, se intentó modelar la estructura partiendo de una serie de puntos predefinidos, aunque este acercamiento no dio los resultados esperados. Se crearon una serie de líneas de tensión que unían los puntos especificados, los puntos fueron unidos mediante otra línea de tensión a una partícula con posición fija en la posición original de los puntos del modelo. El tubo flexible quedaba modelado a través de las líneas de tensión, cuyo comportamiento sería el de tratar de acercarse entre ellas. Los anclajes hacían que la figura mantuviera su posición cercana al modelo de partida.

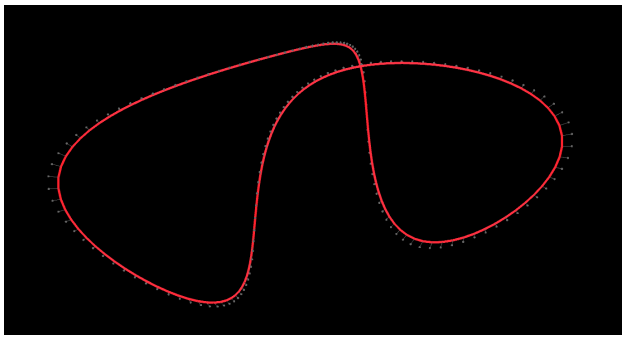

Imagen 49. Elaboración de un marco rígido no simulado. Fuente: Autores

Sobre estas líneas se simularon textiles con una serie de líneas de tensión y partículas ordenadas en forma de una cuadrícula bidimensional. Los textiles se distribuyeron uniformemente y con diferentes configuraciones. Esto mostró deformaciones en los puntos de unión con la línea principal.

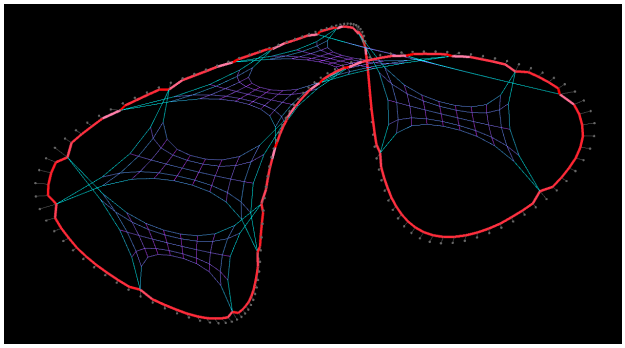

Imagen 50. Marco rígido con un sistema de ténsiles amarrados. Fuente: Autores

Esta forma de modelar el comportamiento del tubo flexible no se acercaba al comportamiento del elemento real debido a que se estaba forzando artificialmente la posición en el espacio tridimensional. Tampoco la forma es un resultado de la manipulación del material, que es parte de lo que se busca en el proyecto.

Sin embargo, hubo varios logros positivos luego de esta exploración:

1. Se genera experiencia en la utilización de la biblioteca iGeo en Processing 2.

2. Se corrobora que la lectura de datos externos (puntos predeterminados de la forma) es relativamente fácil.

3. El comportamiento de los textiles a partir de líneas de tensión sí asemejan formas vistas en el mundo real.

8.4. Sistemas digitales a utilizar según el comportamiento elegido

Barras flexibles

En la búsqueda por simular un material flexible como un tubo de PVC, y partiendo del trabajo realizado en la exploración anterior, se observa que el problema puede reducirse a una serie de partículas distribuidas de forma lineal que tienden a permanecer en línea recta. Existirán fuerzas que afectan a las partículas, desalineándolas de su posición original, como la gravedad, tensiones o interacción con otros elementos, y en respuesta, la estructura debe generar fuerzas que llevan a las partículas a permanecer alineadas entre ellas.

En la biblioteca iGeo se encontró con la clase *IStraightener* que resuelve la versión más simple de este problema, que consiste en modelar tres partículas alineadas que intentan permanecer en línea recta cuando se aplican fuerzas sobre ellas. Podría pensarse en ella como una especie de enderezador. Esta clase se inicializa a partir de tres partículas posicionadas en el espacio tridimensional y, entre otras cosas, un valor de tensión que define la magnitud de la fuerza que trata de mantener a las partículas en línea recta.

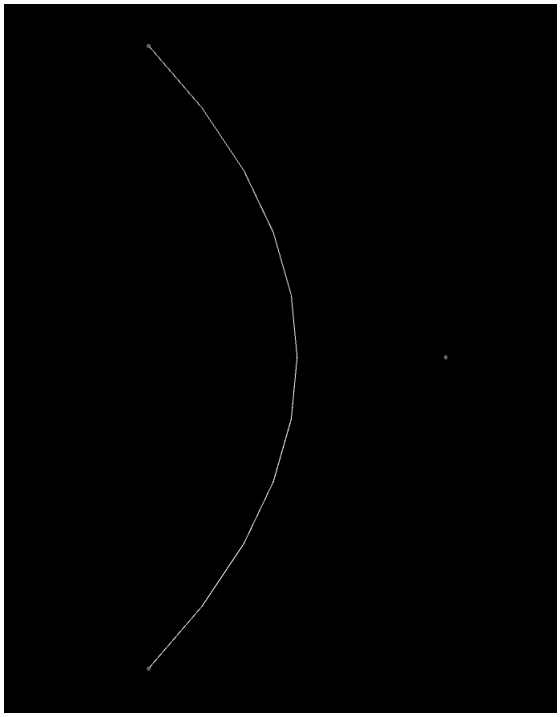

Imagen 51. Primera prueba del uso de la biblioteca simulando el comportamiento flexible. Fuente: Autores

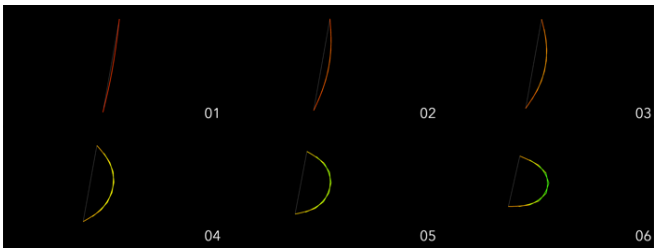

Imagen 52 Secuencia de la simulación de una barra flexible programada en Processing. Fuente: Autores

Este comportamiento tiene similitudes con el comportamiento de un material flexible, pero tiene una diferencia importante que es que la distancia entre las partículas puede aumentar. La clase *IStraightener* ejerce fuerzas sobre las partículas para mantenerlas alineadas, pero no toma en cuenta la distancia entre ellas. Para acercar este comportamiento al buscado, es necesario que las distancias entre las partículas se mantengan constantes.

Para mantener las distancias entre las partículas, se utiliza la clase *IStickLine*, que es una abstracción que representa una conexión fija entre dos partículas. Impide a las partículas aumentar o disminuir la

distancia entre ellas. Al combinarlos con la curva *IStraightener*, se obtiene el comportamiento buscado.

Para generalizar este comportamiento a las partículas de la barra flexible, se crea un *IStraightener* por cada grupo de tres partículas contiguas y un *IStickLine* entre cada dos partículas contiguas. De esta forma, cada partícula de la barra se encuentra conectada con otras partículas de varias formas. Todas las partículas se encuentran conectadas por medio de *IStickLines* con dos partículas vecinas, las de los extremos sólo con una. También estarían conectadas con otras partículas a través de una a tres curvas enderezadoras.

En la imagen 50 se observa una barra flexible cuya posición original es horizontal. Las primeras dos partículas se encuentran fijas, mientras que el resto se encuentra sujeto a una fuerza de gravedad que las empuja hacia abajo. Puede observarse las *IStraigtenerCurve* y *IStickLine* que mantienen la forma de la estructura.

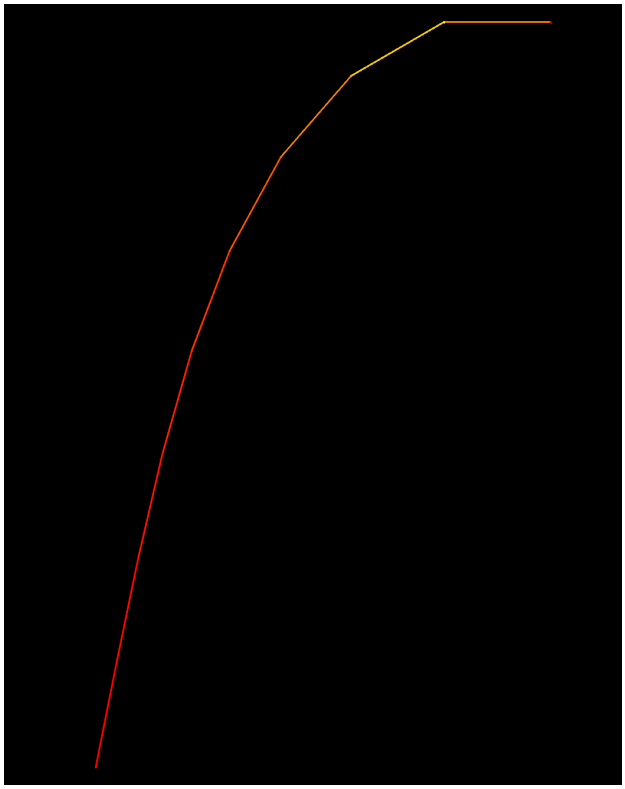

Imagen 53. Marra anclada en el extremo superior derecho, deformada por la fuerza de la gravedad. Fuente: Autores

Con el fin de visualizar la barra como una unidad, se ocultan las curvas y se muestran sólo las líneas que unen las partículas. Por otro lado, con el fin de visualizar el nivel de deformación de la barra, se calcula el ángulo que forma cada partícula con sus partículas vecinas y se asigna un color a la partícula de acuerdo a este valor. Las líneas que unen las partículas también alteran su color realizando una mezcla entre los colores de las partículas que une.

Una siguiente idea para probar el comportamiento de la barra flexible que se usó fue la de conectar los extremos de la barra con un *spring* que dobla la barra. La posición inicial de la barra es en línea recta. Si la barra es perfectamente recta, el *spring* no ocasiona ningún efecto en la barra ya que las fuerzas se balancean perfectamente. Pero si la barra se crea con una ligera curvatura, la barra se dobla ante el efecto del *spring*.

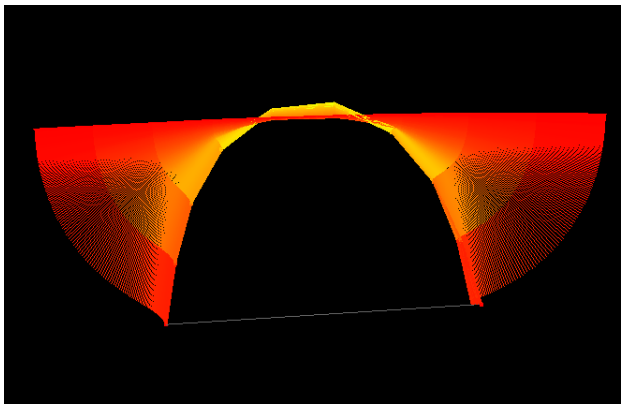

Imagen 54. Deformación de una barra desde sus extremos con el rastro de la animación. Fuente: Autores

En la figura 51 se ilustra el movimiento de la barra mediante un rastro. El color muestra que la región que más se flexiona en la barra es cerca del centro. En este escenario no se utiliza gravedad.

Se experimentó con la forma buscada en la primera exploración, creando una barra flexible circular con un *spring* en su diámetro, conectando dos partículas en lados opuestos. Luego utilizar un nuevo *spring* el un diámetro perpendicular al anterior para generar la segunda curvatura. Nuevamente es necesario curvar

previamente la barra flexible para que se puedan generar las deformaciones.

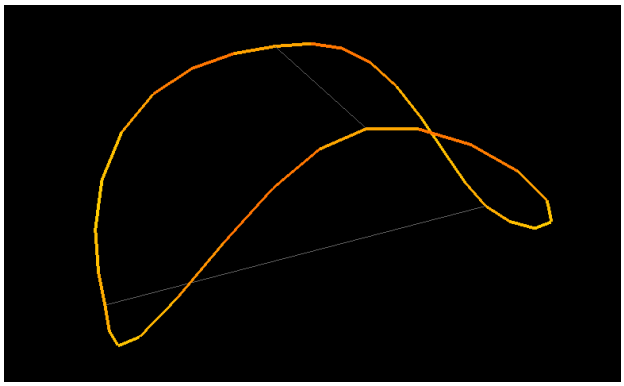

Imagen 55. Degormación de un circulo mediante tensores simulando el marco realizado en la imagen 49.. Fuente: Autores

En la siguiente imagen se observa el rastro del movimiento de la barra conforme se genera la forma. El rastro refleja por medio de colores la cantidad de flexión que ha sufrido la barra a través del tiempo. En ausencia de gravedad fue posible generar la forma propuesta en la exploración anterior. Aunque la forma resultante no coincide exactamente con la prevista, sí tiene un comportamiento muy similar.

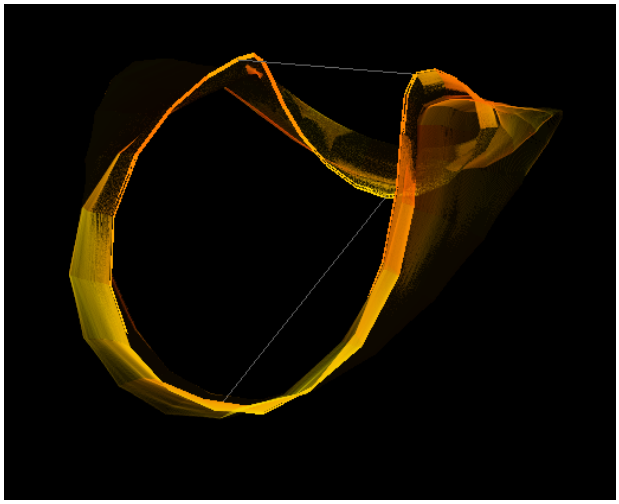

Imagen 56. Simulación con rastro activado y transparencia. Fuente: Autores

Se exploraron diferentes configuraciones basadas en una barra circular. Se ubicaron diferentes ***springs*** que ejercieran fuerza sobre la barra circular, anclados en diferentes puntos de la misma, con el propósito de observar las formas emergentes de cada disposición.

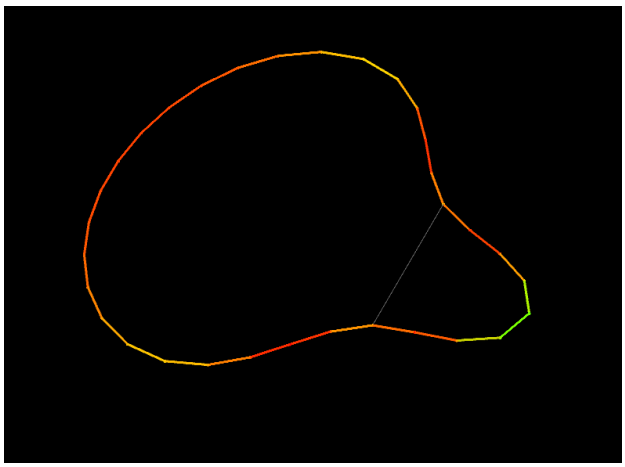

Imagen 57. Degormación de un circulo mediante tensores. Fuente: Autores

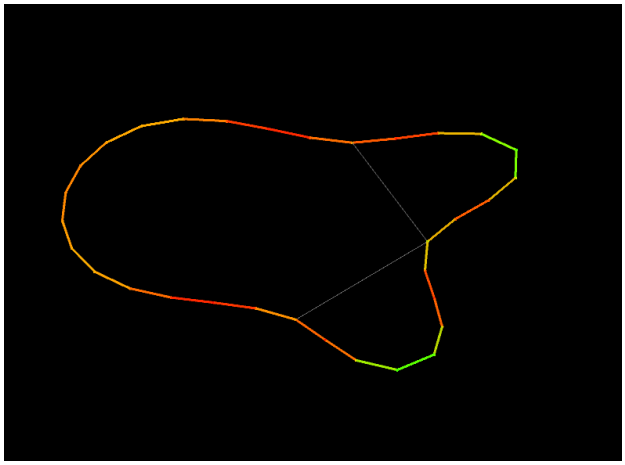

Imagen 58. Degormación de un circulo mediante tensores. Fuente: Autores

Estas formas parten de un anillo perfectamente plano, por lo que la deformación se da en dos ejes y el resultado también es bidimensional.

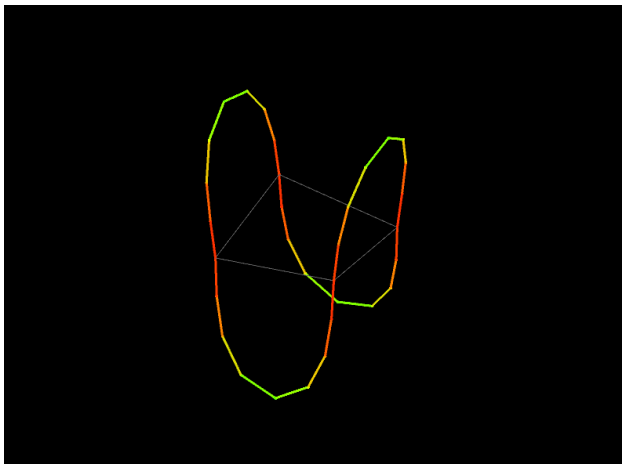

Imagen 59. Fuente: Autores

Al partir de un anillo que no fuera perfectamente plano, las formas resultantes adquirieron tridimensionalidad. Esto abre mayores posibilidades y en la mayoría de los casos producía resultados inesperados o que no eran tan evidentes.

Barras "pegajosas"

Otra de las ideas implementadas fue la de modificar las barras flexibles para que se unieran en una misma estructura de acuerdo a una proximidad definida. De esta manera, las barras que estuvieran a cierta distancia límite o menos, se unirían compartiendo una de sus partículas. Por este motivo, se consideraron como barras "pegajosas".

La programación de esta característica representó un reto mayor de programación al que se había encontrado hasta el momento, ya que requería la consideración de una serie de casos. Por ejemplo:

- Una barra no debería unirse con sí misma.
- Una barra que ya se unió con otra barra no debería unirse en otro punto con la misma barra a menos que la nueva unión se encontrara a determinada distancia de la unión anterior.
- Considerar un radio de unión, si dos partículas de barras diferentes se encuentran dentro de esa distancia, las barras deberían unirse en ese punto.
- Considerar un radio para uniones nuevas, si dos barras se encuentran unidas en un punto, y otras de sus partículas se encuentran dentro del radio de unión, la unión podrá hacerse sólo si están fuera del radio para uniones nuevas.
- Las uniones deben implementarse haciendo que una barra sustituya una de sus partículas por la partícula de otra barra, por lo que deberían actualizarse todas las referencias utilizadas en los IStickLine y IStraightener utilizados dentro de la estructura.

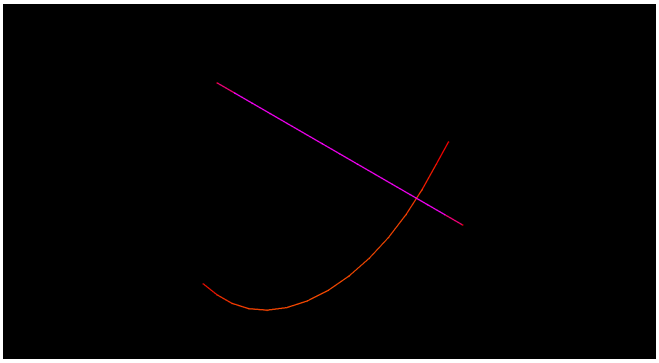

Imagen 60. Dos barras antes de colisionar. Fuente: Autores

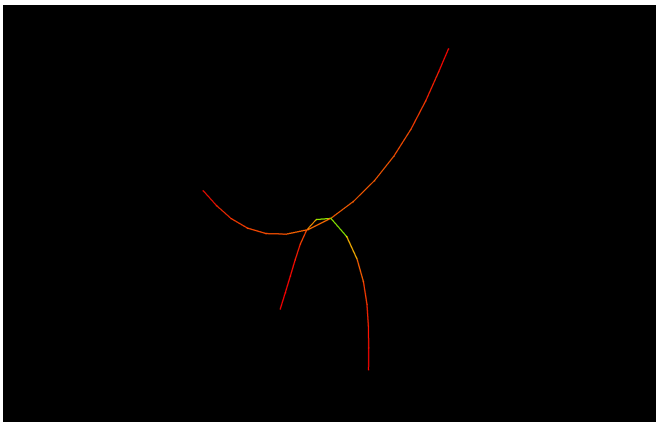

Imagen 61. Dos barras colisionando. Fuente: Autores

En las imágenes 60 y 61, se puede observar una barra con sus extremos fijos, mientras que otra barra totalmente horizontal se deja caer sobre ella. Al cumplirse las consideraciones, se crea una unión en forma de cruz y la segunda barra queda suspendida a partir de la primera.

Para realizar los cálculos necesarios, cada barra debe tener conocimiento de cuáles otras barras se encuentran a su alrededor y la posición de sus partículas. Esto creó la necesidad de una nueva abstracción que represente un sistema de barras. El sistema se encarga de compartir la información entre un grupo de barras y controlar la interacción entre las mismas.

Esta misma idea se exploró con diferente cantidad de barras para generar estructuras más complejas. Se crearon dos grupos de barras, uno con barras con extremos fijos y otro con barras libres que caen por gravedad sobre las barras del grupo anterior. La posición de las barras del segundo grupo se determinó

de forma aleatoria. Como resultado se obtuvieron estructuras de red con cierto nivel de complejidad, pero siempre manteniéndose en la bidimensionalidad.

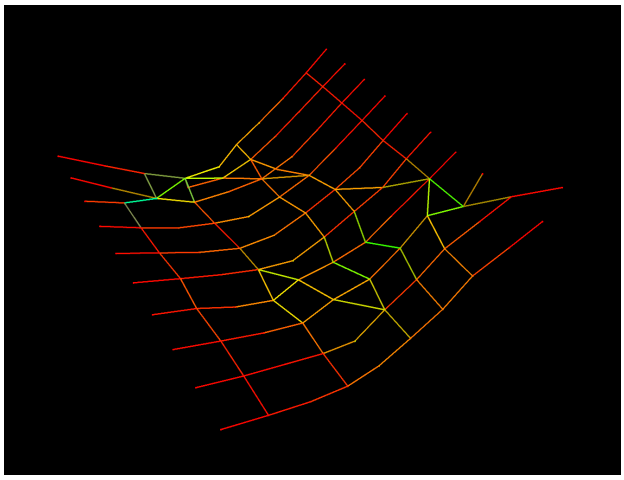

Imagen 62. Resultado de la caída de un conjunto de siete barras sobre una serie de diez barras con puntos fijos. Fuente: Autores

En esta imagen se puede observar el resultado de la caída de un conjunto de siete barras sobre una serie de diez barras con puntos fijos.

En este punto se encontró un problema que provocaba que el programa se detuviera en algunos casos cuando se daba una alta cantidad de conexiones entre barras. Aunque se depuró el código en busca del error que causara referencias nulas, este problema no fue posible solucionarlo. Se determinó que podría ser una limitación de la misma biblioteca, o un error de programación o caso no considerado. Por este motivo, este tipo de barras no fueron utilizadas posteriormente en el proyecto y se optó por las barras flexibles simples.

Atracción entre barras

Como continuación de la exploración anterior, también se propuso la creación de barras que se atrajesen entre ellas, de forma que se acercaran según su proximidad y eventualmente se unieran para formar una sola estructura.

Para implementar esta atracción entre barras se utilizó el modelo gravitatorio de Newton, donde la fuerza de atracción depende de la masa de cada uno de los objetos y es inversamente proporcional a la distancia entre los objetos.

Para lograr esta interacción entre las barras, se utilizaron las partículas que unen sus diferentes segmentos. Estas partículas deben atraerse entre ellas siempre y cuando no pertenezcan a la misma barra. Por lo tanto, cada barra debe conocer cuáles barras están a su alrededor, determinar la fuerza de atracción que existirá entre cada par de partículas y aplicar la fuerza en la dirección calculada. Esta interacción también es manejada por el sistema de barras descrito anteriormente.

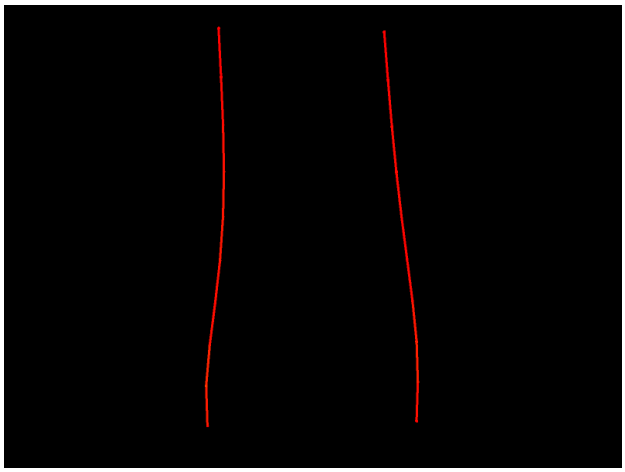

Imagen 63. Dos barras flexibles que deben unirse de acuerdo a parámetros definidos (de cercanía y por los extremos). Fuente: Autores.

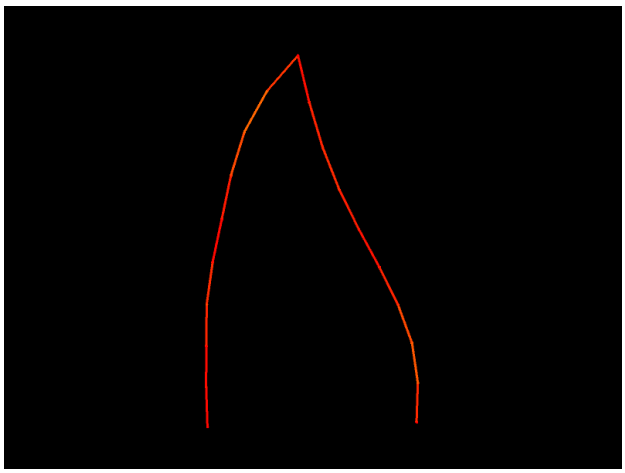

Imagen 64. Unión de dos barras a partir de una atracción generada para emular una desición humana.Fuente: Autores.

En las imágenes anteriores se muestra un ejemplo de barras que se atraen y se unen en una misma

estructura. Se ubicaron dos barras rectas de forma vertical y paralela, con una base fija. La fuerza de atracción vence a la tensión de las barras y las mismas se doblan. En la segunda imagen se observa como la cercanía activó la unión de las dos barras en sus extremos.

8.5. Sistemas tridimensionales para prototipar

Las simulaciones del comportamiento flexible en barras planas, permite la posibilidad de crear u amplia gama de formas al aplicar una serie de operaciones físicas tales como doblar, unir, cortar, presionar la barra, así como unirlas para formar módulos que generen configuraciones mayores. Para ello, se inició comprobando la capacidad del material en computar un arco.

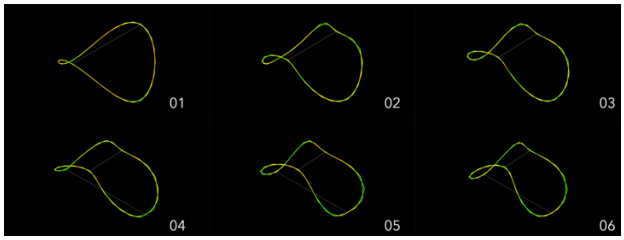

Imagen 65 Secuencia de la simulación de un círculo tensado en dos puntos programado en Processing. Fuente: Autores.

El PVC de media pulgada de diámetro sirvió de manera preliminar para valorar las posibilidades de fabricación de un posible sistema mayor. El espesor del material determina la escala geométrica que puede generar. Por ejemplo, con un tubo de tres metros de largo servía para generar un circulo con la suficiente tensión para mantener su forma, como se ve en la imagen 63.

Imagen 66. Armado de un círculo a partir de un tubo flexionado de PVC. Fuente: Autores.

Al aplicar unos puntos de compresión en extremos opuestos, se puede deformar el circulo para lograr una estructura en tres dimensiones similar a un arco. (imagen 64). Al ser una estructura curva, no ese estable en sí misma, por lo que requiere de anclajes o amarras para mantener su posición, sin embargo, al acoplarlo a otro modulo igual, se logran cuatro puntos de apoyo mejorando considerablemente la estabilidad.

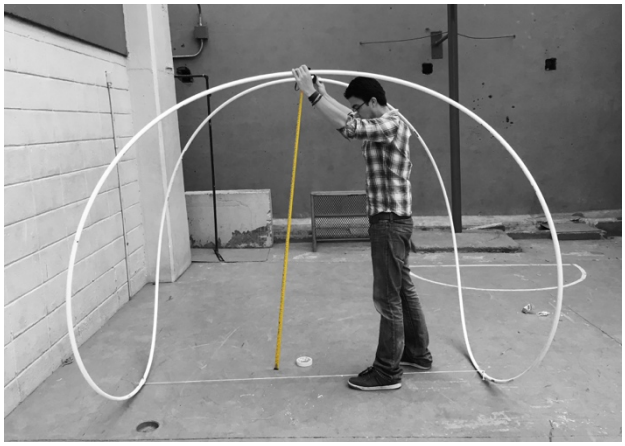

Imagen 67. Fuente: Autores.

Esta forma lograda por el material flexible serviría como soporte para un material ligero que sirviera de superficie para generar sombra. Se utilizan para ello diferentes opciones, partiendo de un material de lona vinílica de desecho producto de vallas publicitarias desmontadas. Este material resultaba prometedor especialmente por ser un material de segundo uso, y por presentar una opacidad al sol del 100%. Sin

embargo, representó una dificultad el peso que representaba para la estructura, y además del sistema de sujeción a la estructura.

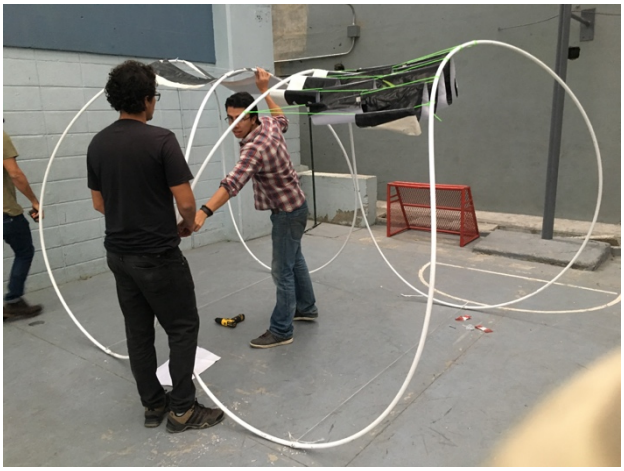

Imagen 68. Prototipos en tubo de ½ pulgada, con propuesta de uso de lona en abanico. Fuente: Autores.

La creación de un tejido de lonas sobre la estructura flexible generaría además del peso, un efecto de vela al verse expuesta al viento en un espacio exterior, por lo que se prueban patrones no continuos que permitieran el paso de la brisa entre las superficies (imágenes 66 y 67). El prototipo a esta escala resultó convincente, pero en espacios interiores controlados. Se debió pensar en otro material más ligero para las superficies.

Imagen 69. Prototipos en tubo de ½ pulgada, con propuesta de uso de lona entre cruzada.Fuente: Autores.

Imagen 70 Tejido de lona generado en la estructura de PVC. Fuente: Autores.

Dentro de las expectativas del proyecto, estaba el determinar a qué escala aplicar el sistema material, siempre considerando que el diseño debía servir para las dimensiones de una superficie arquitectónica. Por ello se exploró la posibilidad de escalar el prototipo anterior aumentando el calibre y el tamaño del diámetro de los tubos en PVC. Aumentar cuatro veces las dimensiones utilizadas para el prototipo, nos permitirían escalar al doble las dimensiones del módulo.

El uso de PVC de dos pulgadas de diámetro computó perfectamente el círculo y el arco tridimensional realizado en el prototipo. (imágenes 68 y 69), y se utilizó una tela de licra blanca muy ligera, que, si bien no filtraba al cien por ciento la luz solar, si tamizaba la incidencia del sol. (imagen 70).

Imagen 71. El sistema se prueba en escala mayor, con tubos de PVC de 2"Ø. Fuente: Autores.

Imagen 72 Prototipo en 4 Tubos de PVC de 2" Ø. Fuente: Autores.

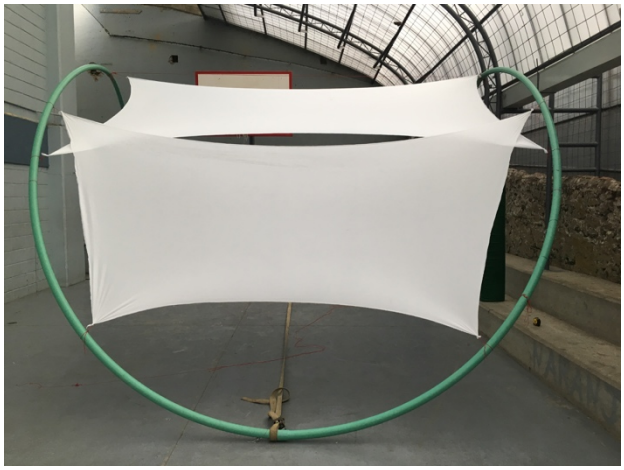

Imagen 73. Primera prueba con licra. Fuente: Autores.

La prueba final de este ejercicio, consistía en probar el módulo en una serie de eventos que requerían la necesidad de cubrir una pequeña área, sustituyendo el típico toldo de feria.

Para el primer evento que se instaló el módulo, se comprobó la simplicidad de la generación de la forma, producto de las cualidades del propio material. Se contó con la participación de cinco personas para el montaje y ensamblaje de la estructura base, el cual se desarrolló en menos de una hora. Sin embargo, la instalación de las lonas generó una serie de contratiempos, especialmente generados por el viento presente en el lugar. Esto hizo muy flexibles las dos estructuras que se ensamblaron por lo que fue necesario tensar las estructuras a anclajes improvisados y eliminar algunas de las telas que se ensamblaron, especialmente las que servían como velas ante el viento.

Imagen 74. Prototipo ensamblado en Curridabat. Fuente: Autores.

Imagen 75. Dos módulos se pudieron instalar con la asistencia de cinco personas. Fuente: Autores.

Posteriormente y aprovechando el evento de Amón Cultural, se pudo probar el módulo a partir de las experiencias aprendidas. Con la colaboración de nueve personas, se pudo armar tres módulos para mejorar la estabilidad del sistema, y las uniones entre módulos se optimizaron por medio de gazas plásticas. Se plantearon mejores anclajes a estructuras existentes y se colocaron las telas en relación a la incidencia solar.

Sin embargo, el viento que se presentó el día de la exhibición, evidenció nuevamente que la estructura es muy flexible, por lo que la sensación debajo de la estructura era de cierta incomodidad o inseguridad. Aún con esta molestia, se cumplió con el objetivo de

minimizar la incidencia solar para los asistentes del evento.

Imagen 76.Istalación del sistema en Barrio Amón. Fuente: Autores.

Las experiencias realizadas a escala real sirvieron para replantear el uso de barras flexibles en la generación de pieles arquitectónicas.

Determinamos que a pesar de que el objetivo es la escala arquitectónica, las partes que componen la piel no necesariamente tienen que ser de dimensiones grandes. Así que se empezó a generar configuraciones tridimensionales con otros materiales que a escala pequeña se desempeñaran de forma adecuada. De la valoración material antes mostrada (Tabla 2) se hace evidente que los mejores materiales son el filamento plástico y las barras de fibra de vidrio, por lo que se empezaron a utilizar en posibles usos y formas. En la imagen 75 se aprecia por ejemplo una configuración tridimensional compleja, utilizando el filamento plástico con únicamente tres puntos de unión. La forma resultante es resultado de la computación del material según las operaciones realizadas, esto es, dejando que el material se exprese en el diseño (Manuel DeLanda).

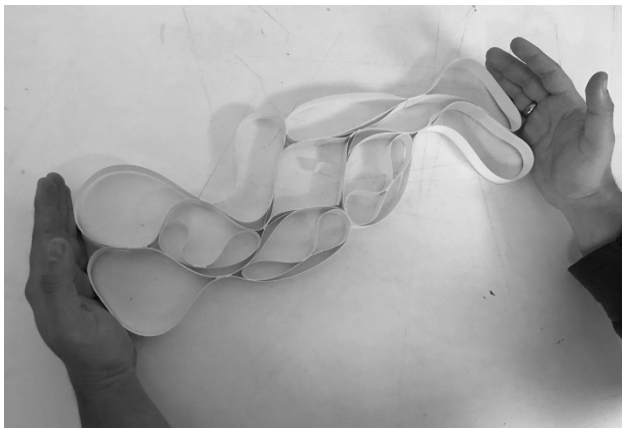

Imagen 77. Muestra de otros materiales en comportamiento flexible. Fuente: Autores.

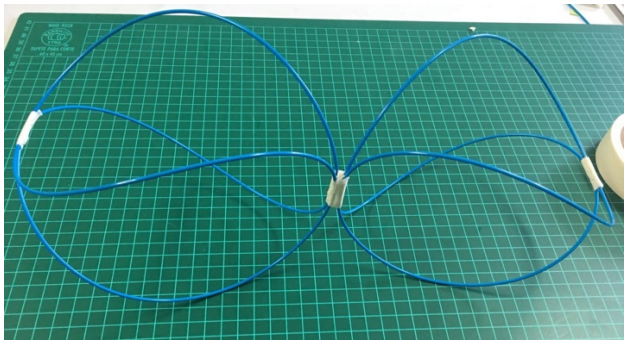

Imagen 78. Prototipo de polímero azul mostrando configuraciones geométricas alternativas. Fuente: Autores.

Como conclusión de esta serie de exploraciones, se define que el material que se va a utilizar en el diseño y las simulaciones digitales, será la barra de fibra de vidrio, y que la escala a trabajar es la que permita el propio material, por lo que el tamaño de los módulos será identificado como una resultante del proceso de sistematización del análisis del material.

8.6. Extracción de datos del material y definición del sistema

Antes de explicar el proceso de extracción de datos es importante aclarar que este es un proceso desarrollado de forma empírica ante la necesidad de medir un material. El enfoque de este proyecto de investigación es el Diseño Computacional y no la Ingeniería en Materiales, por lo que, en este punto lo más importante es la efectividad y la precisión lograda desde las implicaciones geométricas y formales.

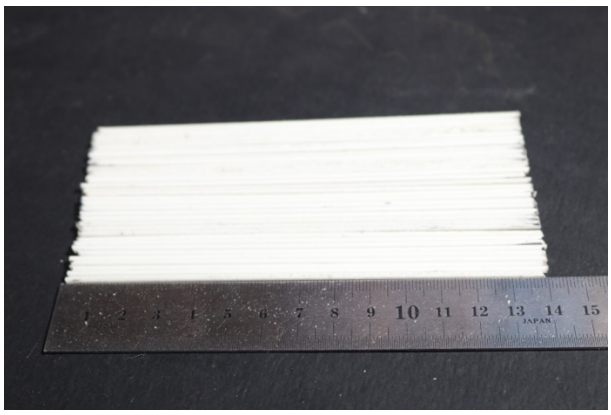

Imagen 79. Barras de fibra de vidrio cortadas a 14cm de longitud. Fuente: Autores.

Las barras de fibra de vidrio tienen aproximadamente 1,92mm de diámetro cada una y se cortaron a 140mm de longitud haciendo un corte diagonal con un taladro manual de precisión a 10.000 r.p.m. para evitar deshilachar el material en los extremos. La velocidad del taladro derrite ligeramente la resina de la barra sellando los extremos. Posteriormente, para proteger los extremos de las puntas de los alicates se les envolvió en cinta adhesiva.

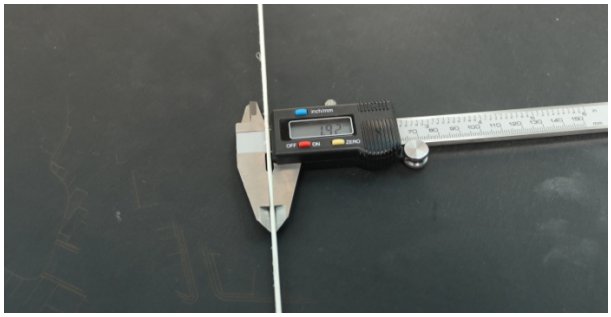

Imagen 80. Calibrador usado para medir los diámetros de las barras de fibra de vidrio. Fuente: Autores.

Imagen 81. Cortando una barra de fibra de vidrio diagonalmente a 10.000 r.p.m. para sellar los extremos con el calor del disco y evitar que se deshilache. Fuente: Autores.

Adicionalmente, los alicates se modificaron agregando madera de balsa para no producir fisuras en los puntos de sujeción. La balsa, al ser una madera suave, ayudó a sujetar con fuerza sin generar daño en la barra.

Imagen 82. Alicate forrado con madera de balsa para proteger los extremos de la fibra de vidrio. Fuente: Autores.

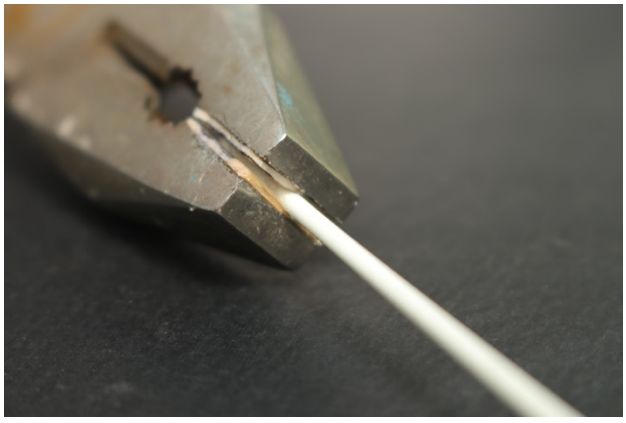

Imagen 83. Alicate forrado con madera de balsa sujetando el extremo de la barra de fibra de vidrio. Fuente: Autores.

Estación de Medición

La estación de medición consiste en un plano de madera vertical con una grilla ortogonal de referencia adherida a la superficie y dos cámaras que graban en

dos ángulos diferentes el momento en el que un material falla al ser doblado. Las cámaras permiten observar diferentes ángulos cambios en la posición de la barra. El objetivo consiste en mapear tridimensionalmente la curva de la barra al instante en el que falla.

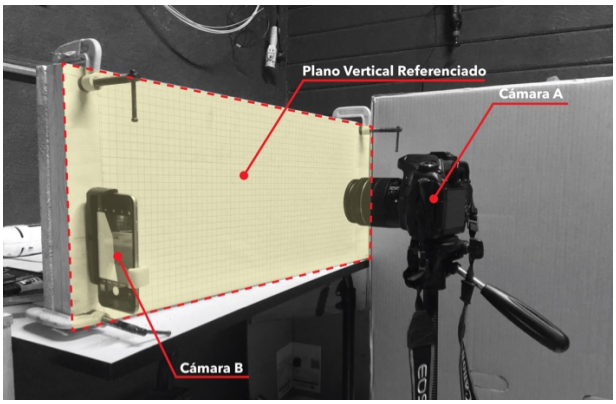

Imagen 84. Estación de medición mostrando las dos cámaras y el plano vertical referenciado con la grilla ortogonal. Fuente: Autores.

Es importante que el material del plano de referencia no sufra expansiones o contracciones térmicas que puedan afectar las mediciones, la madera en este caso tiene coeficientes de expansión térmica despreciables respecto a la precisión del proceso.

Para facilitar es proceso es necesario que las cámaras estén fijas y siempre en la misma posición por lo que se utilizó un trípode para la Cámara A y un anclaje para la Cámara B. El trípode cuenta con un nivel de burbuja, además se marcó con cinta adhesiva en el piso la posición de este para mantener constante su posición. La otra cámara se encuentra anclada de manera fija con un perno al plano vertical que a su vez se niveló de forma convencional.

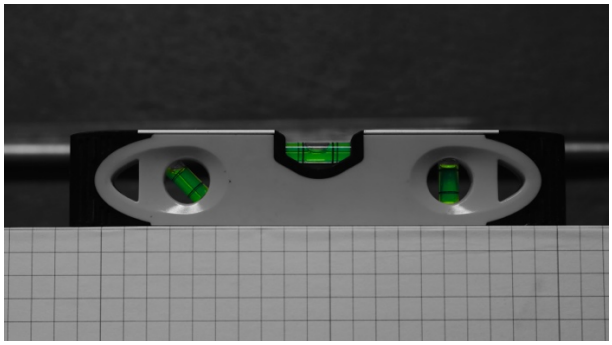

Imagen 85. Comprobación de nivelación convencional del plano vertical. Fuente: Autores.

La grilla de referencia es de 10mm x 10mm y cada 100mm está marca con rojo para mejorar la visibilidad, cerca de la esquina inferior izquierda se le insertaron tornillos para mantener las barras totalmente apoyadas sobre el plano vertical siempre en el mismo punto y para disminuir la posibilidad de movimientos no deseados durante la medición.

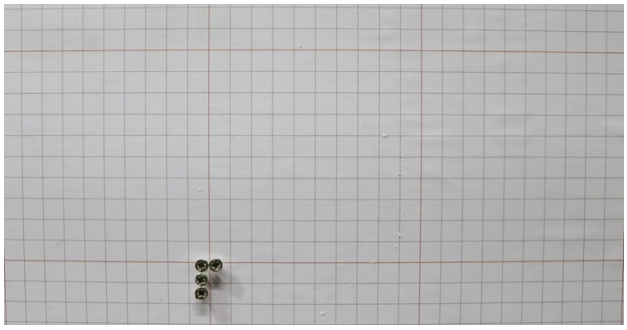

Imagen 86. Tornillos para anclar las muestras durante el proceso de medición. Fuente: Autores.

Sincronización de Audio y Video

Con ambas cámaras grabando cada muestra se insertó verticalmente en el anclaje antes mencionado hasta alinear la marca negra en la barra con línea horizontal roja del plano y se le dobló hacia el lado derecho hasta que se quebrase. Al ser un material compuesto la fibra de vidrio falla de forma paulatina y produce un sonido agudo muy evidente al comenzar a fallar las fibras de forma individual.

Imagen 87. Barra de fibra de vidrio deshilachada después de haber sido doblada. Fuente: Autores.

El sonido específico capturado por los micrófonos de ambas cámaras es el momento preciso en el que la barra comienza a fallar. Debido a que ambas cámaras generan archivos independientes, el primer paso es sincronizar ambas cámaras con la ayuda de algún sonido, en este caso se usó el momento donde falla la fibra de vidrio de manera visual y auditiva ya que produce un pico muy evidente en la curva de sonido y es verificable visualmente. El programa usado permitió comparar en tiempo real las pistas de audio y video para verificar errores.

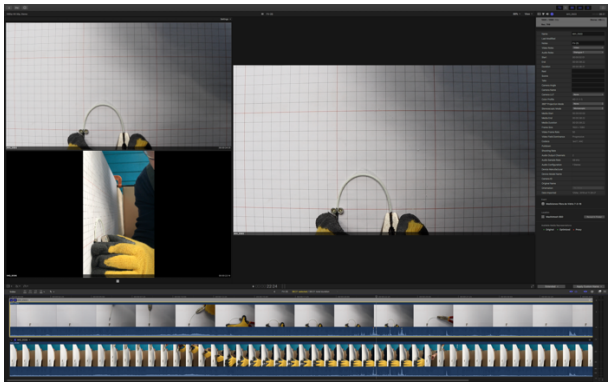

Imagen 88. Proceso de sincronización de audio y video de ambas cámaras. Fuente: Autores.

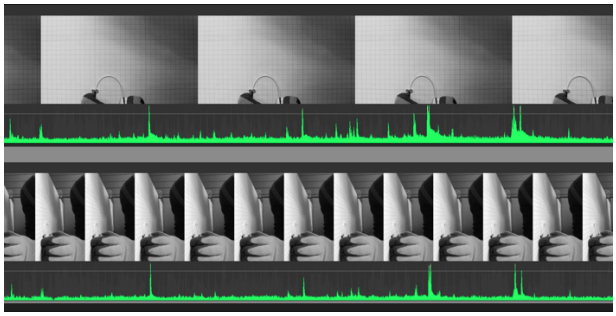

Imagen 89. Detalle de las curvas de sonido de ambas cámaras sincronizadas. Fuente: Autores.

Una vez sincronizadas se procede a escuchar el audio y marcar el momento en el que se escucha la primera falla de la barra. En el punto de la pista donde se marcó la falla, se exporta el cuadro de la pista de video a una imagen fija. Se produjeron entonces dos imágenes fijas, una por cámara, registrando el momento preciso en que falló el material desde dos ángulos diferentes.

Corrección de distorsión

La imagen de la cámara A que grabó perpendicularmente posee una distorsión generada por el lente utilizado, dicha distorsión hace que los objetos verticales y horizontales se deformen de acuerdo con el tipo de lente utilizado. Antes de utilizar la imagen se le debe corregir dicha distorsión.

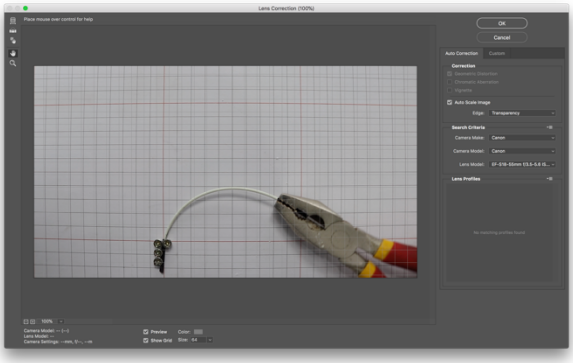

Imagen 90. Imagen de la cámara A sin corrección de distorsión, se observa como las líneas rojas de la fotografía se curvan. Fuente: Autores.

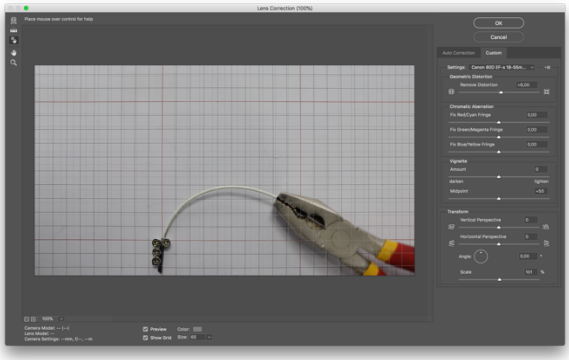

Imagen 91. Imagen de la cámara A corregida, se observa como las líneas rojas de la fotografía se mantienen ortogonales especialmente dentro del cuadrante donde se encuentra la barra. Fuente: Autores.

El mismo proceso se repite con la imagen de la cámara B. El objetivo de la cámara adicional es determinar la rotación de la barra. Se puede observar que la barra tiene una rotación hacia afuera respecto al punto de anclaje. El alicate se encuentra apoyado al plano vertical y sujeta la barra por el centro, lo que quiere decir que el extremo sujetado por alicate se desplaza la mitad del ancho de la cabeza del alicate aproximadamente, desde el plano vertical en dirección hacia la cámara A.

Este proceso de corrección de distorsión se automatizó mediante un "action" en Adobe Photoshop CC que consisten una rutina que se puede reutilizar para automatizar un proceso repetitivo.

Imagen 92. Se puede observar que la barra tiene una rotación hacia afuera respecto al punto de anclaje. Fuente: Autores.

Digitalización 3D de las barras

Con las imágenes debidamente corregidas se procede a digitalizar la posición y la curvatura de la barra en el programa Rhinoceros 3D. La grilla ortogonal del plano vertical permite escalar la imagen y verificar la precisión de la corrección de la distorsión del lente. Utilizando el comando "*interpolate curve*" (<_InterpCrv>) se trazan los bordes de la barra de fibra de vidrio como si se estuviese calcando una imagen.

Las curvas trazadas se proyectan sobre un plano rotado que coincide con el ángulo de rotación de la barra medido en la imagen de la cámara B, en la Imagen 24 se puede observar que la barra tiene una rotación hacia afuera respecto al punto de anclaje, el desplazamiento en la mayoría de los casos es la mitad del ancho de la cabeza del alicate aproximadamente.

Si se mide el diámetro de la barra en el programa, debe coincidir con la medición directa que se hizo previamente, esto es un segundo paso de verificación de errores en el proceso. La medición digital del diámetro se realizó para cada muestra y se encontró el mismo dato en todas las muestras.

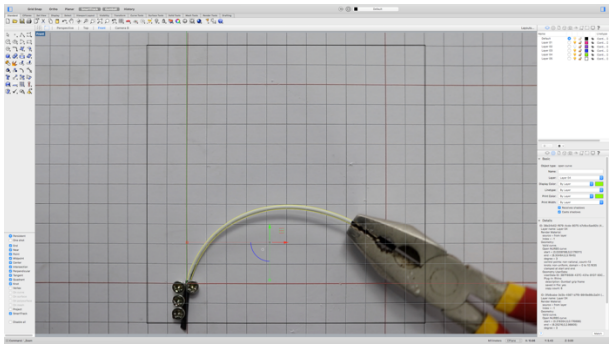

Imagen 93. Imagen escalada correctamente en Rhinoceros 3D con la grilla de la imagen alineada con la grilla del programa. Fuente: Autores.

Análisis de curvaturas

El análisis de la curvatura de cada barra se automatizó mediante una definición de "Grasshopper". La definición requiere de dos curvas y una rotación para generar un radio mínimo y una representación gráfica de los radios cambiantes a lo largo de la curva.

La definición comienza capturando y reconstruyendo las curvas trazadas manualmente con cantidades de puntos variables a curvas con una cantidad constante diez puntos, esto para estandarizar el procedimiento y el grado de precisión del trazo. Después se utiliza un componente para trazar una curva por el centro de las curvas dadas, este representa el eje de la barra. El segundo tramo rota las curvas y las proyecta en un plano inclinado según la rotación ingresada. El tercer tramo analiza la curva y busca el radio más pequeño de la lista, enmarcado en color amarillo al final de la definición en la Imagen 94.

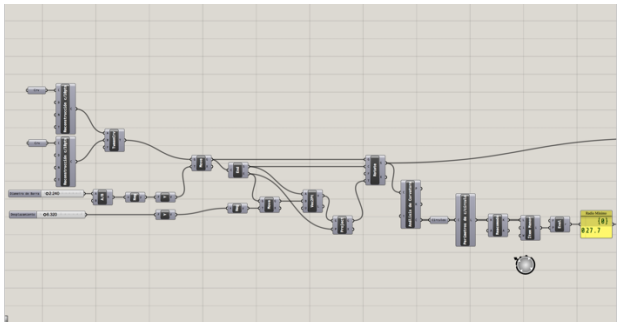

Imagen 94. Captura de pantalla de la definición de "Grasshopper" que analiza la curva y busca el radio más pequeño. Fuente: Autores.

Con el objetivo de poder representar de manera gráfica el análisis del radio de la curva se empleó una definición adicional que pudiese generar geometría exportable a otros programas.

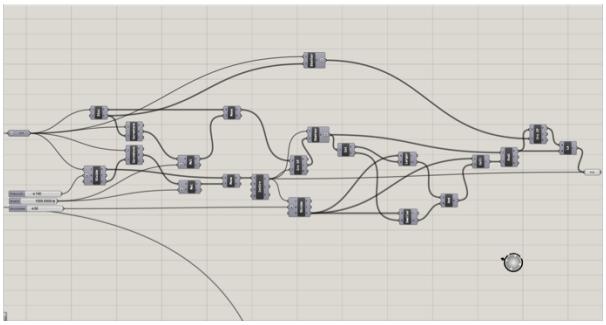

Imagen 95. Definición de "Grasshopper" adicional que genera geometría exportable a otros programas. Fuente: Autores.

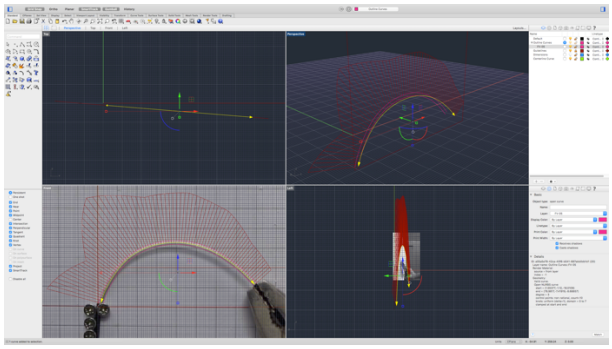

Imagen 96. Captura de pantalla del resultado de la definición de "Grasshopper" en "Rhinoceros 3D" mostrando la representación gráfica del radio a lo largo de la curva. También, en la vista superior (Top), en la esquina superior izquierda se ven las curvas rotadas respecto al plano vertical. Fuente: Autores.

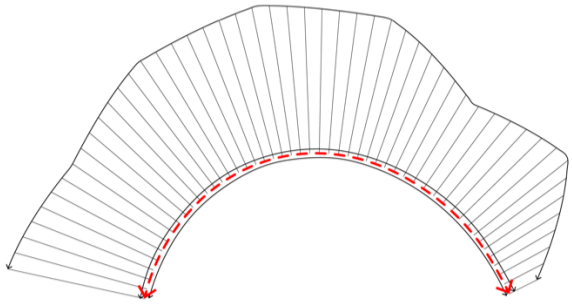

Imagen 97. Representación gráfica del radio cambiante de la curva en rojo, las líneas rectas perpendiculares a la curva cambian de acuerdo a la magnitud del radio en el punto donde se encuentran. Fuente: Autores.

El análisis se repite para cada una de las treinta muestras válidas de las más de noventa que se realizaron en un proceso de prueba y error hasta llegar a la cantidad necesaria. Posteriormente la definición agrupa los radios en una lista que se exporta a un archivo de texto para leer los datos desde otro programa.

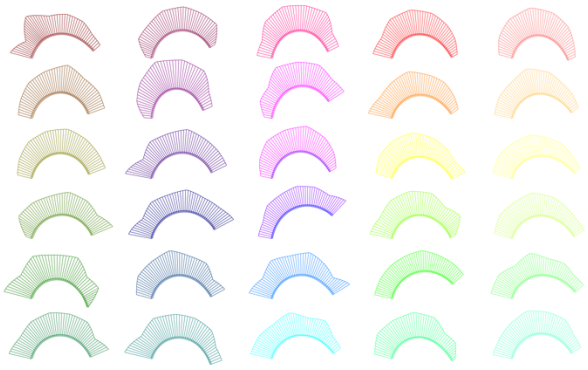

Imagen 98. Representación gráfica del radio cambiante de la curva de las 30 muestras válidas. Fuente: Autores.

Tabulación de datos y resultados

Se obtuvo un promedio de 32,3mm para el radio mínimo al cual se pueden doblar las barras antes de quebrarse. Utilizando un grado de seguridad de tres veces la desviación estándar más el promedio se obtuvo un valor de 41,6mm; este valor corresponde radio mínimo al cual se pueden doblar las barras antes de que alguna comience a fallar.

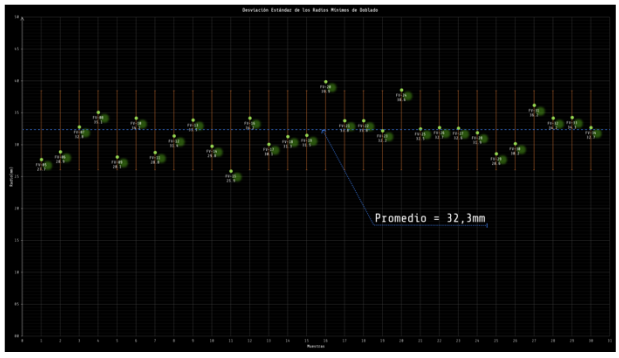

Imagen 99. Promedio del radio en el cual fallaron las muestras, el eje y muestra el radio en mm, el eje x las muestras, las barras de error de cada muestra están en anaranjado y el promedio en azul. Fuente: Autores.

8.7. Ejecución de pruebas

Estructuras modulares

El siguiente paso fue la propuesta de una serie de estructuras modulares que hicieran uso de materiales flexibles. Entre las diferentes propuestas se trabajaron dos: un módulo de base triangular y otro a partir de una figura circular que es tensada diametralmente.

El objetivo de crear estas estructuras modulares es que tengan capacidad de movimiento y de interacción. Se busca un sistema de agentes donde cada módulo tenga capacidad de movimiento regulado por una serie de reglas configurables y también capacidad de interactuar con otros módulos para unirse y formar estructuras más complejas.

Módulos Circulares:

Este módulo está compuesto por una barra “pegajosa” de forma circular y un Springline que tiene como extremos dos puntos opuestos en el círculo y actúa como tensor para formar una figura similar a la de un número 8. La cantidad de puntos utilizados para formar la figura circular afectan la curvatura de la misma, al tener más puntos para trazar el círculo, los ángulos entre cada punto son menos pronunciados resultando en una curva más suave. Otro factor determinante en la figura es la fuerza de tensión del Springline, conforme se aumenta la fuerza de tensión aplicada al círculo, esta determina que tanto se acercan los dos puntos de anclaje opuestos diametralmente.

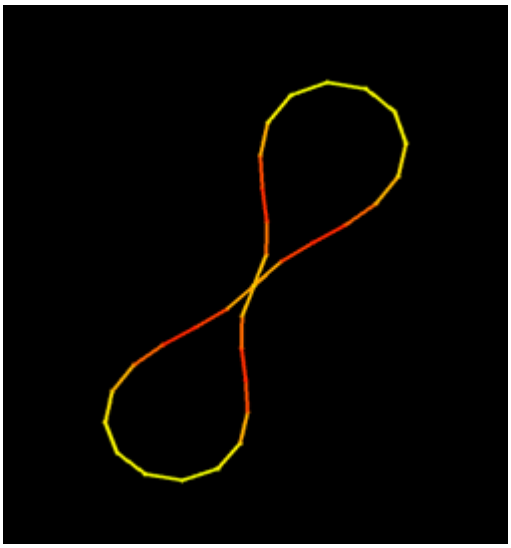

Imagen 100. Figura generada con 32 puntos. Fuente: Autores.

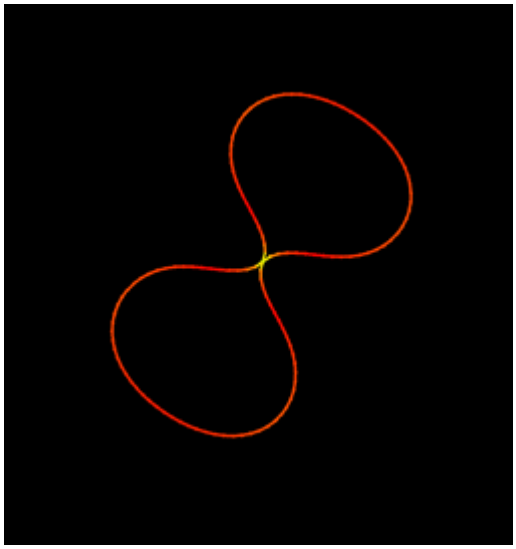

Imagen 101. Figura con 128 puntos (mismos parámetros que la anterior). Fuente: Autores.

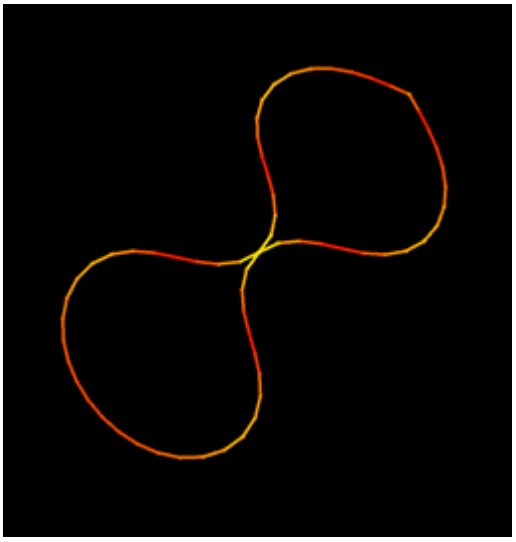

Imagen 102. Valores predeterminados con 64 puntos. Fuente: Autores.

Controles:

Posteriormente se implementaron controles para poder manejar ciertas propiedades principales de las barras, como por ejemplo el tamaño entre los segmentos, la tensión de la barra, la fricción de los puntos, entre otros, estos controles permitieron observar el comportamiento de las barras con diferentes configuraciones y dieron la posibilidad de probar estas distintas configuraciones de manera dinámica y mucho más eficiente que alterando valores en el código y volviendo a compilar y ejecutar para cada prueba.

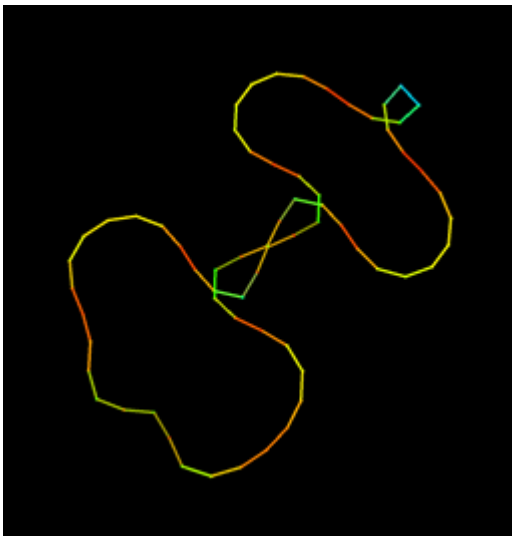

Imagen 103. Aumentando el tamaño entre los segmentos. Fuente: Autores.

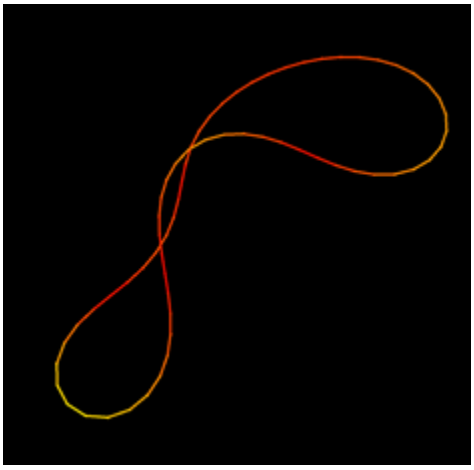

Imagen 104. Aumentando la tensión y la fricción de la barra. Fuente: Autores.

Conexión entre módulos:

Luego de la implementación de los controles se procedió a programar la conexión entre varios módulos circulares, para generar configuraciones más complejas a partir de la unión de varios módulos y la deformación que estas uniones provocan en los mismos.

Se determinó que, para generar configuraciones deseables, los puntos de conexión validos dentro del módulo circular debían ser limitados, los puntos en los que se permiten uniones van desde el grado 0 en incrementos de 45 grados (0,45,90…).

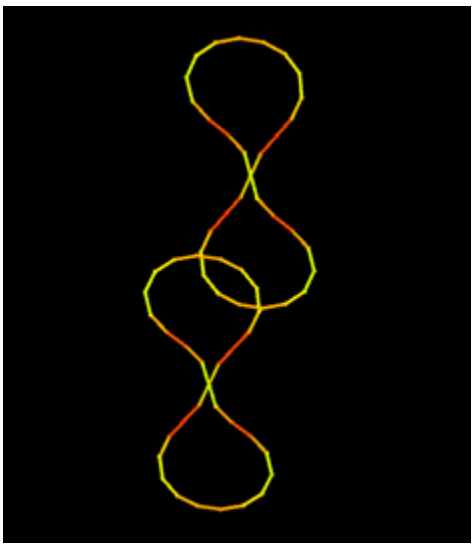

Imagen 105. Si dos módulos se intersecan en puntos no válidos, no se crean conexiones. Fuente: Autores.

Se experimentó con dos tipos de uniones entre los módulos, inicialmente, se programó de tal manera que cuando dos puntos válidos de dos módulos distintos estaban lo suficientemente cerca se creaba la unión por medio de reemplazar la partícula de un módulo con la del que se iba a unir, este tipo de unión generaba problemas una vez formada la unión, cuando uno de los módulos se movía, el otro que estaba unido se deformaba de una manera no deseable.

Esto llevó a que se probara generar las uniones mediante un Springline similar al que actúa como tensor en el centro del módulo, de tal manera que los módulos se pudieran mover individualmente pero simultáneamente afectar la figura del módulo con el que estaban conectados.

Otra ventaja de este acercamiento a las uniones fue que se podía controlar la fuerza y longitud del tensor, lo cual cambia la manera en la que se conectan los módulos y también como el movimiento de uno de ellos afecta al que está conectado.

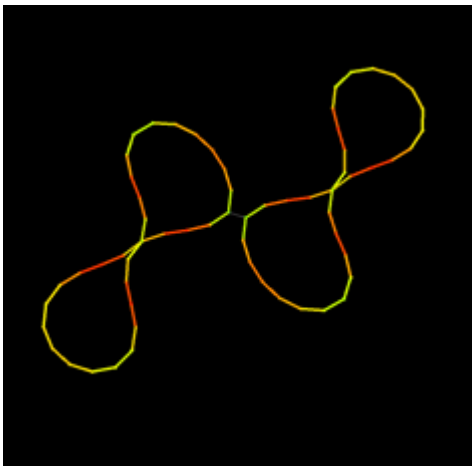

Imagen 106 Tensor de fuerza baja.

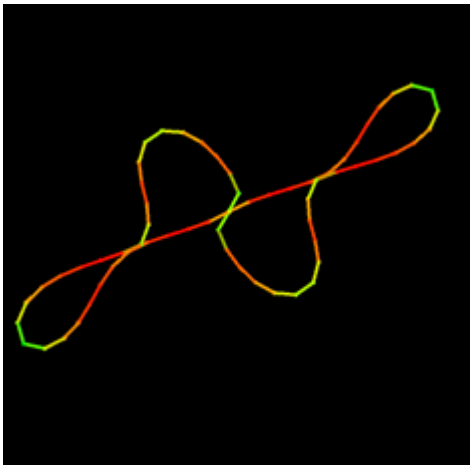

Imagen 107. El movimiento de uno de los módulos deforma al que tiene conectado. Fuente: Autores.

Inicialmente se programaron controles para poder mover los módulos de manera manual para poder hacer pruebas y conectarlos de manera intencional, con el fin de estudiar las conexiones y sus propiedades. Los controles permiten seleccionar uno o varios módulos para los cuales se pueden activar o desactivar, los módulos "activos" se podían mover en dos dimensiones utilizando el teclado.

Para el movimiento de los módulos se ajusta la posición de cada una de las partículas en un monto determinado simultáneamente, este monto también es ajustable en los controles del programa. Se programó de esta manera ya que, si se aplicaba una fuerza a todos los puntos del módulo, el movimiento del mismo provocaba deformaciones y esto no era deseable a menos que dos módulos se unieran.

Otro de los controles que se programó fue uno para crear un nuevo módulo, se dibuja un punto en la pantalla, el cual se puede mover con el teclado y al presionar el botón de crear nuevo módulo, se crea el mismo en el punto seleccionado.

Posteriormente se fijó el objetivo de que el movimiento de los módulos fuera determinado de manera automática sin requerir control del usuario, con el fin de generar configuraciones emergentes de un conjunto de reglas.

Movimiento de agentes:

Para esta etapa el objetivo era que la interacción entre los módulos ocurriera de manera espontánea, siguiendo un conjunto de reglas simples para generar un comportamiento más complejo.

El método que se planteó para lograr esto fue que cada módulo fuera un agente autónomo en un sistema que tuviera reglas de cohesión, separación y de alineamiento con el resto de los agentes del sistema, aplicando principios de Flocking, los cuales habían sido estudiados en etapas previas del proyecto.

Se programaron controles para controlar cada uno de los parámetros, tanto el rango, como la fuerza para la cohesión, alineamiento y separación entre los agentes del sistema.

La manera en la que se programaron los agentes fue que cada uno de los módulos, tenía una "partícula agente" en su punto central, y a partir de este punto se calculan las fuerzas del Flocking con respecto a la distancia del resto de los agentes del sistema.

Esta etapa de la exploración fue la que presentó más complicaciones, particularmente integrando el movimiento de agentes a los módulos. Esto porque la manera en la que se mueven los módulos no es mediante fuerzas sino alterando la posición de cada uno de sus puntos.

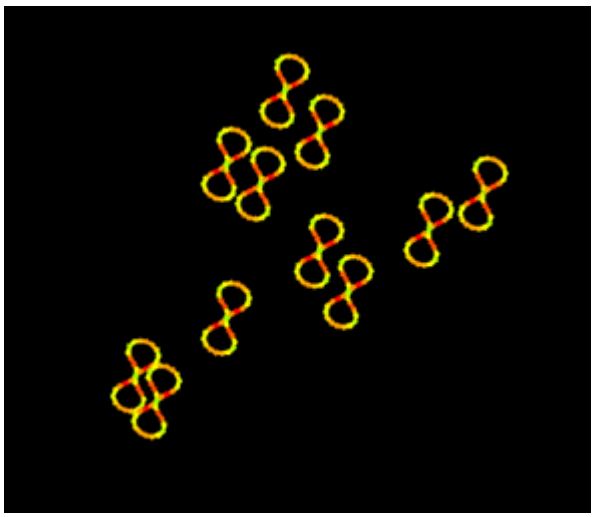

Imagen 108. Sistema con 11 agentes. Fuente: Autores.

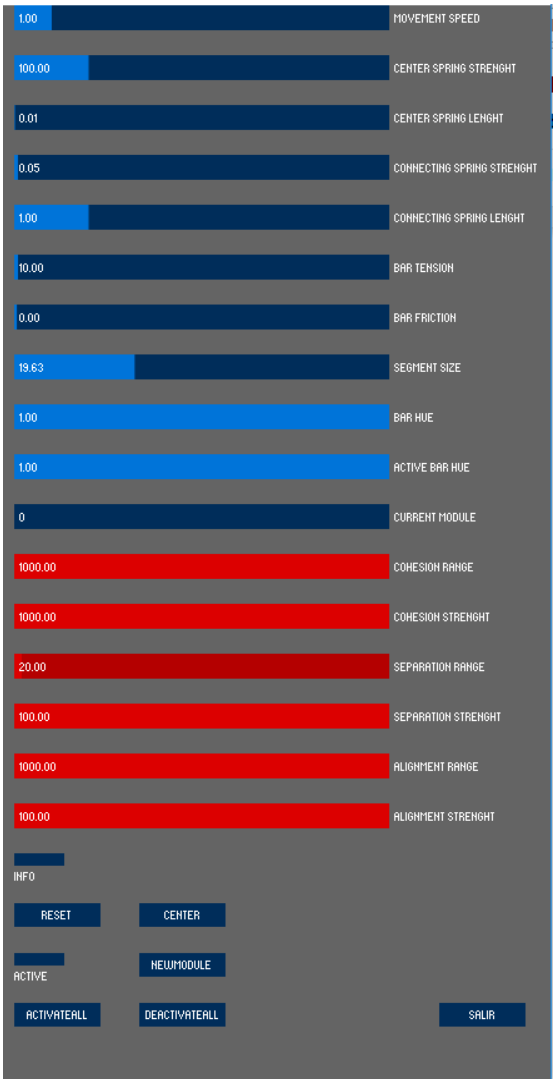

Imagen 109. Ventana de controles, las barras rojas corresponden a los parámetros de Flocking, las barras azules corresponden a los parámetros de los módulos. Fuente: Autores.

Módulo triangular

Este módulo consta de una base triangular de tamaño fijo. Esta base se crea por medio de barras de tamaño fijo *IStickLine*. Entre los vértices se ubican tres arcos construidos con barras flexibles que se orientan perpendiculares a la base. Los tres arcos se unen en su punto más alto con un triángulo formado por líneas elásticas que generan una curvatura en el arco y le dan estabilidad a la forma.

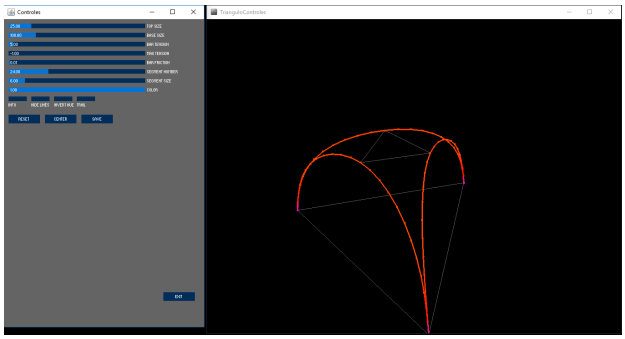

Imagen 110.Módulo contralo en tiempo real mesiante la biblioteca Control P5. Fuente: Autores.

Los diferentes atributos del módulo pueden ser modificados en vivo durante la simulación. Los que se manejaron inicialmente fueron: tamaño del triángulo superior, tamaño del triángulo base, grado de tensión de las barras, tensión máxima soportada por las barras antes de quebrarse, fricción de las partículas al moverse por el espacio, tamaño del segmento que une las partículas de las barras y color de la barra.

La construcción del módulo no tuvo mayores complicaciones, la experiencia con el sistema de barras "pegajosas" fue muy provechosa en la elaboración de la figura.

Cuando se propuso que el diseño de la piel debe responder a una necesidad local, se tenía la duda de la utilidad de la piel. Esta investigación debía por lo tanto tener una finalidad práctica y relevante, no como un objeto simplemente estético o como una investigación abierta sin utilidad.

El análisis realizado en el capítulo 2, presentó dos utilidades principales, identificándose la función de

ventilar los espacios como la de mayor potencial a este proyecto. Por esta razón se quiso relacionar el comportamiento material con las variables relacionadas al efecto Venturi de aceleración del viento.

Sin embargo, para que este comportamiento material logre el efecto deseado, se requiere generar a partir de él, configuraciones geométricas que permitan crear una estructura tridimensional tipo embudo. De todas las posibilidades analizadas, se definieron tres configuraciones que pueden generar un efecto Venturi y que resulten viables de ensamblar. De una barra doblada sobre sí misma se obtiene un módulo elipsoidal; de dos barras dobladas de concavidad opuesta y separadas por barras rígidas mediante un ángulo de 90 grados, se obtiene un módulo romboidal; de tres barras unidas entre sí por sus extremos, y tensadas cada una para generar la curvatura, se obtiene el módulo triangular. Para este momento, el sistema de programación algorítmica podía simular cada uno de estos módulos, de forma que, al ingresar los parámetros del material, el sistema computaba la forma según correspondía, es decir, si las barras eran muy largas el sistema se hacía muy flexible, y si se juntaban las bases de los triángulos, las barras se quebrarían justo como lo harían en realidad. (imagen 7)

Cada uno de esos módulos se agrupó en una secuencia repetitiva para formar una pared plana, con áreas específicas destinadas a la colocación de membranas textiles, emulando una envolvente capaz de re-direccionar el flujo de aire. Esta pared se sometió a una simulación digital de viento en dos ángulos de incidencia para poder valorar su efecto de aceleración. Este efecto de aceleración de un flujo de aire por angostamiento, se conoce como fenómeno Venturi, el cual es a su vez, una interpretación del principio de Bernoulli, desarrollado para el estudio de mecánica de fluidos. En términos técnicos, este comportamiento se describe como un aumento en la velocidad, y disminución en la presión que experimenta un fluido en movimiento dentro de un conducto cerrado, al ser canalizado a través de una sección meno

A partir de estas premisas teóricas, y ensayos experimentales, se plantearon tres diseños de paneles modulares en función del angostamiento y la aceleración de un flujo de aire incidente.

Cada una de estas propuestas fue configurada en un conjunto repetitivo que reproduce la eventual repetición reticulada que se busca a través de los algoritmos generativos. A continuación, el detalle de las tres configuraciones realizadas.

Módulo Romboidal

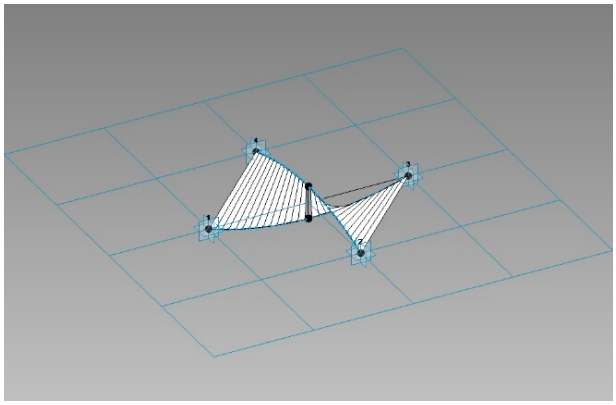

Imagen 111. Panel de geometría romboidal a partir de la interpolación de curvas opuestas. Fuente: Autores.

El primero de estos módulos es producto de la interpolación entre dos curvas de concavidad opuesta, con un desfase de orientación de 90 grados entre sí. La superficie parabólica resultante proveerá un cambio de dirección y el ensamble repetitivo se encargaría del angostamiento.

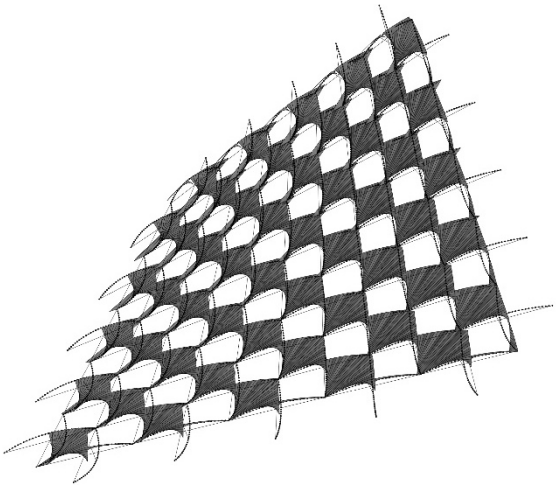

Imagen 112. Conjunto regular de paneles romboidales para la aceleración del aire incidente. Fuente: Autores.

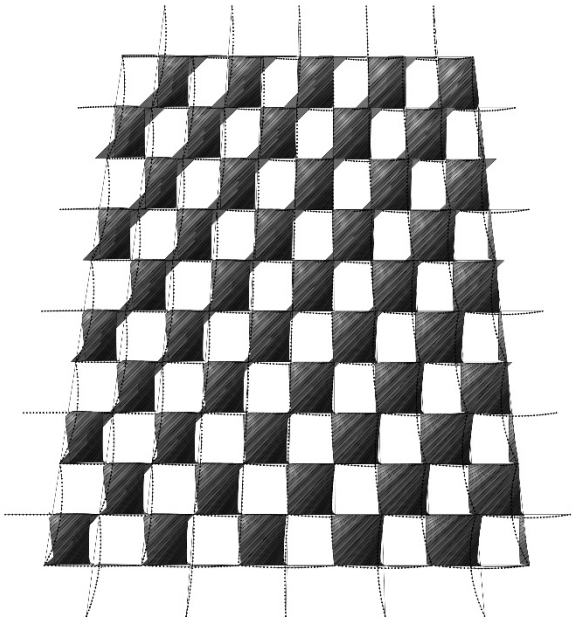

Imagen 113. Conjunto regular de paneles romboidales para la aceleración del aire incidente. Fuente: Autores.

La primera serie de pruebas realizadas se concentró en la tasa de flujo de aire al atravesar la membrana de paneles. Durante estas simulaciones se logró apreciar un excesivo comportamiento de redireccionamiento, el cual limita la aceleración, y con ello, el alcance de la ventilación en el interior del espacio habitable.

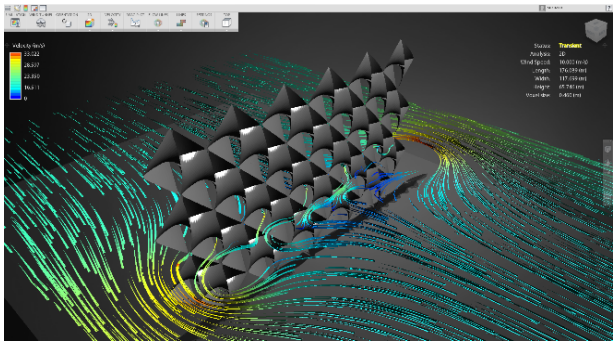

Imagen 114. Simulación CFD sobre la panelización romboidal, con visualización de líneas de flujo. Fuente: Autores.

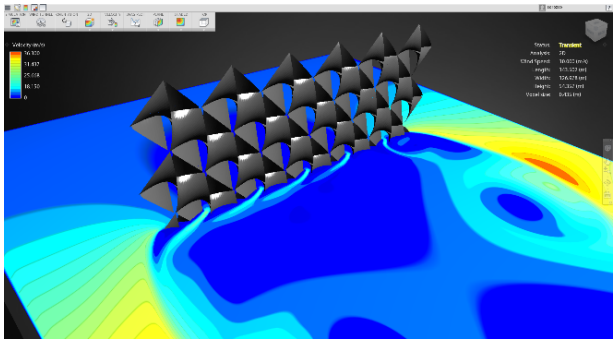

Imagen 115. Simulación CFD sobre la panelización romboidal, con visualización de tasa de flujo. Fuente: Autores.

La hipótesis para justificar este fenómeno, es la creación de pequeños vórtices de turbulencia en el espacio tridimensional comprendido entre las parábolas de cada módulo.

Posteriormente, se analizó la presión ejercida sobre la membrana para poder apreciar su potencial deformación. En este caso, se aprecian resultados ampliamente asimétricos derivados de la poca capacidad de aceleración del flujo.

Los valores de velocidad del aire resultantes al costado de sotavento se encuentran entre los 0-8 m/s.

Esta asimetría en la concentración de presión indica un alto potencial de deformación del conjunto gracias a un excesivo efecto de vela. Esta situación podría comprometer la integridad estructural de la membrana y por lo tanto debería evitarse.

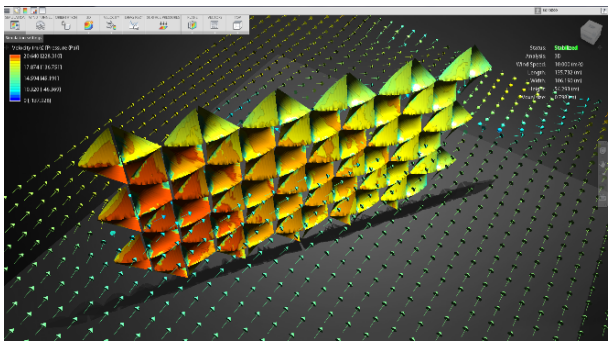

Imagen 116. Fuente: Autores.

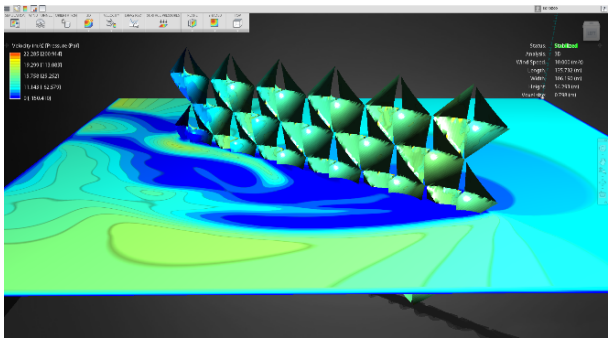

Imagen 117. Simulación CFD sobre la panelización romboidal, con visualización de presión y tasa de flujo. Fuente: Autores.

Finalmente, al combinar los factores de presión y velocidad, es posible apreciar un rendimiento poco satisfactorio, especialmente notorio sobre el costado interno de la membrana, donde no se da un efecto de succión adecuado, y por lo tanto no se logran los objetivos de ventilación buscados.

Módulo Triangular

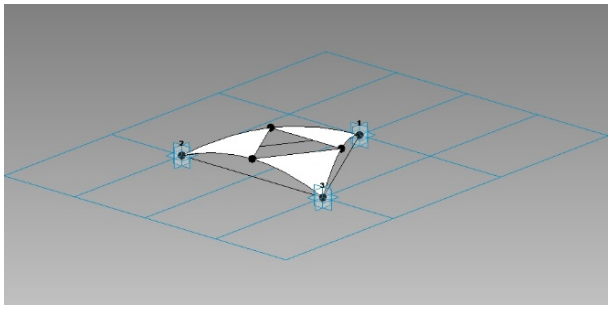

Imagen 118. Panel de geometría triangular a partir de la interpolación de curvas cóncavas. Fuente: Autores.

El segundo diseño en ser analizado mediante las herramientas de simulación fue una composición geométrica más simple, de perímetro triangular con tres curvas que al interpolarse generan una superficie entre sí. Finalmente, se realiza una perforación triangular entre los tres puntos de altura máxima de cada curva.

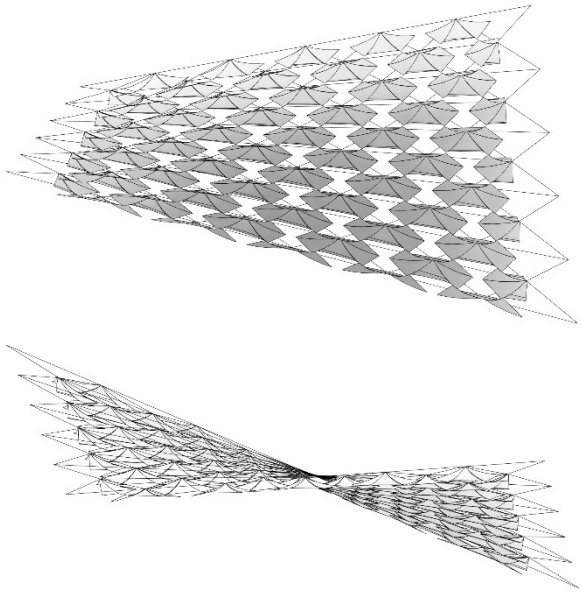

Imagen 119. Conjunto irregular de paneles triangulares para la aceleración del aire incidente. Fuente: Autores.

Las primeras pruebas de tasa de flujo (velocidad), revelaron un comportamiento interesante a efectos de la aceleración del movimiento a medida que se atraviesa la membrana.

La geometría específica de cada panel crea una canalización tipo embudo, que permite un flujo consistente de aire, pero con un aumento significativo de presión a barlovento, que genera una mayor velocidad de ingreso en el lado de sotavento (eventual interior del espacio habitable).

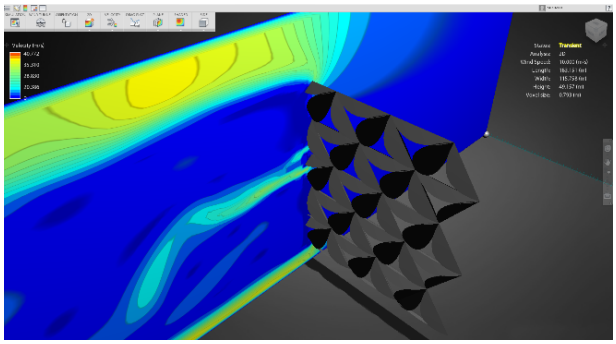

Imagen 120. Simulación CFD sobre la panelización triangular, con visualización de tasa de flujo. Fuente: Autores.

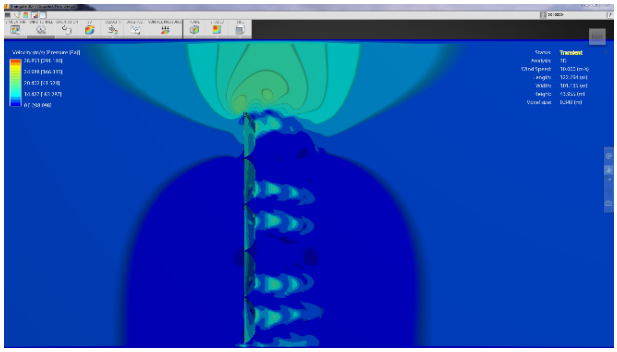

Imagen 121. Simulación CFD sobre la panelización triangular, con visualización de tasa de flujo. Fuente: Autores.

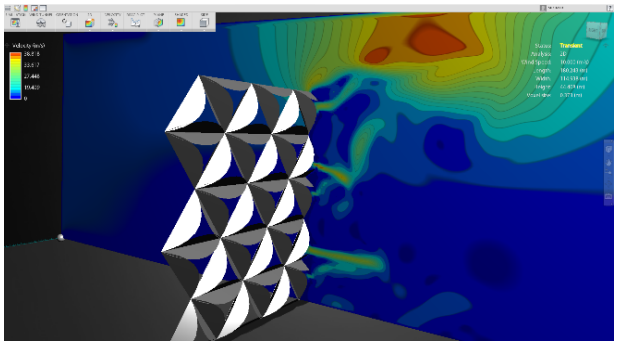

Imagen 122. Simulación CFD sobre la panelización triangular, con visualización de tasa de flujo. Fuente: Autores.

Estas corridas de simulación arrojaron valores de velocidad de entre 19-27 m/s en el costado a sotavento lo cual podría disminuir entre un 20-25% en condiciones reales, dada la ausencia de barreras artificiales del túnel de viento.

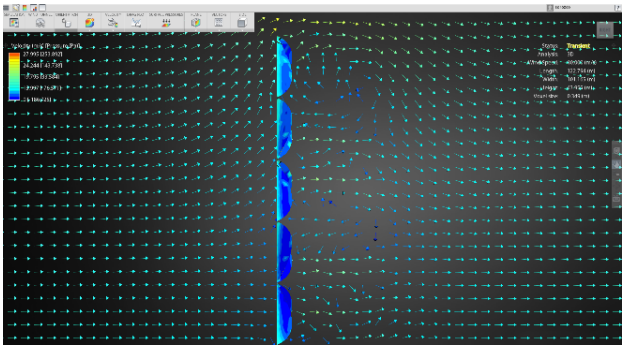

Imagen 123. Simulación CFD sobre la panelización romboidal, con visualización de presión y vectores de flujo. Fuente: Autores.

La prueba de vectores de flujo es útil para determinar el impacto de eventuales turbulencias en el costado a sotavento, el cual es significativamente menor en este caso que en el módulo romboidal. Esto refuerza la conclusión preliminar de un mejor rendimiento.

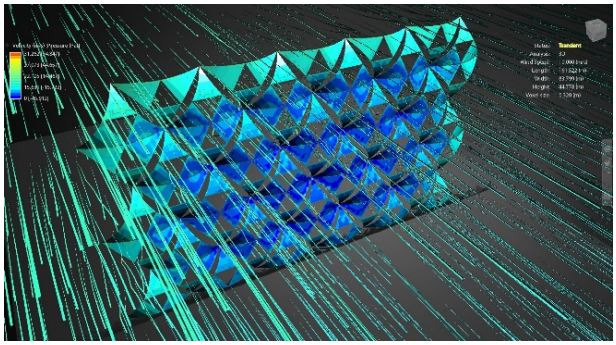

Imagen 124. Simulación CFD sobre la panelización romboidal, con visualización de presión y líneas de flujo.. Fuente: Autores.

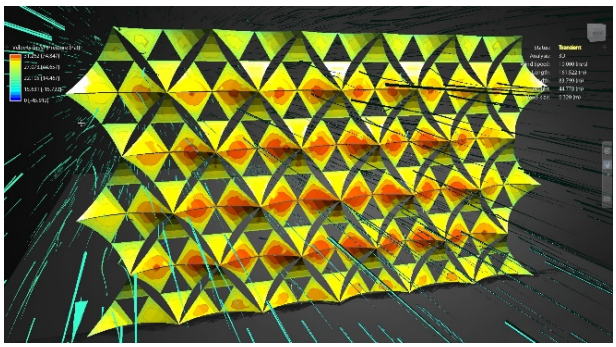

Imagen 125. Simulación CFD sobre la panelización romboidal, con visualización de presión y líneas de flujo. Fuente: Autores.

Finalmente, la simulación conjunta de velocidad y presión muestra una distribución más uniforme de esta última variable tanto a barlovento como a sotavento, lo cual permite presumir mejores posibilidades de integridad estructural de la membrana ante ráfagas de viento más fuertes.

Simulaciones posteriores podrían enfocarse en el dimensionamiento de la perforación central de cada módulo, así como el diseño específico de la superficie tensada.

Módulo Elipsoidal

La tercera variante de diseño está compuesta por una elipse vertical que se angosta en su parte central hasta crear un contacto tangencial entre los dos puntos centrales de cada curva.

Una vez creada esta figura de soporte, se genera una membrana que cubre cada módulo semi-circular con una leve curvatura convexa hacia el lado de barlovento,

con el objetivo de canalizar el aire a través de pequeñas aperturas en sus extremos.

Para este módulo se obtienen resultados mixtos entre una buena capacidad de aceleración, pero una direccionalidad difusa del flujo resultante a sotavento, es decir, en el costado interior de la membrana.

Los valores de tasa de flujo se encuentran entre los 2-16 m/s, sin embargo, dependiendo del ángulo de incidencia, el flujo puede llegar a re-direccionarse significativamente o incluso a suprimirse por completo.

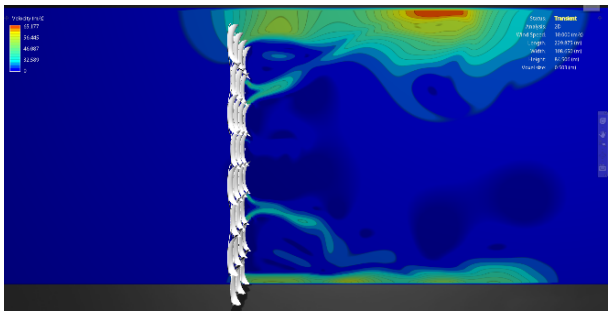

Imagen 126. Simulación CFD sobre la panelización elipsoidal, con visualización de tasa de flujo. Fuente: Autores.

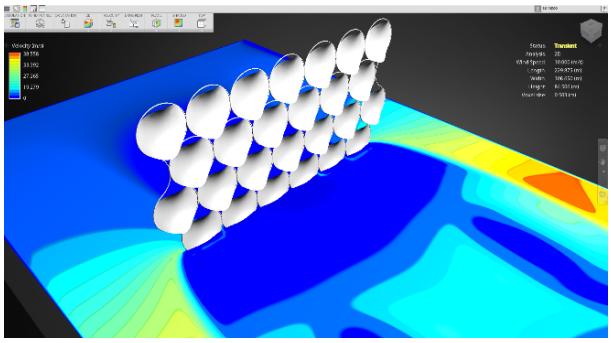

Imagen 127. Simulación CFD sobre la panelización elipsoidal, con visualización de tasa de flujo. Fuente: Autores.

A pesar de lo anterior, la distribución de presión sobre la superficie de la membrana también se comporta de manera simétrica y equitativa, compartiendo los mismos beneficios potenciales del módulo triangular.

En síntesis, el principal problema de esta panelización radica en la dificultad predictiva de su rendimiento geométrico.

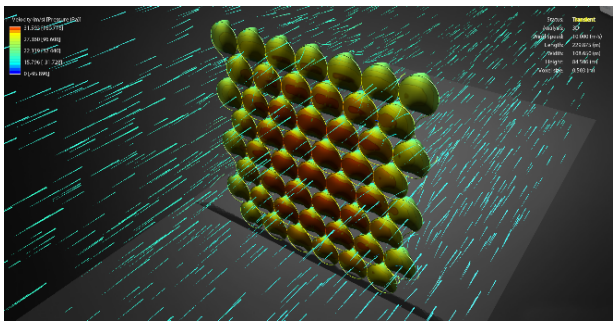

Imagen 128. Simulación CFD sobre la panelización elipsoidal, con visualización de presión y líneas de flujo. Fuente: Autores.

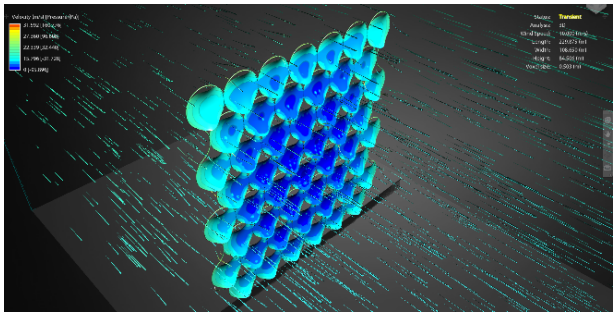

Imagen 129. Simulación CFD sobre la panelización elipsoidal, con visualización de presión y líneas de flujo.. Fuente: Autores.

8.8. Fabricación Digital

Los datos del material obtenidos son utilizados en el Sistema Material Programado para alimentar una simulación geométrica desarrollada en Processing que responde a la realidad del material utilizado. Es decir, en el proceso de simulación, las curvas generadas se mantienen dentro de los parámetros reales que el material permite. Esto garantiza que los resultados generados sean viables desde el punto de vista de la fabricación e implementación del material, lo que representa una de las grandes diferencias con respecto a una búsqueda de la forma digital en el diseño convencional.

Uno de los sistemas materiales programados que se seleccionó para fabricación en el proceso de investigación, consiste en una base triangular con arcos que van de un nodo a otro formando un módulo triangular.

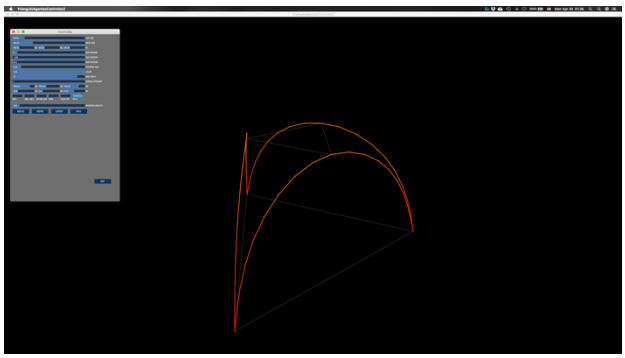

Imagen 130 Simulación del módulo básico del sistema material. Las barras de fibra de vidrio están representadas en color anaranjado. Fuente: Autores.

Los módulos se mueven interactuando entre sí mismos para formar un arreglo que puede conformar una de las posibles soluciones formales. Existen diferentes posibilidades formales e iteraciones formales del sistema.

A partir de este punto se tomó una de tantas para desarrollar los enlaces a fabricación y su posterior fabricación. Es evidente que no se pueden fabricar todas las posibles iteraciones geométricas del sistema y esa tampoco es la finalidad, el objetivo es desarrollar los enlaces con una de las iteraciones geométricas como excusa para verificar el funcionamiento de los mismos y continuar con la fabricación de prototipos.

Transferencia de datos

El sistema material genera un archivo de texto de los puntos de los módulos. El primer paso de esta parte es desarrollar un componente que pueda leer los datos de Processing en Grasshopper.

A pesar de que ya existe un componente destinado para tal fin, éste no es funcional en la versión de Grasshopper 0.9.0080 para Mac OS. A la luz de lo anterior se decidió desarrollar un script en Python que cumpliese con la finalidad deseada:

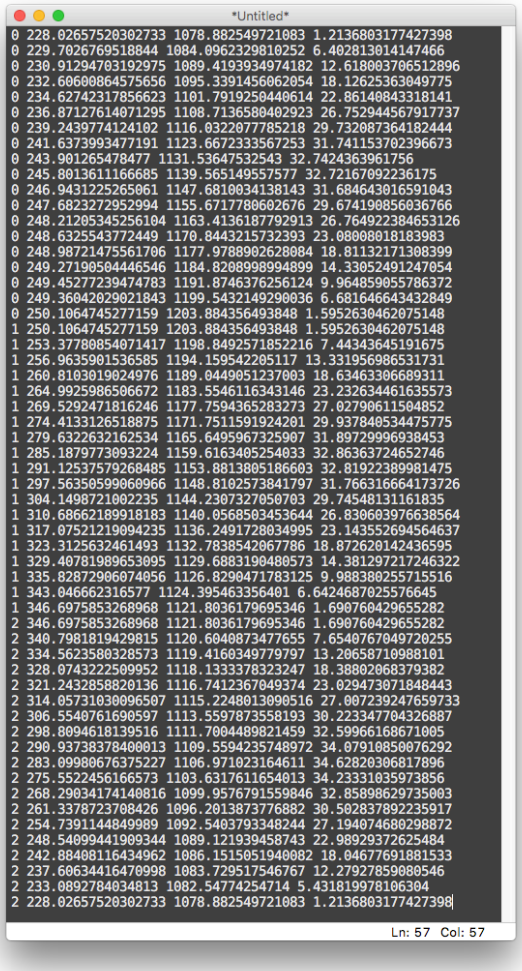

Imagen 131 Lista de puntos generada por Processing. Fuente: Autores.

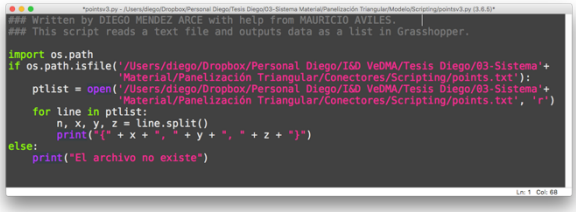

Imagen 132 Código del componente de Python para leer los datos de Processing en Grasshopper. Fuente: Autores.

El traslado de la información digital a la fabricación implica resolver detalles físicos que no eran relevantes de incluir en la simulación. Ejemplo de esto son las uniones entre las barras. En el ambiente digital, esto basta con una coincidencia en la ubicación de los extremos de dos barras diferentes para que ya se entiendan unidas. Sin embargo, al trabajar con materiales reales, la manera de generar conexiones

entre ellos es un tema fundamental. Dado que la escala del material seleccionado permite trabajar con dimensiones pequeñas, se supuso que todas las uniones necesarias se podrían fabricar a la medida por medio de la impresión 3d.

Para el final del proceso de sistematización de variables y entendimiento material, el sistema de programación algorítmico desarrollado, podía simular cada uno de los módulos expuestos en el capítulo anterior, de forma que, al ingresar los parámetros del material, el sistema computaba la forma según correspondía, es decir, si las barras eran muy largas el sistema se hacía muy flexible, y si se juntaban las bases de los triángulos, las barras se quebrarían justo como lo harían en realidad. (imagen 130 y 131). El paso siguiente consistía en utilizar estos resultados como insumo para la fabricación del prototipo final, con el material real.

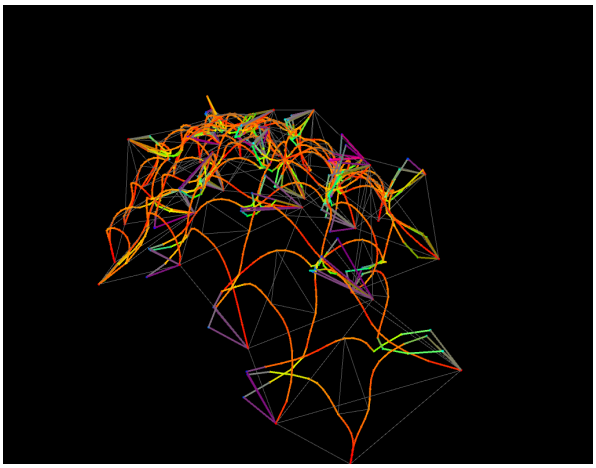

Imagen 133. Módulos regulares triangulados con la generación de formas irregulares producto de la aplicación de parámetros diferentes a los del material real. Fuente: Autores.

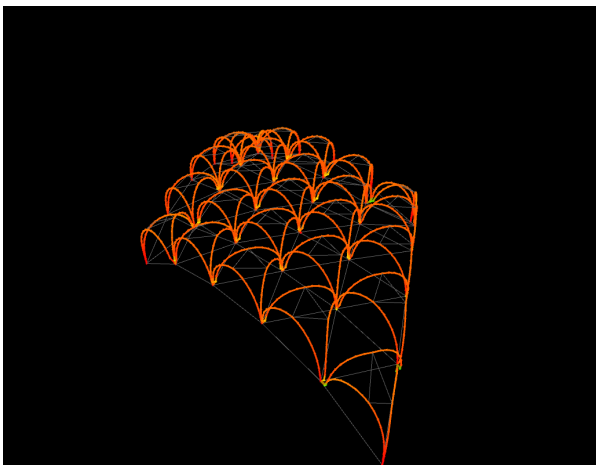

Imagen 134. Módulos regulares triangulados con la generación de arcos según los parámetros del material. Fuente: Autores.

8.9. Fabricación del prototipo

Eventualmente, el diseño de una envolvente de este tipo puede estar compuesta por cientos de módulos, cada uno de ellos de tamaño y geometría diferente. Si tomamos en cuenta que cada módulo se compone de tres barras flexibles, 3 cables, tres nodos o conectores y una membrana, la cantidad de componentes necesarios para la producción y ensamblaje de una estructura real sería abrumadora. Por esta razón el proceso debe continuar mediante la fabricación digital. Las uniones de las barras de fibra de vidrio están parametrizadas de tal forma que, si la geometría del módulo cambia, la unión se ajusta a los cambios en tiempo real. Cada conector es capaz de recibir seis módulos y cada módulo le aporta dos barras de fibra de vidrio que entran de forma inclinada y dos barras rígidas de acero o cables que entran paralelas a la base, para un total de 24 perforaciones. Estos conectores fueron fabricados en impresión 3d luego de una variedad de pruebas para asegurar que soportaran la tensión de las barras dobladas.

Para el corte de las membranas, se probaron una serie de materiales de tipo textil que pudieran ser manipulados mediante corte láser, pero cosidos y ensamblados manualmente. Una tela tipo licra de bajo costo fue suficiente para alcanzar el objetivo de crear una barrera al viento y direccionarlo. Para su fabricación, se realizó el mismo proceso paramétrico aplicado a las uniones antes descritas, de manera que

sin importar la forma de la membrana que se extrae de los datos de la simulación, se produce la información requerida para el corte digital.

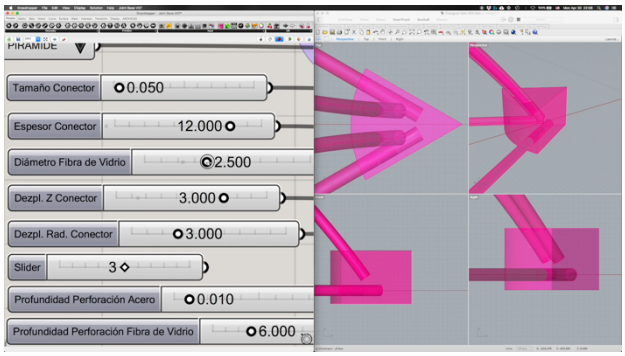

Imagen 135. Proceso de parametrización de los conectores en Grasshopper. Fuente: Autores.

Las uniones de las barras de fibra de vidrio están parametrizadas de tal forma que, si la geometría del módulo cambia, la unión se ajusta a los cambios en tiempo real. Cada conector es capaz de recibir seis módulos y cada módulo le aporta dos barras de fibra de vidrio que entran de forma inclinada y dos barras rígidas de acero que entran paralelas a la base, para un total de 24 perforaciones.

Debido a que los conectores enlazan varios elementos triangulares su definición de Grasshopper comienza por un segmento circular de sesenta grados del conector que rota instancias independientes de forma repetitiva para generar componentes únicos de forma automatizada. Para entender esto podemos emplear una metáfora, el conector es como una pizza que está hecha de seis rebanadas, cada rebanada es responsable de conectar un módulo diferente.

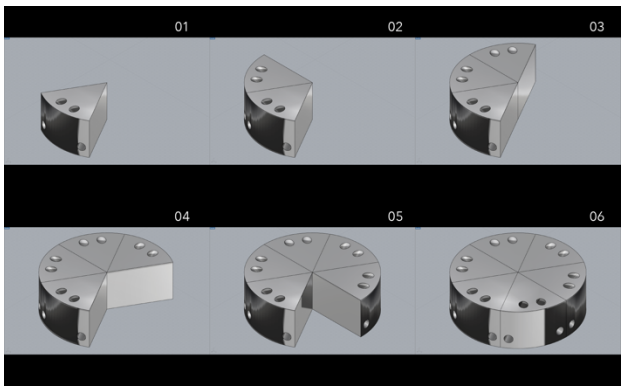

Imagen 136 Visualización de la unión parametrizada. Fuente: Autores.

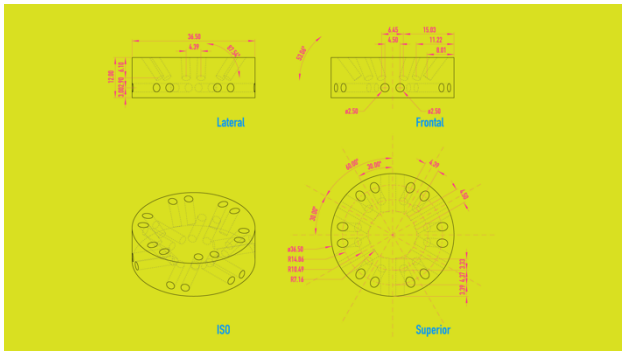

Imagen 137 Dibujo técnico derivado del modelo paramétrico de una de las configuraciones geométricas del sistema programado. Fuente: Autores.

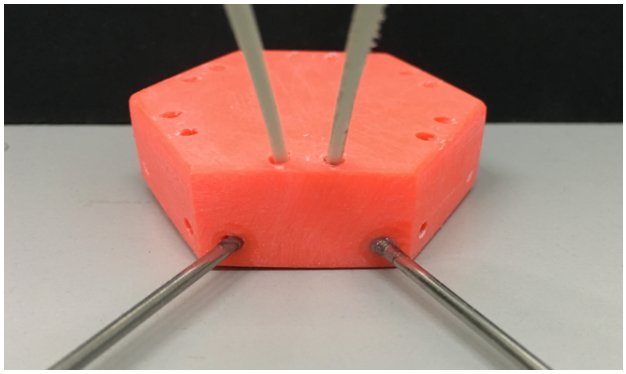

Imagen 138. Prototipo de conector impreso en filament plástico mediante impression 3D. Fuente: Autores.

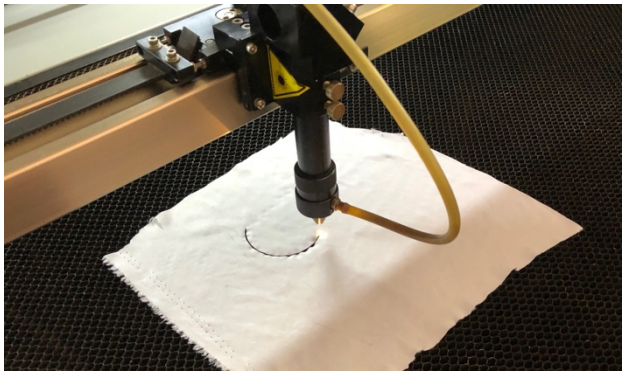

Imagen 139. Prueba de corte laser a la tela de licra a utilizar en el prototipo. Fuente: Autores.

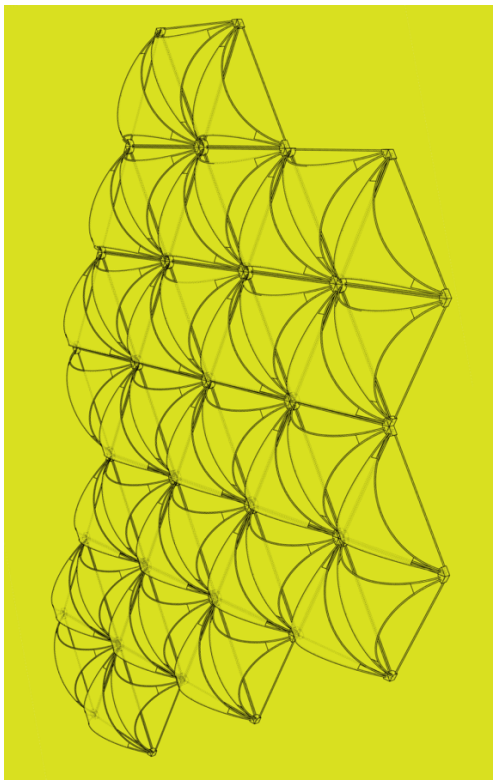

Imagen 140. Dibujo del módulo a fabricar como prototipo final. Fuente: Autores

Imagen 141. Detalle del prototipo final del Proyecto. Fuente: Autores.

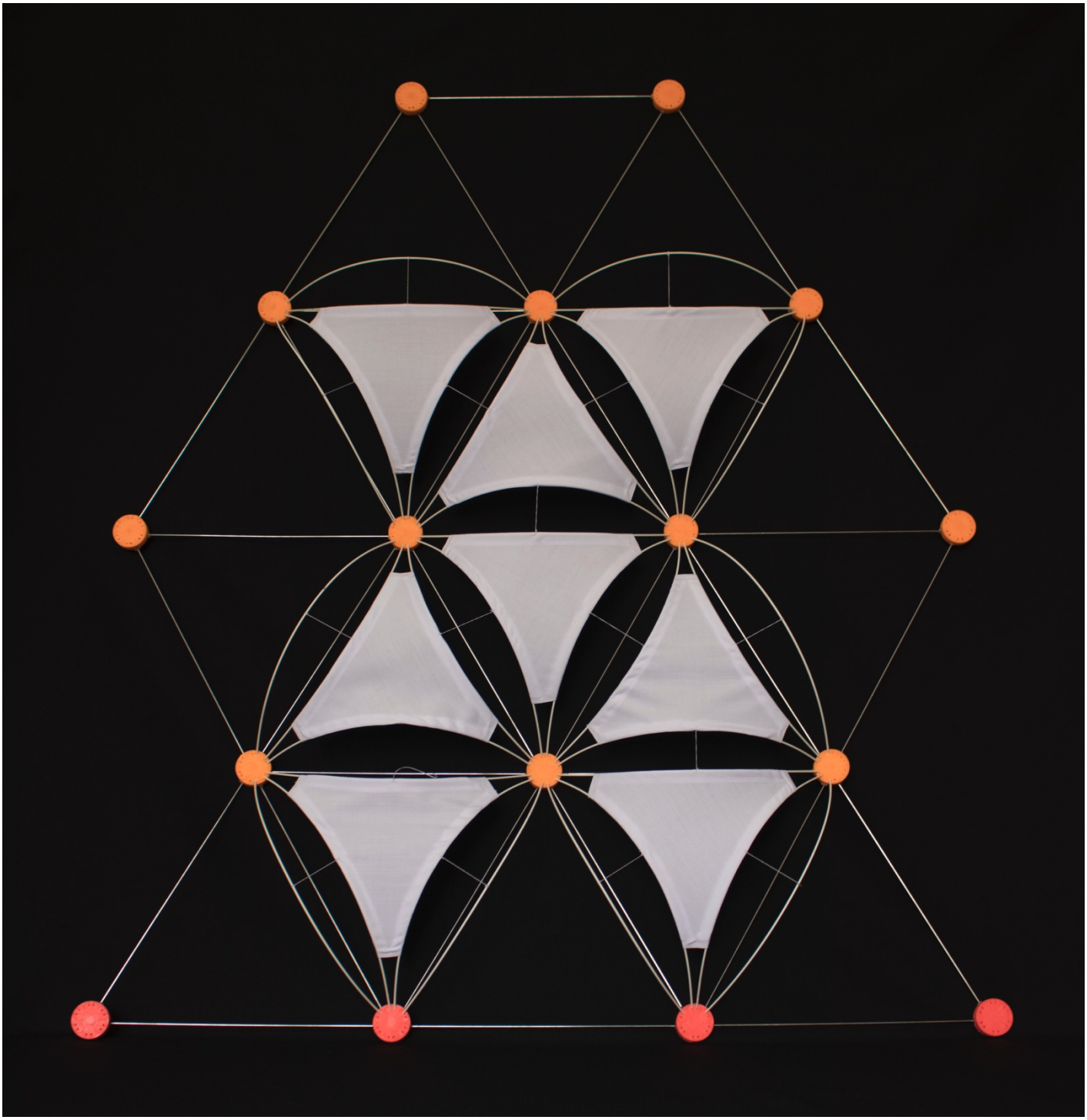

Imagen 142. Prototipo final, que consistió en 18 módulos, 8 de los cuales presentan la piel en licra. Fuente: Autores

9. Discusión y conclusiones

Se ha visto en investigaciones relacionadas, que el campo de acción mayoritario está orientado hacia la innovación tecnológica y la especulación sobre futuros usos en el diseño. El trabajo de Gilles Retsin y Manuel Jiménez (2016) ejemplifica esta dirección, ya que al usar métodos computacionales discretos, logran diseños que incluyen las restricciones físicas de la manufactura robótica, disminuyendo así los posibles errores de fabricación, al tiempo que mejoran el uso de los recursos computacionales disponibles en el manejo de grandes cantidades de datos. Pero como se ha argumentado, el diseño algorítmico tiene un potencial de uso más amplio si se integran las restricciones básicas que van más allá de la vanguardia tecnológica.

En contextos ***low tech***, la aplicación del diseño computacional avanzado se propicia debido a la universalización de las tecnologías digitales, muchas de uso libre como ***Processing***, que facilitan la creación de ambientes geométricos tridimensionales, así como a la universalización de tecnologías de manufactura digital básicas, incluyendo las impresoras en 3D y las cortadoras láser. Por ello es que resulta fundamental identificar las cualidades que debe tener una exploración en esta línea de investigación para ser aplicada en una arquitectura apropiada a su condición contextual. Esta investigación determina hacer énfasis en dos aspectos clave: primero, que la elección de los materiales sea adecuada en relación a las tecnologías de fabricación disponibles y en relación al desempeño geométrico deseado, y segundo, considerar los costos asociados tanto a los materiales, como a los procesos de manufactura.

El proceso realizado mediante la definición de la técnica de programación digital adecuada, permite comprobar que la simulación del material corresponde a los resultados físicos del mismo, gracias a una correlación desde el punto de vista geométrico. Mientras el éxito del diseño se pueda medir en resultados prácticos, esta metodología podrá ser empleada sin la necesidad de grandes inversiones tecnológicas. De ahí se desprende que no todas las características de un material son necesarias de sistematizar en la programación. Resulta vital elegir aquellas que correspondan a las posibilidades geométricas del mismo, ya que los resultados digitales coinciden con las posibilidades de uso reales de los materiales utilizados, generando una amplitud ilimitada de posibilidades medibles de diseño.

Para el caso específico de la utilidad planteada en esta investigación, la simulación conjunta de velocidad y presión muestra una distribución más uniforme de eventuales turbulencias, tanto a barlovento como a sotavento, lo cual permite presumir mejores posibilidades de integridad estructural de la membrana triangular ante ráfagas de viento más fuertes. Simulaciones posteriores podrían enfocarse en el dimensionamiento de la perforación central de cada módulo, así como el diseño específico de la superficie tensada. Esto se suma a que la retroalimentación por el continuo testeo del diseño o inclusión de la optimización en el proceso de diseño, son temas pendientes de incluir en el sistema programado para aprovechar aún más las capacidades de computar datos de los equipos actuales, generando con ello un sistema más robusto en cuanto a la generación de diseños prácticos, orientados a la aplicabilidad de problemas cotidianos donde no medie el requerimiento de la alta tecnología.

10. Recomendaciones

El proyecto abordado constituyó en un excelente primer acercamiento a la investigación arquitectónica integrando la programación digital en el proceso de diseño. Esta línea de investigación tiene varios posibles enfoques o intereses que son viables de investigar en nuestro contexto y con el cuerpo humano disponible en el Instituto Tecnológico. Este proyecto propuso un interés que se diferencia en relación a las investigaciones realizadas en el campo en otros centros de investigación mundial, que es la indagación del diseño computacional en contextos de "***low tech***". Fue precisamente esta idea la que se comunicó en el artículo académico presentado al final del proyecto, que se tituló: "Diseño algorítmico de envolventes arquitectónicas para la aceleración del viento en contextos de ***low-tech***"

Además se dejan insinuados claramente dos futuros problemas a incluir en esta misma línea de investigación: primero, la de incluir en el sistema programado los resultados de la simulación del desempeño del diseño, el cual en esta investigación se realizó en otro ambiente 3d, que arrojaba datos para ser analizados por los investigadores pero no retroalimentaban automáticamente el sistema programado. Esto implica generar un proceso digital tipo reiterativo de constante mejora del diseño.

Segundo, la posibilidad de generar un prototipo a escala real para un espacio real, a partir de los aprendizajes encontrados en la elaboración de los modelos preliminares, e incluyendo los resultados de módulo variable que se lograron con la programaci´øn digital, pero que no se pudieron incluir en el prototipo final de este proyecto.

11. Agradecimientos

Queremos manifestar el agradecimiento al ICAT, de Virginia Tech y a la Fundación Costa Rica para la Innovación por la retroalimentación y recomendaciones durante el proceso. Así mismo, a la VIE y a la Escuela de Arquitectura por el apoyo generado durante la formulación y ejecución de este primer proyecto de investigación.

12. Referencias

Ahlquist, S., Kampowski, T., Torghabehi, O., & Speck, T. (2015). Development of a digital framework for the computation of complex material and morphological behavior of biological and technological systems. Sciencedirect.com. Retrieved 23 March 2015, from http://www.sciencedirect.com/science/article/pii/S0010448514000141

Bentley, P., & Corne, D. (2002). Creative evolutionary systems. San Francisco, CA.: Morgan Kaufmann.

Building envelope. (2007). En.wikipedia.org. Retrieved 6 July 2017, from https://en.wikipedia.org/wiki/Building_envelope#cite_ref-Straube_3-1

Burry, M. (2011). Scripting cultures. Chichester, UK: Wiley.

Dimčić, M. (2011). Structural Optimization of Grid Shells based on Genetic Algorithms (Doctorado). Institut für Tragkonstruktionen und Konstruktives Entwerfen Universität Stuttgart.

Frazer, J. (1995). An evolutionary architecture. London: Architectural Association.

Hensel, M. (2013). Performance-oriented architecture. Hoboken, N.J.: Wiley.

Hensel, M., Menges, A., & Weinstock, M. (2004). Emergence. Chichester: Wiley-Academy.

Hensel, M., Menges, A., & Weinstock, M. (2010). Emergent technologies and design. Oxon [England]: Routledge.

Menges, A., & Ahlquist, S. (2011). Computational design thinking. Chichester, UK: John Wiley & Sons.

OFS Furukawa Company. (2018). OFS 2 Count Mini LT Flat Drop Cable. Retrieved from https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=2ahUKEwiK2ZSJz9faAhVOmeAKHZJnDlkQjRx6BAgAEAU&url=https%3A%2F%2Ftsrcomsupply.com%2F2017%2F03%2F16%2Fofs-flat-drop-2-fiber-optic-cable%2F&psig=AOvVaw2-RqW1mO2XCqGDgFNzKF42&ust=1524820964841147

Parisi, L. (2013). Contagious architecture. Cambridge, MA: MIT Press.

Peters, B., & Kestelier, X. D. (2013). Computation works: The building of algorithmic thought. Chichester: John Wiley & Sons.

Peters, B., & Peters, T. (2013). Inside Smartgeometry. Chichester, West Sussex, [England]: John Wiley & Sons.

Reichert, S., Menges, A., & Correa, D. (2015). Meteorosensitive architecture: Biomimetic building skins based on materially embedded and hygroscopically enabled responsiveness. Sciencedirect.com. Retrieved 23 March 2015, from http://www.sciencedirect.com/science/article/pii/S0010448514000438

Rutten, D. (2010). Grasshopper 3D (Version 0.9.0076). Seattle, Washington: Robert McNeel & Associates.

Saad El Ahmar, S. (2011). Biomimicry As A Tool For Sustainable Architectural Design Towards Morphogenetic Architecture (Maestría). Faculty of Engineering, Alexandria University.

Shiffman, D. (2012). The nature of code. United States: The Magic Book Projekt.

Tedeschi, A., Andreani, S., & Wirz, F. (2016). AAD_Algorithms-Aided Design: Parametric strategies using Grasshopper. Brienza: Le Penseur Publisher.

Terzidis, K. (2006). Algorithmic architecture. Oxford: Architectural Press.

Weinstock, M. (2010). The architecture of emergence (1st ed.). Chichester: Wiley.

13. Apéndices

A continuación, se incluye el código algorítmico completo que se escribió para la realización de los últimos diseños modulares variables, utilizando el programa ***Processing***. Este código permite generar estructuras trianguladas de barras flexibles que, mediante controles interactivos en tiempo real, permiten la modificación de las variables incluidas en el sistema:

```
import processing.opengl.*;
import igeo.*;
import controlP5.*;
import java.awt.Frame;

boolean windowPosFlag = true;
boolean saveImages = false;
float gravityMag = 0;
ControlFrame cf;
TriangleAgentSystem system;
IGravity gravity;
int maxConnections = 4;


void setup() {
  size(910, 690, IG.GL);
  controllersInit();
  igeoInit();
}

void draw() {
  if (windowPosFlag) {
    frame.setLocation(445, 0);
    windowPosFlag = false;
  }
  if (saveImages) {
    saveFrame("img/#####.png");
  }
}

void igeoInit() {
  gravity = new IGravity(0, 0, gravityMag);
  IG.bg(0);
  system = new TriangleAgentSystem(10);
  IG.perspective();
}

void controllersInit() {
  Frame frame = new Frame("Controles");
  ControlFrame controlFrame = new ControlFrame(this,
455, 715);
  frame.add(controlFrame);
  frame.setSize(controlFrame.w, controlFrame.h);
  frame.setLocation(0, 0);
  frame.setVisible(true);
  controlFrame.init();
}
import igeo.*;
import igeo.gui.INavigator;

// Clase para controlar la cámara de IGeo
// Todos los métodos son estáticos
// Los parámetros de rotación se dan en radiantes
// Puede tener problemas al rotar exactamente PI
class Camera {
  static float horRotationInc = 0.0f;
  static float verRotationInc = 0.0f;
  static INavigator navigator =
IG.axonometricPane().navigator();

  // Rotación hacia la derecha
  static void rotateRight(float angle) {
    navigator.viewAngle = navigator.view.getAngles();
    navigator.updateRotation(angle, 0.0);
  }
  // Rotación hacia la izquierda
  static void rotateLeft(float angle) {
    rotateRight(-angle);
  }
  // Rotación automática hacia la derecha cada vez que
se invoca update
  static void setRightRotation(float inc) {
    horRotationInc = inc;
  }
  // Rotación automática hacia la izquierda cada vez
que se invoca update
  static void setLeftRotation(float inc) {
    horRotationInc = -inc;
  }
  // Rotación hacia arriba
  static void rotateUp(float angle) {
    navigator.viewAngle = navigator.view.getAngles();
    navigator.updateRotation(0.0, angle);
  }
  // Rotación automática hacia arriba cada vez que se
invoca update
  static void setUpRotation(float inc) {
    verRotationInc = inc;
  }
  // Rotación hacia abajo
  static void rotateDown(float angle) {
    rotateUp(-angle);
  }
  // Rotación automática hacia abajo acada vez que se
invoca
  static void setDownRotation(float inc) {
    verRotationInc = -inc;
  }
  // Movimiento hacia la derecha
  static void panRight(float inc) {
    navigator.viewPos = navigator.view.location();
    navigator.viewTarget = navigator.view.target();
    navigator.updatePan(inc, 0);
```

```
  }
  // Movimiento hacia la izquierda
  static void panLeft(float inc) {
    panRight(-inc);
  }
  // Movimiento hacia arriba
  static void panUp(float inc) {
    navigator.viewPos = navigator.view.location();
    navigator.viewTarget = navigator.view.target();
    navigator.updatePan(0, inc);
  }
  // Movimiento hacia abajo
  static void panDown(float inc) {
    panUp(-inc);
  }
  // Movimiento de zoom hacia dentro
  static void zoomIn(float inc) {
    navigator.viewPos = navigator.view.location();
    navigator.viewAxonRatio =
navigator.view.getAxonometricRatio();
    navigator.updateZoom(-inc);
  }
  // Movimiento de zoom hacia fuera
  static void zoomOut(float inc) {
    zoomOut(-inc);
  }
  // Actualiza la vista de la cámara. Debe invocarse
desde el método draw de Processing
  static void update() {
    rotateRight(horRotationInc);
    rotateDown(verRotationInc);
  }
}
class ControlFrame extends PApplet {
  int w;
  int h;
  int bcolor = 100;
  float margenH = 10;
  float margenV = 10;
  float separacionV = 20;
  float separacionHToggle = 50;
  float separacionHBoton = 80;
  int moduleAmmount = 14;
  int barPoints = 14;
  ControlP5 cp5;
  Object parent;

  ControlFrame(Object parent, int w, int h) {
    this.parent = parent;
    this.w = w;
    this.h = h;
  }
  void setup() {
    size(w, h);
    cp5 = new ControlP5(this);
    float i = 0;
    cp5.addSlider("topLines")
      .setRange(0, 150)
       .setValue(25)
        .setPosition(margenH, margenV + separacionV *
i)
         .setSize(300, 10)
          .setCaptionLabel("Top size");
    i++;
    cp5.addSlider("baseLines")
      .setRange(0.01, 350)
       .setValue(100)
        .setPosition(margenH, margenV + separacionV *
i)
         .setSize(300, 10)
          .setCaptionLabel("Base size");
    i++;
    cp5.addSlider("baseLine1")
      .setRange(0.01, 350)
       .setValue(100)
        .setPosition(margenH, margenV + separacionV *
i)
         .setSize(90, 10)
          .setCaptionLabel("B1");
    cp5.addSlider("baseLine2")
      .setRange(0.01, 350)
       .setValue(100)
        .setPosition(margenH + 105, margenV +
separacionV * i)
         .setSize(90, 10)
          .setCaptionLabel("B2");
    cp5.addSlider("baseLine3")
      .setRange(0.01, 350)
       .setValue(100)
        .setPosition(margenH + 210, margenV +
separacionV * i)
         .setSize(90, 10)
          .setCaptionLabel("B3");
    i++;
    cp5.addSlider("tension")
      .setRange(0, 150)
       .setValue(5)
        .setPosition(margenH, margenV + separacionV *
i)
         .setSize(300, 10)
          .setCaptionLabel("Bar tension");
    i++;
    cp5.addSlider("maxTension")
      .setRange(-1, 150)
       .setValue(-1)
        .setPosition(margenH, margenV + separacionV *
i)
         .setSize(300, 10)
          .setCaptionLabel("max tension");
    i++;
    cp5.addSlider("friction")
      .setRange(0, 1)
       .setValue(1)
        .setPosition(margenH, margenV + separacionV *
i)
         .setSize(300, 10)
          .setCaptionLabel("Bar friction");
    i++;
```

```
  cp5.addSlider("segmentSize")
    .setRange(0.01, 50)
     .setValue(6)
      .setPosition(margenH, margenV + separacionV *
i)
       .setSize(300, 10)
        .setCaptionLabel("Segment size");
  i++;
  cp5.addSlider("barHue")
    .setRange(0, 1)
     .setValue(1)
      .setPosition(margenH, margenV + separacionV *
i)
       .setSize(300, 10)
        .setCaptionLabel("Color");
  i++;
  cp5.addSlider("barPointsNumber")
    .setRange(1, 20)
     .setValue(18)
      .setPosition(margenH, margenV + separacionV *
i)
       .setSize(300, 10)
        .setCaptionLabel("Bar Points");
  i++;
  cp5.addSlider("moduleAmmount")
    .setRange(1, 50)
     .setValue(moduleAmmount)
      .setPosition(margenH, margenV + separacionV *
i)
       .setSize(300, 10)
        .setCaptionLabel("Module Ammount");
  i++;
  cp5.addSlider("cohesionDist")
    .setRange(0, 1000)
     .setValue(800)
      .setPosition(margenH, margenV + separacionV *
i)
       .setSize(90, 10)
        .setCaptionLabel("CD");
  cp5.addSlider("separationDist")
    .setRange(0, 500)
     .setValue(200)
      .setPosition(margenH + 105, margenV +
separacionV * i)
       .setSize(90, 10)
        .setCaptionLabel("SD");
  cp5.addSlider("alignDist")
    .setRange(0, 700)
     .setValue(500)
      .setPosition(margenH + 210, margenV +
separacionV * i)
       .setSize(90, 10)
        .setCaptionLabel("AD");

  i++;

  cp5.addSlider("cohesionRatio")
    .setRange(0, 0.01)
     .setValue(0.0025)
      .setPosition(margenH, margenV + separacionV *
i)
       .setSize(90, 10)
        .setCaptionLabel("CR");
  cp5.addSlider("separationRatio")
    .setRange(0, 0.03)
     .setValue(0.005)
      .setPosition(margenH + 105, margenV +
separacionV * i)
       .setSize(90, 10)
        .setCaptionLabel("SR");
  cp5.addSlider("alignRatio")
    .setRange(0, 0.01)
     .setValue(0.005)
      .setPosition(margenH + 210, margenV +
separacionV * i)
       .setSize(90, 10)
        .setCaptionLabel("AR");
  i++;
  float j = 0;
  cp5.addToggle("info")
    .setPosition(margenH + separacionHToggle * j,
margenV + separacionV * i)
     .setSize(40, 10)
      .setCaptionLabel("Info");
  j++;
  cp5.addToggle("hideLines")
    .setPosition(margenH + separacionHToggle * j,
margenV + separacionV * i)
     .setSize(40, 10)
      .setCaptionLabel("Hide lines");
  j++;
  cp5.addToggle("invertHue")
    .setPosition(margenH + separacionHToggle * j,
margenV + separacionV * i)
     .setSize(40, 10)
      .setCaptionLabel("Invert Hue");
  j++;
  cp5.addToggle("trail")
    .setPosition(margenH + separacionHToggle * j,
margenV + separacionV * i)
     .setSize(40, 10)
      .setCaptionLabel("Trail");
  j++;
  cp5.addToggle("toggleSaveImages")
    .setPosition(margenH + separacionHToggle * j,
margenV + separacionV * i)
     .setSize(40, 10)
      .setCaptionLabel("Save Img");
  j++;
  cp5.addToggle("toggleFixZ")
    .setPosition(margenH + separacionHToggle * j,
margenV + separacionV * i)
     .setSize(40, 10)
      .setCaptionLabel("Fix Z")
       .setValue(true);
  j++;
  i += 2;
  cp5.addSlider("setGravityForce")
```

```
      .setRange(0, 10)
       .setValue(5)
        .setPosition(margenH, margenV + separacionV *
i)
         .setSize(300, 10)
          .setCaptionLabel("Inverted Gravity")
           .lock();
    i++;
    j = 0;
    cp5.addButton("delete")
      .setPosition(margenH + separacionHBoton * j,
margenV + separacionV * i);
    j++;
    cp5.addButton("create")
      .setPosition(margenH + separacionHBoton * j,
margenV + separacionV * i);
    j++;
    cp5.addButton("center")
      .setPosition(margenH + separacionHBoton * j,
margenV + separacionV * i);
    j++;
    cp5.addButton("save")
      .setPosition(margenH + separacionHBoton * j,
margenV + separacionV * i);
    j++;
    cp5.addButton("exit")
      .setPosition(350, 600);
  }
  void draw() {
    background(bcolor);
  }
  ControlFrame() {
  }
  ControlP5 control() {
    return cp5;
  }
  void topLines(float value) {
    system.setTopLinesSize(value);
  }
  void baseLines(float value) {
    system.setBaseLinesSize(value);
  }
  void baseLine1(float value) {
    system.setBaseLine1Size(value);
  }
  void baseLine2(float value) {
    system.setBaseLine2Size(value);
  }
  void baseLine3(float value) {
    system.setBaseLine3Size(value);
  }
  void tension(float value) {
    system.setTension(value);
  }
  void maxTension(float value) {
    system.setMaxTension(value);
  }
  void friction(float value) {
    system.setFriction(value);
  }
  void delete(int value) {
    if (!system.isDeleted()) {
      IG.pause();
      system.delete();
      IG.resume();
    }
  }
  void create(int value) {
    if (system.isDeleted()) {
      IG.pause();
      system = new
TriangleAgentSystem(this.moduleAmmount,
this.barPoints);
      Toggle t = (Toggle)cp5.getController("toggleFixZ");
      system.setFixedZ(t.getState());
      IG.resume();
    }
  }
  void save(int value) {
    system.savePoints();
  }
  void exit(int value) {
    exit();
  }
  void center(int value) {
    IG.perspective();
  }
  void segmentSize(float value) {
    system.setSegmentSize(value);
  }
  void info(boolean value) {
    system.setInfo(value);
  }
  void barHue(float value) {
    system.setBarsHue(value);
  }

  void barPointsNumber(int value) {
    this.barPoints = value;
  }
  void invertHue(boolean value) {
    system.setInvertHue(value);
  }
  void trail(boolean value) {
    system.setTrail(value);
  }
  void hideLines(boolean value) {
    system.hideLines(value);
  }
  void toggleSaveImages(boolean value) {
    saveImages = value;
  }

  void toggleFixZ(boolean value) {
    system.setFixedZ(value);
    if (value) {
      cp5.getController("setGravityForce").unlock();
      gravity.del();
```

```
    } else {
      cp5.getController("setGravityForce").lock();
      gravity.del();
      gravityMag = 0;
    }
    gravity = new IGravity(0, 0, gravityMag);
  }

  void moduleAmmount(int value) {
    this.moduleAmmount = value;
  }

  void cohesionDist(float value) {
    system.cohesionDist(value);
  }

  void separationDist(float value) {
    system.separationDist(value);
  }

  void alignDist(float value) {
    system.alignDist(value);
  }


  void cohesionRatio(float value) {
    system.cohesionRatio(value);
  }

  void separationRatio(float value) {
    system.separationRatio(value);
  }

  void alignRatio(float value) {
    system.alignRatio(value);
  }

  void setGravityForce(float value) {
    gravityMag = value;
    gravity.del();
    gravity = new IGravity(0, 0, gravityMag);
  }
}
// Clase que simula el comportamiento de un material flexible unidimensional.
// Se modela como una serie de puntos que están unidos por lineas de un largo
// fijo. Los puntos tienden a enderezar sus posiciones en una línea recta.
class FlexibleBar extends IAgent {
  ArrayList<IParticle> particles;
  ArrayList<IStickLine> lines;
  ArrayList<IStraightenerCurve> curves;
  ArrayList<IText> labels;
  boolean isCircular;          // Dice si el final debe estar unido al inicio
  float particleFric;          // Friccion de los puntos al moverse por el espacio
  int weight = 3;          // Grosor de las líneas y los puntos
  float tension;          // Tensión con la que se enderezan los puntos
  float maxTension;          // Tensión máxima antes de quebrarse. -1 para no usar límite.
  float hue = 0.0;          // Color en reposo (0=rojo, 0.25=verde, 0.5=cian, 0.75=magenta)
  boolean invertHue = false; // Dice si la dirección en que cambia la tonalidad se invierte
  float hueChange = 1;       // Qué tanto cambia el color cuando se flexiona la barra
  boolean trail = false;     // Dice si dibuja un rastro del movimiento efectuado
  boolean info = false;      // Dibuja información adicional en pantalla
  int labelUpdateCount = 0;
  int labelRefreshRate = 20;

  FlexibleBar(ArrayList<IVec> controlPoints, float particleFric, float tension, float maxTension, boolean isCircular) {
    particles = new ArrayList();
    lines = new ArrayList();
    curves = new ArrayList();
    labels = new ArrayList();
    this.particleFric = particleFric;
    this.isCircular = isCircular;
    this.tension = tension;
    this.maxTension = maxTension;
    createParticles(controlPoints);
    createLines();
    createCurves();
  }

  FlexibleBar(ArrayList<IVec> controlPoints, float particleFric, float tension, float maxTension) {
    this(controlPoints, particleFric, tension, maxTension, false);
  }

  FlexibleBar(ArrayList<IVec> controlPoints, float particleFric, float tension) {
    this(controlPoints, particleFric, tension, -1, false);
  }

  void createParticles(ArrayList<IVec> controlPoints) {
    for (IVec cp : controlPoints) {
      IParticle p = new IParticle(cp);
      p.fric(particleFric);
      p.hsb(hue, 1.0, 1.0);
      p.weight(weight + 1);
      particles.add(p);
      IText label = new IText(" ", 2, p.pos(), new IVec(1, 0, 1));
      labels.add(label);
    }
  }

  void createLines() {
    int stop = 1;
```

```
    if (isCircular) {
      stop = 0;
    }
    if (lines.size() > 0) {
      for (IStickLine l : lines) {
        l.del();
      }
      lines.clear();
    }
    for (int i = 0; i < particles.size ()-stop; i++) {
      IStickLine l = new IStickLine(particles.get(i),
particles.get((i+1) % particles.size()));
      l.weight(weight);
      lines.add(l);
    }
  }

  void createCurves() {
    int stop = 2;
    if (isCircular) {
      stop = 0;
    }
    if (curves.size() > 0) {
      for (IStraightenerCurve c : curves) {
        c.del();
      }
      curves.clear();
    }
    for (int i = 0; i < particles.size ()-stop; i += 1) {
      IStraightenerCurve c = new
IStraightenerCurve(particles.get(i), particles.get((i+1)
% particles.size()), particles.get((i+2) %
particles.size()));
      c.tension(tension);
      c.maxTension(maxTension);
      c.hide();
      curves.add(c);
    }
  }

  void fixParticle(int i) {
    IParticle p = particles.get(i);
    p.fix();
  }

  void unfixParticle(int i) {
    IParticle p = particles.get(i);
    p.unfix();
  }

  IParticle getParticle(int i) {
    return particles.get(i);
  }

  void replaceParticle(int i, IParticle p) {
    IParticle old = particles.get(i);
    p.fric(particleFric);
    p.hsb(hue, 1.0, 1.0);
    p.weight(weight + 1);
    p.pos(old.pos().dup());
    if (old.fixed()) {
      p.fix();
    }
    particles.set(i, p);
    createLines();
    createCurves();
    old.del();
  }

  void replaceParticleWithoutDelete(int i, IParticle p) {
    IParticle old = particles.get(i);
    p.fric(particleFric);
    p.hsb(hue, 1.0, 1.0);
    p.weight(weight + 1);
    p.pos(old.pos().dup());
    if (old.fixed()) {
      p.fix();
    }
    particles.set(i, p);
    createLines();
    createCurves();
  }
  void update() {
    labelUpdateCount = labelUpdateCount == 0?
labelRefreshRate: labelUpdateCount - 1;
    for (int i = 0; i < curves.size (); i++) {
      IStraightenerCurve c = curves.get(i);
      IParticleI p1 = c.pt1();
      IParticleI p2 = c.pt2();
      IParticleI p3 = c.pt3();
      IVec dif1 = p1.pos().dup().sub(p2.pos());
      IVec dif2 = p3.pos().dup().sub(p2.pos());
      float angle = (float)dif1.angle(dif2);
      float h = hue + (invertHue? 1 : -1) * (angle / (1* PI));
      ((IParticle)p2).hsb(h, 1.0, 1.0);
      if (info && labelUpdateCount == 0) {
        IText l = labels.get((i + 1) % labels.size());
        float flex = abs(degrees(PI - angle));
        l.text("  " + Float.toString(flex));
        l.hsb(h + 0.5, 1.0, 1.0);
        l.show();
      } else if (!info) {
        IText l = labels.get((i + 1) % labels.size());
        l.hide();
      }
    }
    if (!isCircular) {
      IParticle p = particles.get(0);
      //p.hsb(hue, 1.0, 1.0);
      IParticle p2 = particles.get(particles.size() - 1);
      //p2.hsb(hue, 1.0, 1.0);
    }
    for (int i = 0; i < lines.size (); i++) {
      IStickLine l = lines.get(i);
      IParticle p1 = (IParticle)l.pt1();
      IParticle p2 = (IParticle)l.pt2();
      IColor c = p1.getColor().dup();
      c.blend(p2.getColor());
```

```
      l.setColor(c);
      if (trail) {
        ILineAgent line = new ILineAgent(l.pos1().dup(),
l.pos2().dup().sub(l.pos1()));
        line.setColor(c, 0.2);
      }
    }
  }
  void setTension(float tension) {
    for (IStraightenerCurve c : curves) {
      c.tension(tension);
    }
  }
  void setFriction(float friction) {
    for (IParticle p : particles) {
      p.friction(friction);
    }
  }
  void setSegmentSize(float size) {
    for (IStickLine s : lines) {
      s.stickDynamics.len = size;
    }
  }
  void setMaxTension(float tension) {
    for (IStraightenerCurve c: curves) {
      c.tension(tension);
    }
  }
  void delete() {
    for (IText t : labels) t.del();
    for (IStraightenerCurve c : curves) c.del();
    for (IStickLine s : lines) s.del();
    for (IParticle p : particles) p.del();
  }
  void push(IVec force) {
    for (IParticle p : particles) {
      p.push(force);
    }
  }
  void move(IVec force) {
    for (IParticle p : particles) {
      p.pos().add(force);
    }
  }
}
class FlexibleBarSystem extends IAgent {
  ArrayList<StickyFlexibleBar> bars;
  FlexibleBarSystem() {
    bars = new ArrayList();
  }
  void add(StickyFlexibleBar bar) {
    bars.add(bar);
  }
  void update() {
    for (StickyFlexibleBar b1 : bars) {
      for (StickyFlexibleBar b2 : bars) {
        if (b1 != b2) {
          b1.join(b2);
        }
      }
    }
  }
}
class StickyFlexibleBar extends FlexibleBar {
  float threshold; // Distancia mínima para que la barra
se una con otra
  float sharedDistance; // Distancia mínima de una
partícula con una partícula compartida para que se una
a otra barra
  boolean joined;
  boolean attractor;
  float attraction;

  StickyFlexibleBar(ArrayList<IVec> controlPoints, float
particleFric, float tension, float maxTension, boolean
isCircular, float threshold, float sharedDistance, float
attraction) {
    super(controlPoints, particleFric, tension,
maxTension, isCircular);
    this.threshold = threshold;
    this.sharedDistance = sharedDistance;
    joined = false;
    attractor = true;
    this.attraction = attraction;
  }

  StickyFlexibleBar(ArrayList<IVec> controlPoints, float
particleFric, float tension, float maxTension, boolean
isCircular, float threshold, float sharedDistance) {
    this(controlPoints, particleFric, tension, maxTension,
isCircular, threshold, sharedDistance, 0);
    attractor = false;
  }
  void join(StickyFlexibleBar b) {
    ArrayList<IParticle> shared = sharedParticles(b);
    for (IParticle p1 : particles) {
      for (IParticle p2 : b.particles) {
        if (attractor) {
          IVec v1 = p1.pos.dup();
          IVec v2 = p2.pos.dup();
          IVec force = v1.sub(v2);
          double dist = v1.dist(v2);
          double strength = attraction * 1/(dist*dist);
          force = force.mul(-strength/2);
          p1.addForce(force);
          p2.addForce(force.dup().mul(-1));
        }
        if (joinDistance(p1, p2)) {
          if (outsideSharedDistance(p1, shared)) {
            int pos = particles.indexOf(p1);
            if (pos != -1) {
              replaceParticle(pos, p2);
              joined = true;
              b.joined = true;
            }
          }
        }
      }
```

```
    }
  }
  ArrayList<IParticle>
sharedParticles(StickyFlexibleBar b) {
    ArrayList<IParticle> res = new ArrayList();
    for (IParticle p : particles) {
      if (b.particles.contains(p)) {
        res.add(p);
      }
    }
    return res;
  }
  boolean joinDistance(IParticle p1, IParticle p2) {
    double distance = p1.pos().dist(p2.pos());
    return distance < threshold;
  }
  // recibe una partícula y una lista de partículas
compartidas
  // dice si se encuentra a una distancia mayor que
sharedDistance con todas ellas
  boolean outsideSharedDistance(IParticle p,
ArrayList<IParticle> shared) {
    for (IParticle sharedParticle : shared) {
      double distance =
p.pos().dist(sharedParticle.pos());
      if (distance <= sharedDistance) {
        return false;
      }
    }
    return true;
  }
  boolean isJoined() {
    return joined;
  }
}
class TriangleAgent extends IAgent {
  TriangleModule module;
  double cohesionDist = 800;
  double cohesionRatio = 0.0025;
  double alignmentDist = 500;
  double alignmentRatio = 0.005;
  double separationDist = 200;
  double separationRatio = 0.005;
  ArrayList<TriangleAgent> agents;
  ArrayList<TriangleAgent> connectedAgents[];
  float joinDistance = 50;
  boolean fixedZ;
  int joined[] = {
    0, 0, 0
  };

  int maxConnections;

  TriangleAgent(ArrayList<IVec> points, int barPoints,
ArrayList<TriangleAgent> agents, int maxConnections)
{
    module = new TriangleModule(points, barPoints);
    this.agents = agents;
    fixedZ = false;
    this.maxConnections = maxConnections;
    connectedAgents = new ArrayList[3];
    for (int i = 0; i < 3; i++) {
      connectedAgents[i] = new ArrayList();
      connectedAgents[i].add(this);
    }
  }
  void push(IVec force) {
    for (FlexibleBar b : module.bars) {
      b.push(force);
    }
  }
  void move(IVec force) {
    for (FlexibleBar b : module.bars) {
      b.move(force);
    }
  }
  void update() {
    if (fixedZ) {
      for (IParticle p : module.baseJoints) {
        p.pos().z(0);
      }
    }
    join();
    flock();
  }
  IVec getPos() {
    IVec p0 = module.baseJoints.get(0).pos();
    IVec p1 = module.baseJoints.get(1).pos();
    IVec p2 = module.baseJoints.get(2).pos();
    IVec center = new IVec(0, 0, 0);
    center.add(p0);
    center.add(p1);
    center.add(p2);
    center.div(3);
    return center;
  }
  IVec getVel() {
    IVec vel = module.baseJoints.get(0).vel().dup();
    vel.add(module.baseJoints.get(1).vel());
    vel.add(module.baseJoints.get(2).vel());
    vel.div(3);
    return vel;
  }
  void cohere() {
    IVec average = new IVec(0, 0, 0);
    int count = 0;
    for (TriangleAgent t : agents) {
      if (t != this) {
        if (t.getPos().dist(getPos()) < cohesionDist) {
          average.add(t.getPos());
          count++;
        }
      }
    }
    if (count > 0) {
      average.div(count);
      average.sub(getPos());
      average.mul(cohesionRatio);
```

```
    move(average);
   }
  }
  void separate() {
   IVec average = new IVec(0, 0, 0);
   int count = 0;
   for (TriangleAgent t : agents) {
    if (t != this) {
     double dist = t.getPos().dist(getPos());
     if (dist < separationDist) {

average.add(getPos().dif(t.getPos()).len(separationDis
t - dist));
      count++;
     }
    }
   }
   if (count > 0) {
    average.mul(separationRatio / count);
    move(average);
   }
  }
  void align() {
   IVec average = new IVec(0, 0, 0);
   int count = 0;
   for (TriangleAgent t : agents) {
    if (t != this) {
     if (t.getPos().dist(getPos()) < alignmentDist) {
      average.add(t.getVel());
      count++;
     }
    }
   }
   if (count > 0) {
    average.div(count);
    average.sub(getVel());
    average.mul(alignmentRatio);
    move(average);
   }
  }
  void flock() {
   cohere();
   separate();
   align();
  }


  void join() {
   for (TriangleAgent t : agents) {
    if (t != this) {
     checkAndJoinVertices(t);
    }
   }
  }
  void checkAndJoinVertices(TriangleAgent t) {
   for (int i = 0; i < 3; i++) {
    if (connectedAgents[i].size() < maxConnections &&
!connectedAgents[i].contains(t)) {
     for (int j = 0; j < 3; j++) {
      if (t.connectedAgents[j].size() +
connectedAgents[i].size() < t.maxConnections) {
       IVec p1 = module.getJointParticle(i).pos();
       IVec p2 = t.module.getJointParticle(j).pos();
       double dist = p2.dist(p1);
       if (dist < joinDistance) {
        joinMultipleConnections(i, j, t);
       }
      }
     }
    }
   }
  }
  void joinMultipleConnections(int vertex, int
otherVertex, TriangleAgent t) {
   IParticle old = module.baseJoints.get(vertex);
   IParticle p = t.module.getJointParticle(otherVertex);

t.connectedAgents[otherVertex].addAll(connectedAge
nts[vertex]);
   for (TriangleAgent connectedTriangle :
connectedAgents[vertex]) {
    int localIndex =
connectedTriangle.module.baseJoints.indexOf(old);

connectedTriangle.module.replaceJoint(localIndex, p);
    connectedTriangle.connectedAgents[localIndex] =
t.connectedAgents[otherVertex];
   }
  }
  void delete() {
   module.delete();
  }
  void setFixedZ(boolean fixedZ) {
   this.fixedZ = fixedZ;
  }
}
```