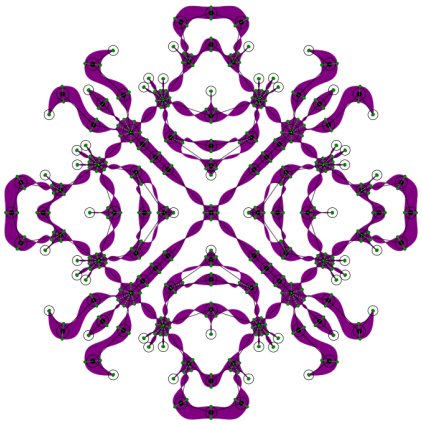
ULPGC
Universidad de
Las Palmas de
Gran Canaria

ESCUELA DE
INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

Demo Online para la representación interactiva de formas abstractas



Grado en Ingeniería Informática

AUTOR: Néstor Maneiro Pérez

TUTORIZADO POR:
Agustín Rafael Trujillo Pino y Nelson Monzón López

Enero 2024

Agradecimientos

Tanto a Agustín Rafael Trujillo Pino y a Nelson Monzon Lopez, por su participación, ayuda y animos ofrecidos en este proyecto y a Luis Álvarez León, por su participación en el proyecto, en gran parte del apartado de C++ y con la idea inicial.

Índice general

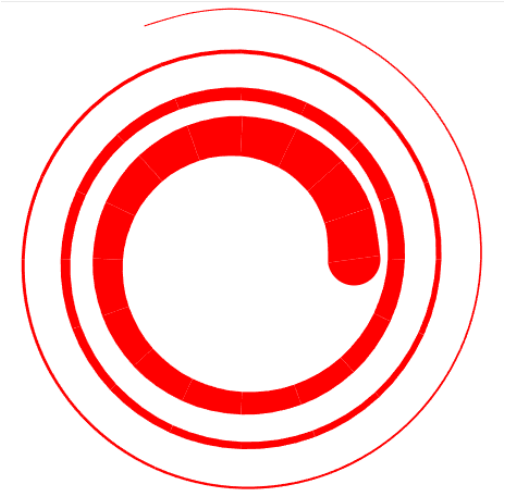
Capítulo 1

Introducción

1.1. Descripción del proyecto

Este proyecto se centra en el desarrollo de un entorno web dedicado a la generación de figuras vectoriales mediante el uso de curvas de Bézier. La utilización de este enfoque permite la creación de imágenes de alta calidad con un tamaño reducido, lo que resulta especialmente beneficioso en aplicaciones como la creación de logotipos. Además, brinda al usuario la capacidad de interactuar de manera cómoda y sencilla con los diversos nodos que componen estas imágenes, facilitando la modificación de parámetros de manera intuitiva.

La lógica subyacente en la creación de las figuras ha sido implementada en código nativo de C++, el cual ha sido adaptado al entorno web mediante el uso de WebAssembly. Esta elección no solo simplifica significativamente la tarea de trasladar la lógica existente, sino que también permite ejecutar este potente código nativo de manera más eficiente y optimizada en comparación con JavaScript.



1.2. Competencias

A continuación, se detalla cómo el desarrollo y la implementación de este proyecto han contribuido al cumplimiento de las competencias específicas establecidas para este trabajo:

1. **CI6:** *Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.*

   En este proyecto, la implementación de operaciones de creación y manipulación de figuras SVG implica la aplicación directa de algoritmos en C++. El análisis de la idoneidad y complejidad de estos algoritmos se presenta en el desglose detallado de cada objetivo, destacando la eficiencia y eficacia en la manipulación de figuras geométricas.

2. **CI8:** *Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.*

   La planificación y desarrollo de la interfaz web que permite a los usuarios interactuar con las figuras SVG generadas en tiempo real refleja la capacidad para construir aplicaciones de forma segura y eficiente. La elección de tecnologías, la arquitectura del sistema y el diseño robusto de la interfaz son ejemplos de cómo se aborda esta competencia.

3. **TI1:** *Capacidad para comprender el entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones.*

   La creación de una aplicación que permite la creación y manipulación de figuras SVG de manera dinámica responde directamente a la comprensión del entorno y las necesidades de los usuarios finales. El proyecto se adapta a las necesidades específicas de aquellos que buscan una herramienta versátil para la visualización y manipulación de figuras SVG.

4. **TI3:** *Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.*

   El enfoque en la usabilidad y accesibilidad en el desarrollo de la interfaz web demuestra la aplicación de metodologías centradas en el usuario. Se han utilizado técnicas de diseño centradas en el usuario para asegurar una experiencia intuitiva y amigable al interactuar con las figuras SVG.

5. **TI5:** *Capacidad para seleccionar, desplegar, integrar y gestionar sistemas de información que satisfagan las necesidades de la organización, con los criterios de coste y calidad identificados.*

   La elección de tecnologías y la integración de C++ y WebAssembly para la creación de una aplicación que satisface las necesidades identificadas en el contexto de manipulación de figuras SVG son ejemplos de cómo se aborda esta competencia.

6. **TI6:** *Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.*

   La concepción de una aplicación web que utiliza tecnologías de red para permitir la interacción en tiempo real con figuras SVG abarca conceptos de Internet, web y computación móvil, contribuyendo así al cumplimiento de esta competencia.

1.3. Trabajo Previo y Evolución del Proyecto

En este proyecto, partimos de un código en C++, capaz de generar figuras vectoriales utilizando un concepto llamado "Shape". Este nos ofrece una representación de formas geométricas compuestas por círculos, estos círculos interconectados en segmentos, permiten la formación de la figura. Cada entrada en el conjunto de datos sigue un formato específico que se describe a continuación, dando los datos más importantes:

```
Número de Círculos (N):
  - Indica la cantidad total de círculos en la forma.

Coordenadas de los Círculos (X, Y):
  - Las siguientes N líneas contienen las coordenadas (X, Y)
  de cada círculo en el plano.

Radio de los Círculos:
  - Las siguientes N líneas representan los radios de cada círculo.

Smooth Factor de los Círculos:
  - Las siguientes N líneas indican el "smooth factor" de cada círculo,
  que influye en la suavidad de la forma.
```

Este "Shape", mediante diversos datos, tenía la capacidad de producir figuras de manera específica. La propuesta se centró en la expansión de este código, incorporando opciones de edición y facilitando la interacción para que cualquier usuario, incluso sin conocimientos previos, pudiera generar y modificar estas figuras.

La visión de crear un entorno web surgió con el objetivo de proporcionar una plataforma donde los usuarios pudieran realizar ediciones de manera intuitiva y visualizar los cambios en tiempo real. Este enfoque permitiría transformar un conjunto de datos abstractos en una figura manipulable, brindando la posibilidad de redimensionar, mover y manipular la forma resultante.

La clave radica en diseñar una interfaz de usuario amigable y comprensible, que posibilite la modificación de parámetros como el radio, la posición de los círculos y otros factores relevantes. Además, se contempla la opción de generar figuras de forma aleatoria, ofreciendo así un amplio espectro de posibilidades y facilitando la exploración creativa por parte del usuario.

En resumen, este proyecto se enfoca en la creación de un entorno web interactivo que simplifica la generación y edición de figuras vectoriales, haciendo accesible esta capacidad incluso para aquellos usuarios sin conocimientos técnicos previos.

1.3.1. Contexto y Antecedentes

El proyecto se inserta en un contexto amplio de aplicaciones relacionadas con la manipulación y generación de gráficos vectoriales en entornos web. A continuación, se exploran algunos antecedentes y aplicaciones similares:

Herramientas de Diseño Gráfico:

- ✔ **Adobe Illustrator y CorelDRAW:** Son programas de diseño gráfico que permiten la creación de ilustraciones vectoriales mediante el uso de herramientas similares, incluyendo la manipulación de curvas de Bézier.
- ✔ **Inkscape:** Una alternativa de código abierto que se centra en la edición de gráficos vectoriales, ofreciendo funcionalidades similares a las herramientas comerciales.

Librerías de Desarrollo:

- ✔ **D3.js y SVG.js:** Bibliotecas JavaScript utilizadas para la manipulación y representación de gráficos vectoriales en la web.
- ✔ **Canvas API de HTML5:** Permite la creación de gráficos, incluyendo gráficos vectoriales, directamente en el elemento Canvas de HTML5.

Modelado 3D y Animación:

- ✔ **Blender y Autodesk Maya:** Herramientas de modelado y animación que utilizan principios similares, pero en un contexto tridimensional.
- ✔ **Three.js:** Biblioteca de JavaScript para crear gráficos 3D interactivos en la web.

Proyectos de Experimentación Creativa:

- ✔ **Generación de Arte Aleatorio:** Proyectos artísticos que exploran la generación de formas y patrones aleatorios, donde las curvas de Bézier pueden desempeñar un papel crucial.

1.4. Tecnologias utilizadas

1.4.1. Selección de Tecnologías

Dada la necesidad de desarrollar un entorno web interactivo, se hizo imprescindible utilizar las tecnologías fundamentales del desarrollo web, como HTML, CSS y JavaScript. Para posibilitar la compilación del código C++ en este entorno, se optó por WebAssembly. Esta tecnología permitió la ejecución eficiente del código C++ en los navegadores web, proporcionando un puente efectivo entre el código existente y el entorno web.

En el ámbito de las herramientas específicas para la visualización y manipulación de gráficos vectoriales en un entorno web, la elección de WebAssembly se complementó con la adopción de D3.js. Esta biblioteca especializada se reveló como una poderosa herramienta para la manipulación dinámica de SVG en navegadores web, ofreciendo un conjunto completo de funcionalidades.

1.4.2. Exploración de Alternativas

Aunque la combinación de WebAssembly y D3.js demostró ser la elección más adecuada para este proyecto, es esencial explorar alternativas que podrían haberse considerado:

Canvas API de HTML5: Esta API permite dibujar gráficos, incluyendo gráficos vectoriales, directamente en el elemento Canvas de HTML5. Aunque eficiente, podría no ofrecer la misma flexibilidad y dinamismo proporcionado por D3.js.

SVG.js: Como una biblioteca ligera para manipular y animar SVG en el navegador, SVG.js es robusta, pero podría carecer de la integración fluida con código C++ a través de WebAssembly, lo que podría requerir más esfuerzo de desarrollo.

Three.js: Si bien se especializa en gráficos 3D, Three.js podría haber sido una opción si el proyecto hubiera necesitado representaciones tridimensionales de figuras. Sin embargo, su enfoque en gráficos 3D podría haber introducido complejidades innecesarias para aplicaciones principalmente basadas en gráficos 2D.

La elección de tecnologías y herramientas siempre es un equilibrio entre los requisitos específicos del proyecto y las capacidades ofrecidas por cada opción disponible. En este caso, la combinación de WebAssembly y D3.js proporcionó una solución eficaz y eficiente para la visualización y manipulación de figuras SVG en un entorno web.

1.5. Avance

1.5.1. Evolución del Proyecto

Desde su concepción inicial, el proyecto ha experimentado una transformación significativa, inicialmente centrado en la visualización de figuras en un entorno web propio. A lo largo de su desarrollo, se ha enriquecido con funcionalidades que permiten una interacción más dinámica con las figuras, posibilitando modificaciones en tiempo real, todo ello sin perder de vista los principios fundamentales establecidos anteriormente.

1.5.2. Desafíos Superados y Decisiones de Diseño

El proyecto se enfrentó a desafíos notables, como la integración de código nativo en C++ en un entorno web, para lo cual WebAssembly desempeñó un papel crucial. Se llevaron a cabo optimizaciones tanto en el uso de WebAssembly como en la gestión eficiente de la memoria caché del navegador.

La evolución del proyecto fue iterativa, con revisiones periódicas durante los sprints, lo que permitió sugerir cambios y mejoras de manera constante. Este enfoque favoreció un crecimiento rápido y sólido del proyecto, destacándose, por ejemplo, en la transformación de la interfaz gráfica de un entramado complejo de botones a una interfaz más amigable.

Adquirir nuevas habilidades fue una necesidad para superar los desafíos emergentes, enriqueciendo tanto el proyecto como el desarrollo personal.

1.5.3. Lecciones Aprendidas

A lo largo de este proyecto, se han extraído lecciones valiosas que van más allá de las enseñanzas académicas convencionales. La aplicación de una metodología ágil proporcionó una experiencia que simulaba de cerca un entorno laboral, subrayando la importancia de la comunicación efectiva, la escritura de código limpio y bien organizado, así como la valoración de los sprints y sus revisiones.

Destaco especialmente el aprendizaje en WebAssembly, una herramienta poderosa, y la comprensión profunda de la optimización, que se revela significativa a medida que el proyecto escala. Estas lecciones no solo han fortalecido el proyecto de investigación, sino que también han enriquecido mi formación personal de manera integral.

Este testimonio refleja no solo el crecimiento del proyecto, sino también el crecimiento personal y profesional que ha acompañado esta experiencia única.

1.6. Presupuesto del Proyecto

A continuación se presenta un desglose estimado de los costos asociados con el desarrollo e implementación del proyecto.

Según la tabla salarial del convenio de Despachos y Oficinas, vigente desde el 1 de enero de 2024, a 31 de diciembre de 2024, un ingeniero percibe unos ingresos salariales de 28.799,59 euros brutos anuales. El trabajador percibirá la parte proporcional correspondiente a la duración del contrato de trabajo.

1.6.1. Desarrollo del Software

Ingeniero de Frontend (HTML, CSS, JS):

- ✔ Horas estimadas de desarrollo: 160 (1 mes, 40 semanales * 4 semanas)
- ✔ **Subtotal mensual: 2399,96 euros brutos**

Ingeniero de Backend (C++):

- ✔ Horas estimadas de desarrollo: 160 (1 mes, 40 semanales * 4 semanas)
- ✔ **Subtotal mensual: 2399,96 euros brutos**

1.6.2. Consultoría Matemática

- ✔ Costo de consulta (5 horas): 300 euros

1.6.3. Hospedaje con Dominio Propio

- ✔ Registro del dominio: 15 euros por año
- ✔ Hospedaje en otro proveedor: 10 euros por mes
- ✔ **Subtotal anual: (15 euros por año) + (10 euros por mes $\times$ 12 meses) = 135 euros**

1.6.4. Total del Presupuesto

- ✔ Desarrollo del software: **4800 euros**
- ✔ Consultorías: **300 euros**
- ✔ Hospedaje y dominio: **135 euros**
- ✔ **Total: 5.235 euros**

Capítulo 2

Recursos Utilizados

Ilustración 2.1: Ejemplo Figura Persona

2.1. WebAssembly

WebAssembly (WASM) es un estándar abierto que posibilita la ejecución de código nativo de lenguajes como C, C++, y Rust en entornos web. Su principal contribución radica en ofrecer un rendimiento cercano al código nativo de la máquina, permitiendo realizar operaciones computacionales intensivas de manera eficiente en navegadores. Esta capacidad de ejecutar código compilado de otros lenguajes ha sido esencial para optimizar el rendimiento y la eficiencia en proyectos, como el desarrollo de aplicaciones que manipulan gráficos vectoriales, como es el caso de SVG. WebAssembly se integra de manera armoniosa con JavaScript, permitiendo una coexistencia efectiva y aprovechando las fortalezas de ambos para ofrecer experiencias web más rápidas y potentes.

En resumen, WebAssembly ha sido fundamental para la ejecución eficiente de operaciones complejas, como la manipulación de gráficos vectoriales, en entornos web, contribuyendo así al éxito de este proyecto.

2.2. Tecnologías Web Adoptadas

Durante el desarrollo de este proyecto, se hizo uso de diversas tecnologías web clave para construir una aplicación interactiva y dinámica. La selección de estas tecnologías se basó en su amplia adopción y su capacidad para ofrecer una experiencia de usuario excepcional.

2.2.1. HTML, CSS y JavaScript

HTML, CSS y JavaScript se constituyeron como los pilares fundamentales para la construcción de la interfaz de usuario. Estas tecnologías, reconocidas por su versatilidad y compatibilidad universal, posibilitaron la creación de páginas web interactivas y visualmente atractivas. HTML (Lenguaje de Marcado de Hipertexto) proporcionó la estructura básica, CSS (Hojas de Estilo en Cascada) permitió el diseño y la presentación, mientras que JavaScript añadió dinamismo y funcionalidades interactivas.

2.2.2. D3.js

D3.js destacó como la biblioteca principal para la manipulación de gráficos vectoriales (SVG). Su capacidad única para vincular datos a elementos del DOM y crear visualizaciones dinámicas fue fundamental para la representación y manipulación eficientes de figuras SVG en tiempo real. En la evaluación de alternativas, D3.js demostró ser la elección ideal por su flexibilidad y manejo eficaz de conjuntos de datos complejos.

2.2.3. Ventajas de D3.js

D3.js proporcionó ventajas significativas, permitiendo una vinculación dinámica de datos a elementos SVG. Su flexibilidad posibilitó la creación de visualizaciones personalizadas y complejas. La comunidad activa y los recursos disponibles hicieron que D3.js sobresaliera como una elección robusta para el proyecto.

Es importante destacar que la elección de estas tecnologías, especialmente D3.js, se vio influenciada por la existencia de un código base estructurado en C++. La transición a un entorno web se simplificó gracias a WebAssembly, lo que permitió integrar el código C++ existente. Aunque esta elección presentó claras ventajas, se evaluaron con cuidado las consideraciones y limitaciones asociadas al uso de estas tecnologías en el contexto específico del proyecto.

2.3. Otros Recursos Utilizados

Además de las tecnologías mencionadas, se utilizaron recursos adicionales para garantizar el éxito del proyecto.

2.3.1. Hardware

El desarrollo del proyecto se llevó a cabo utilizando computadoras personales y dispositivos móviles para las pruebas. La diversidad de dispositivos garantizó que la aplicación fuera accesible y funcional en diferentes entornos, contribuyendo a la calidad y versatilidad del producto final.

2.3.2. Herramientas de Desarrollo

Para optimizar el desarrollo y la gestión del código, se optó por emplear un entorno de desarrollo integrado (IDE), y la elección principal fue WebStorm. Esta decisión se fundamentó en la robusta compatibilidad de WebStorm con las tecnologías web y su conjunto completo de características, que incluyen herramientas de depuración y una integración sólida con sistemas de control de versiones.

WebStorm se destacó no solo por su capacidad para proporcionar un entorno de desarrollo intuitivo y eficiente, sino también por facilitar la implementación de un servidor local a través de su función LiveServer. Esta característica resultó invaluable al realizar pruebas en tiempo real, permitiendo una iteración ágil y eficaz en el desarrollo del proyecto.

Además, la utilización de GitHub Pages como servidor para el proyecto ofreció la ventaja adicional de permitir la verificación del rendimiento y la funcionalidad del entorno web en diversos dispositivos. Esta versatilidad contribuyó a garantizar una experiencia consistente y óptima para los usuarios finales.

En resumen, la elección de WebStorm como IDE central y la combinación estratégica con GitHub Pages no solo simplificaron el desarrollo, sino que también proporcionaron herramientas efectivas para realizar pruebas y garantizar la calidad del proyecto en diferentes contextos y dispositivos. Este enfoque integral respaldó la eficiencia y la calidad del desarrollo a lo largo del proyecto.

2.3.3. Conexión a Internet

La conexión a Internet desempeñó un papel crucial para acceder a recursos en línea, compartir avances y realizar pruebas en entornos colaborativos. La conexión estable garantizó una comunicación fluida y eficiente durante todo el desarrollo.

2.3.4. Dispositivos Móviles

La aplicación se sometió a pruebas exhaustivas en dispositivos móviles para garantizar la compatibilidad y la experiencia del usuario en diferentes tamaños de pantalla y plataformas. Esto incluyó la verificación de la versión móvil de la aplicación para garantizar un rendimiento óptimo en dispositivos móviles.

Estos recursos, combinados con las tecnologías web seleccionadas, jugaron un papel integral en el desarrollo exitoso de la aplicación, asegurando su funcionalidad, rendimiento y accesibilidad.

Capítulo 3

Estado Actual y Objetivos Iniciales

3.1. Descripción del Estado Actual

En el panorama actual, si bien existen varias herramientas para la manipulación de imágenes vectoriales, el proyecto que presentamos destaca por su excepcional facilidad de uso y versatilidad. Proporciona a los usuarios la capacidad de crear y modificar imágenes vectoriales de manera intuitiva, todo desde un entorno web sencillo y amigable.

Lo que distingue a este proyecto es la incorporación de C++ como lenguaje nativo, aprovechando su potencia para realizar cálculos precisos y eficientes en el ámbito de la manipulación de imágenes vectoriales. Esta elección de lenguaje no solo contribuye a la robustez del sistema, sino que también permite una experiencia de usuario fluida y enriquecedora.

A diferencia de otras herramientas, este proyecto busca proporcionar a los usuarios una solución integral, permitiéndoles tanto la creación como la modificación de imágenes vectoriales de forma rápida e intuitiva. La simplicidad de su entorno web hace que la manipulación de gráficos sea accesible para usuarios con diversos niveles de experiencia, convirtiéndolo en una herramienta valiosa en el ámbito de diseño y edición de imágenes vectoriales.

3.2. Formato SVG y Otros Formatos

Las imágenes SVG son gráficos vectoriales escalables y se almacenan como código XML que describe la geometría y las propiedades visuales de la imagen. Este formato basado en vectores es muy eficiente en términos de tamaño de archivo, ya que solo almacena la información necesaria para representar la imagen y no requiere píxeles individuales como en las imágenes de mapa de bits como las PNG. La ventaja de reducción de tamaño puede ser significativa, especialmente para imágenes simples o logotipos. En cambio, las imágenes PNG son mapas de bits que almacenan información sobre cada píxel en la imagen, lo que puede llevar a tamaños de archivo más grandes, especialmente para imágenes complejas o con detalles finos.

Objetivos Iniciales del TFG

Los objetivos iniciales trazados para este proyecto se centran en la integración de tecnologías **C++** y web para la creación dinámica de figuras **SVG**. Estos objetivos y su implementación se detallan a continuación:

1. **Implementación de Funciones en C++:** Desarrollar funciones en **C++** que permitan la generación de figuras geométricas a partir de un *shape* definido. Esto implica la codificación de algoritmos eficientes para la construcción y manipulación de elementos **SVG**.
2. **Desarrollo de Interfaz Web:** Crear una interfaz web que posibilite la interacción eficiente con las figuras **SVG** generadas en tiempo real. Esto incluye la implementación de controles y herramientas que faciliten la manipulación y visualización de las figuras.
3. **Facilitación de Creación y Edición:** Proporcionar herramientas intuitivas y accesibles para la creación y edición de figuras **SVG**. Esto implica la implementación de opciones que permitan a los usuarios realizar modificaciones de manera fácil e intuitiva, fomentando la creatividad y la experimentación.
4. **Gestor Gráfico Interactivo:** Desarrollar un gestor gráfico interactivo que permita a los usuarios trabajar con **SVG** de manera similar a otros programas de edición de imágenes. Este gestor gráfico proporcionará funcionalidades para la creación, modificación y gestión de gráficos vectoriales, especialmente útil para la creación y edición de logos.

Estos objetivos iniciales reflejan la ambición de crear una herramienta integral que combine eficientemente la potencia de **C++** con la accesibilidad y versatilidad del entorno web para la manipulación creativa de gráficos **SVG**.

Capítulo 4

Marco teórico

4.1. Representación vectorial de formas

Una forma (o silueta) se puede representar como una imagen de píxeles donde asignamos un color distinto del fondo a los píxeles del interior de la forma, el formato habitual de imágenes para almacenar una forma de esta manera es PNG. Las principales limitaciones de esta representación son las siguientes:

- ✔ Cuando aumentamos el tamaño de la forma haciendo un zoom, la forma sale pixelada y pierde calidad.
- ✔ La edición de la forma resulta compleja porque para modificarla tendríamos que hacerlo pixel a pixel.

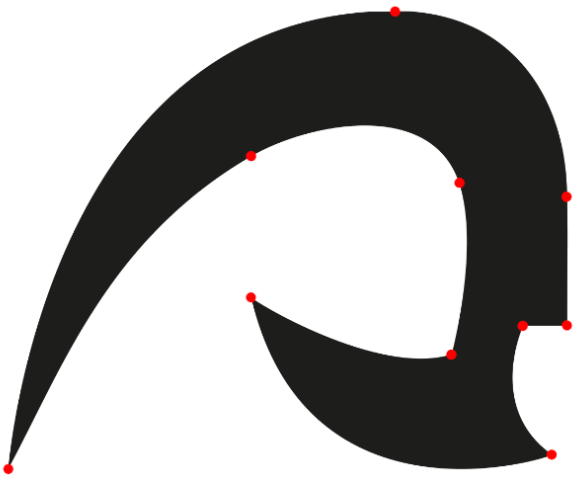


Ilustración 4.1: Logo ULPGC vectorizado

Una forma mucho más eficiente y versátil de representar una forma es la representación

vectorial, donde la forma se define a través de una colección de puntos sobre su borde y unas curvas continuas que unen esos puntos para definir por completo el borde de la forma. Las curvas que se usan habitualmente son las curvas cúbicas de Bézier. En la figura 4.1, ilustramos la representación vectorial del logo de la ULPGC donde los puntos rojos representan la colección de puntos seleccionados sobre el borde de la forma y entre cada dos de estos puntos, el contorno de la forma queda definido por una curva de Bézier que se parametriza usando los puntos de control. Es decir, que para almacenar una forma vectorial lo que tengo que guardar son las coordenadas de los puntos seleccionados del borde de la forma y las coordenadas de los puntos de control de cada curva de Bézier. Esto es lo que hace el formato SVG, que es el formato estándar para almacenar y gestionar el formato vectorial de formas. Este formato se puede editar usando software especializado como Inkscape o Adobe Illustrator. Por ejemplo, la figura del logo de la ULPGC se ha obtenido leyendo, con Inkscape, el logo de la Universidad que, entre otros, se suministra en formato SVG. Esta representación vectorial de las formas presenta las siguientes ventajas:

- ✔ En el momento de visualizar la forma se genera una imagen a partir de las curvas de Bézier que se ajusta a la resolución del dispositivo donde vemos la forma. Ello hace que no se pierda calidad al hacer un zoom de la forma. Es decir, podemos hacer un zoom tan grande de la forma como queramos sin que el resultado aparezca pixelado.
- ✔ El formato vectorial ocupa mucho menos espacio que un formato imagen. Por ejemplo, el logo de la ULPGC solo requiere almacenar las coordenadas de 10 puntos (los puntos rojos de la figura) y las coordenadas de los puntos de control de las 11 curvas de Bézier que definen el contorno.
- ✔ Los navegadores web leen y gestionan automáticamente el formato SVG y por tanto, puedo insertar gráficos vectoriales en páginas web sin tener que preocuparse de la resolución de los gráficos. Los usuarios pueden hacer un zoom tan grande como quieran de la página y la imagen, no se pixelarará.
- ✔ La edición del formato vectorial es mucho más sencilla y eficiente que la del formato imagen. Los softwares especializados mencionados permiten editar de forma interactiva las formas vectoriales cambiando la posición de los puntos seleccionados sobre el borde o los puntos de control de las curvas de Bézier.

4.2. La presentación vectorial de formas usando un grafo con círculos de control

En la representación vectorial de formas vista en la sección anterior, los puntos seleccionados sobre el borde y los puntos de control de las curvas de Bézier no están organizados entre sí de ninguna manera y para editar la forma hay que editar cada uno de estos puntos de forma individual. Para obtener una representación vectorial que permita la creación y edición de formas de una manera más compacta, los profesores Luis Alvarez, Agustín Trujillo y Nelson Monzón de la Universidad de Las Palmas de Gran Canariu y el profesor Jean-Michel Morel de la City University of Hong Kong han diseñado una representación de formas vectoriales basada en un grafo de círculos de control (que llamaremos C_k) para representar formas ramificadas. La forma está definida por una colección de círculos de control conectados entre sí mediante un grafo. En la figura 4.2, se ve un ejemplo de tal representación: Las circunferencias en negros representan el contorno de los círculos de control. Los segmentos negros representan las conexiones entre los círculos. Los puntos verdes representan los puntos extremos de las curvas de Bézier que va a generar la forma. Si el círculo solo tiene una conexión, los puntos verdes son los puntos sobre el contorno del círculo en la línea perpendicular al segmento que une el círculo con el otro círculo al que está conectado. Si el círculo está conectado a varios círculos, los puntos verdes están calculados sobre la bisectriz (dibujada con una línea verde discontinua) del ángulo formado por cada par de segmentos que unen el círculo con el resto de círculos a los que está conectado. Para definir totalmente las curvas de Bézier que unen los puntos verdes, imponemos que la tangente de la curva al llegar al punto verde apunte en la dirección del vector perpendicular a la bisectriz (o en la dirección de la tangente al círculo si el círculo solo está conectado a otro círculo). La magnitud del vector tangente se controla usando un parámetro s_k asociado a cada círculo C_k. Por tanto, si $s_k > 0$, se asegura que las curvas de Bézier conecten con suavidad en los puntos verdes. Si por el contrario $s_k = 0$, las curvas de Bézier pueden generar un pico en el punto verde. Nótese que, en esta representación, la curva de Bézier queda determinada al fijar los vectores tangentes con los que llega a sus extremos. A partir de estos vectores tangentes se puede calcular los puntos de control de la curva de Bézier. Dicho de otra manera, una curva de Bézier queda totalmente determinada por el valor de las tangentes en los puntos extremos, y, en general, resulta más intuitivo editar la curva de Bézier usando las tangentes. De hecho, el software Inkscape utiliza las tangentes para editar interactivamente las curvas de Bézier.

Resumiendo, esta representación de formas ramificadas mediante un grafo con círculos de control está determinada por los siguientes parámetros:

- ✔ La colección de círculos de control C_k definidos por las coordenadas de su centro y su radio.
- ✔ El factor de regularidad s_k asociado a cada círculo C_k que determina la suavidad con que conectan las curvas de Bézier que determinan las ramas de la forma.
- ✔ Para cada círculo de control, las conexiones que tiene con otros círculos de control que determinan las ramas del grafo.

Esta representación de una forma es muy compacta, con pocos parámetros que son muy intuitivos. Ello facilita mucho la edición y creación de formas. Respecto a la edición, podemos fácilmente modificar interactivamente los círculos de control, las conexiones entre círculos o el parámetro s_k para controlar la suavidad con que conectan las ramas que llegan al círculo de control. Respecto a la creación, es posible diseñar fácilmente algoritmos sencillos para generar automáticamente grafos de este tipo visualmente atractivos. Para ello, basta usar reglas sencillas sobre cómo distribuir círculos de control, su tamaño, las conexiones entre sí, introducir simetrías y periodicidad, etc. De tal manera, además, que si mantenemos las reglas y volvemos a lanzar la construcción de la forma con una semilla aleatoria distinta, nos saldrán formas del mismo estilo.

Es importante señalar que, para visualizar estos grafos, podemos generar fácilmente un fichero SVG estándar porque el borde de la forma generada viene dada por una colección de curvas de Bézier y semicircunferencias (para completar la forma en los círculos que solo tienen una conexión).

Ilustración 4.2: Ejemplos de figuras generadas

4.3. Historia de las curvas de Bézier

Las curvas de Bézier, concebidas por el ingeniero francés Pierre Bézier en la década de 1960, han dejado una huella significativa en el ámbito gráfico y de diseño. Su introducción se centró inicialmente en el diseño de carrocerías de automóviles, pero su versatilidad y elegancia en la representación de formas suaves han llevado a su adopción en diversas disciplinas.

A nivel gráfico, las curvas de Bézier ofrecen un juego creativo extraordinario. Su capacidad para modelar formas complejas con solo cuatro puntos de control, como se puede ver en la figura 4.3, permite la creación de curvas suaves y fluidas, otorgando a los diseñadores una herramienta potente para dar vida a sus visiones. Esta flexibilidad ha impulsado su uso en campos como la animación, el diseño de fuentes tipográficas, la creación de logotipos y el diseño asistido por computadora.

Sin entrar en la teoría específica de los puntos de control, cabe destacar que las curvas de Bézier constan de un punto inicial, un punto final y dos puntos adicionales que definen la

dirección y la magnitud de la curva. Este enfoque intuitivo ha facilitado su implementación en diversas aplicaciones y ha permitido a los creadores visualizar y ajustar las curvas de manera eficiente.

En resumen, las curvas de Bézier son una herramienta gráfica poderosa que ha enriquecido la expresión visual en numerosos campos creativos, proporcionando a diseñadores y artistas una manera elegante y flexible de dar forma a sus ideas.

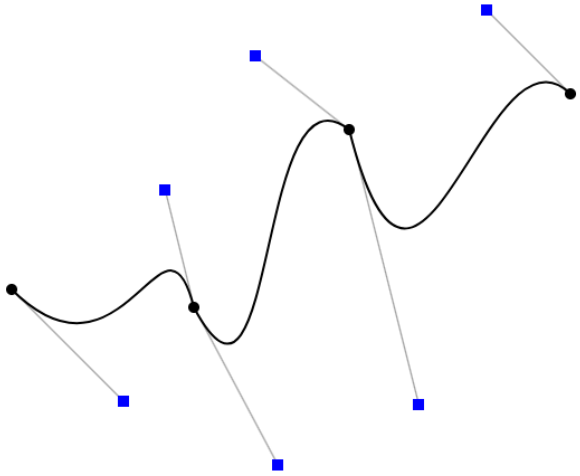


Ilustración 4.3: Curvas de Bézier

4.4. Descripcion de las curvas de Bézier

Una curva de Bézier es una representación matemática utilizada en el diseño de gráficos por computadora y en la modelización de formas suaves y curvas en general.

La idea principal detrás de las curvas de Bézier es describir una curva suave mediante un conjunto de puntos de control. Estos puntos de control determinan la forma y dirección de la curva, y la curva resultante se interpola suavemente entre ellos.

En su forma más común, una curva de Bézier se define mediante tres o más puntos de control. La curva que conecta el primer y último punto de control se llama çurva de Bézier de tercer orden.º çurva de Bézier cúbica". La posición de los puntos de control y sus relaciones afecta la forma y la dirección de la curva resultante.

La representación paramétrica de una curva de Bézier cúbica en 2D se expresa mediante las ecuaciones:

$$B(t) = (1-t)^3 \cdot P_0 + 3 \cdot (1-t)^2 \cdot t \cdot P_1 + 3 \cdot (1-t) \cdot t^2 \cdot P_2 + t^3 \cdot P_3$$

Donde:

- ✔ t es el parámetro que varía de 0 a 1,
- ✔ P_0, P_1, P_2, y P_3 son los puntos de control.

Esta fórmula genera puntos a lo largo de la curva para diferentes valores de t. Al variar t, puedes explorar y visualizar la curva completa. La propiedad importante de las curvas de Bézier es que son suaves y continuas, y su forma está determinada por la posición relativa de los puntos de control.

Estas curvas se utilizan extensivamente en diseño gráfico y modelado por computadora debido a su capacidad para describir formas complejas de manera intuitiva y eficiente. Además de las curvas cúbicas, existen variantes de mayor orden que utilizan más puntos de control y ofrecen mayor flexibilidad en la representación de curvas más elaboradas.

4.5. ¿Por qué son Importantes?

Las curvas de Bézier desempeñan un papel crucial en varias disciplinas, destacando especialmente en diseño gráfico, animación, modelado 3D y representación de formas suaves. Su importancia radica en diversas razones:

1. **Suavidad y Estética:** Las curvas de Bézier permiten la creación de trazados suaves y estéticos. Su capacidad para generar formas naturales las convierte en una herramienta valiosa para diseñadores y artistas.
2. **Diseño Gráfico y Tipografía:** En diseño gráfico, estas curvas se utilizan para dibujar formas precisas, como logotipos e ilustraciones. Además, son fundamentales en la creación de tipografías con detalles finos y control preciso.
3. **Animación Fluida:** En animación, las curvas de Bézier definen trayectorias de movimiento fluidas, permitiendo movimientos realistas y naturales en personajes, objetos y cámaras.
4. **Modelado 3D:** En entornos tridimensionales, estas curvas son esenciales para definir superficies y formas complejas, proporcionando una herramienta eficiente para esculpir y dar forma a objetos en entornos 3D.
5. **Software de Diseño:** Herramientas como Adobe Illustrator y Photoshop aprovechan las curvas de Bézier para la creación y edición precisa de trazados, brindando a los usuarios un control detallado.
6. **Aplicaciones en Ingeniería y CAD:** En campos como ingeniería y diseño asistido por computadora (CAD), estas curvas son vitales para modelar superficies y contornos con precisión.
7. **Interpolación y Suavizado de Datos:** Las curvas de Bézier encuentran aplicaciones en la interpolación y suavizado de datos, representando funciones matemáticas y ofreciendo una visualización clara de patrones en los datos.

8. **Eficiencia Computacional:** Desde el punto de vista computacional, estas curvas son eficientes y permiten evaluaciones rápidas, lo que las hace idóneas para aplicaciones en tiempo real, como gráficos interactivos y videojuegos.

En conclusión, las curvas de Bézier son fundamentales para la creación y manipulación de formas gráficas, ofreciendo versatilidad, precisión y una representación visual atractiva en diversas disciplinas creativas y técnicas.

Capítulo 5

Aportaciones del Proyecto

5.1. Aportación al Entorno Socioeconómico y Social

El proyecto de generación y manipulación de figuras SVG tiene implicaciones significativas en el entorno socioeconómico y social. A través de esta aplicación, se busca influir positivamente en diferentes aspectos, abordando problemáticas específicas y contribuyendo al bienestar general. Las principales aportaciones en este ámbito son:

1. **Acceso Simplificado a Herramientas Gráficas:** La aplicación proporciona un acceso simplificado a herramientas gráficas avanzadas, permitiendo a usuarios con diversos niveles de habilidad crear y manipular figuras SVG de manera intuitiva. Esto democratiza el uso de tecnologías gráficas, brindando a un público más amplio la capacidad de expresarse creativamente en el entorno digital.

2. **Fomento de la Creatividad y la Expresión Visual:** Al facilitar la generación de figuras SVG, la aplicación promueve la creatividad y la expresión visual. Esto tiene un impacto positivo en la sociedad al proporcionar una plataforma accesible para el desarrollo de habilidades artísticas y la creación de contenido gráfico.

3. **Mayor Claridad en la Comprensión de SVG y Curvas de Bézier:** Una de las contribuciones fundamentales de este proyecto radica en su capacidad para ofrecer una comprensión más clara y accesible de las curvas de Bézier y su integración en SVG. La implementación de herramientas interactivas permite a los usuarios visualizar de manera intuitiva cómo las curvas de Bézier influyen en la forma y estructura de los elementos SVG, haciendo que el proceso sea más accesible.

5.2. Aportación Personal

Mi participación activa en el desarrollo de esta aplicación ha llevado consigo una serie de aportaciones personales que han influido en mi crecimiento y desarrollo. Estas contribuciones abarcan tanto el ámbito profesional como el personal, brindándome una experiencia enriquecedora y perspectivas valiosas. Las principales aportaciones personales incluyen:

1. **Desarrollo de Competencias Técnicas:** El proyecto me ha permitido fortalecer mis competencias técnicas, especialmente en el manejo de tecnologías web y la implementación de algoritmos para la generación de curvas de Bézier. La resolución de desafíos técnicos ha sido una oportunidad para el aprendizaje continuo y el perfeccionamiento de habilidades específicas.

2. **Exploración y Aplicación de Conceptos Teóricos:** La implementación de curvas de Bézier en la generación de figuras SVG no solo ha sido un ejercicio técnico, sino también una oportunidad para explorar y aplicar conceptos teóricos en un contexto práctico. Esta experiencia ha enriquecido mi comprensión de los fundamentos matemáticos y su aplicación en proyectos tangibles.

3. **Contribución a un Proyecto Innovador:** Participar en un proyecto innovador de generación de SVG ha sido una experiencia gratificante, contribuyendo a la creación de una herramienta única en su clase. Esta contribución al desarrollo de tecnologías gráficas web ha ampliado mi perspectiva profesional y ha proporcionado una sensación de logro al formar parte de un proyecto distintivo.

4. **Crecimiento en la Resolución de Problemas:** La resolución de desafíos específicos, como la optimización de la representación gráfica y la mejora de la usabilidad, ha sido un catalizador para mi crecimiento en la resolución de problemas. Afrontar y superar obstáculos técnicos ha fortalecido mi capacidad para abordar desafíos complejos en el desarrollo de software.

Capítulo 6

Análisis

En el desarrollo del proyecto, se adoptó una metodología ágil para abordar los desafíos planteados por el profesor Luis Alvarez León. La tarea consistía en transformar un programa existente en C++ que generaba imágenes SVG a través de un archivo de formas en un entorno web más intuitivo y dinámico. El objetivo final era permitir a los usuarios no solo generar imágenes mediante SVG, sino también modificarlas según sus preferencias, añadiendo formas o creando nuevas figuras.

Tras varias reuniones, en las cuales se definieron los requisitos y se priorizaron estos mismos y las funciones necesarias se dividió el proyecto en 3 releases, conformadas en total en 5 sprints contando la validación del código.

Al finalizar cada sprint, se llevó a cabo una revisión exhaustiva con el Product Owner, en este caso, Luis Alvarez León. Durante esta sesión, se presentaron los resultados obtenidos, se discutieron las funcionalidades implementadas y se recopilaron sugerencias y comentarios valiosos. La colaboración estrecha con el Product Owner permitió ajustes continuos para mejorar la satisfacción del usuario y alinear el proyecto con las expectativas.

6.1. Decisión de Diseño: Metodología Ágil y Desglose en Sprints

La elección de una metodología ágil se justifica por la naturaleza iterativa y colaborativa del proyecto. Esta metodología permitió la adaptación continua a los cambios y la entrega incremental de funcionalidades. El proyecto se dividió en sprints, cada uno enfocado en un conjunto específico de tareas para lograr una implementación progresiva.

6.2. Decisiones Técnicas: Uso de WebAssembly, D3.js y Estructura del Proyecto

La decisión de emplear WebAssembly para integrar el código C++ en funciones de JavaScript fue esencial para aprovechar las capacidades del programa original en un entorno web. Además, se optó por utilizar la biblioteca D3.js para la manipulación eficiente de documentos basados en datos en el navegador.

La elección de D3.js se basó en varios beneficios clave que ofrecía para el proyecto:

- ✔ **Manipulación de Documentos Basada en Datos:** D3.js es conocido por su capacidad para enlazar datos a elementos del DOM, facilitando la creación de visualizaciones dinámicas y interactivas.
- ✔ **Facilidad de Integración:** La integración de D3.js con WebAssembly y JavaScript resultó fluida, permitiendo una colaboración efectiva entre las funcionalidades del código C++ y las capacidades de manipulación del DOM proporcionadas por D3.js.
- ✔ **Comunidad Activa y Recursos Abundantes:** D3.js cuenta con una comunidad activa y una abundancia de recursos, lo que facilita la resolución de problemas y la implementación eficiente de funcionalidades avanzadas en la interfaz de usuario.

El Sprint 0 se centró en la configuración del entorno de desarrollo, la implementación de la estructura básica de la aplicación web, la integración de WebAssembly y la incorporación de D3.js para potenciar la manipulación y visualización de datos en el navegador.

6.3. Desglose de Sprints: Primera Release (Sprints 1 y 2)

La primera release del proyecto, compuesta por los sprints 1 y 2, se enfocó en la implementación de funcionalidades fundamentales:

- ✔ Visualización en tiempo real de los cambios dinámicos.
- ✔ Métodos para la creación o eliminación de círculos y segmentos.
- ✔ Desarrollo de funcionalidades para la creación de nuevas figuras, ya sea aleatorias o similares a la actual.
- ✔ Implementación de la capacidad de descargar el SVG o el shape generado y cargarlos.

6.4. Segunda Release (Sprints 3 y 4): Mejoras y Características Avanzadas

La segunda release amplió las capacidades del sistema con sprints 3 y 4:

- ✔ Implementación de una interfaz gráfica de usuario (GUI) dentro del SVG para modificar parámetros de círculos.
- ✔ Mejoras de experiencia del usuario, incluyendo un botón de ayuda con explicaciones detalladas de las funciones.
- ✔ Incorporación de la capacidad de deshacer o rehacer cambios.

6.5. Tercera release (Sprint 5): Cierre y validación del proyecto

El cierre y la validación del proyecto, abordados en el sprint 5, se centraron en aspectos cruciales:

- ✔ Refactorización y optimización del código para mejorar la calidad y el rendimiento.
- ✔ Validación del producto a través de pruebas de rendimiento y corrección de errores.
- ✔ Evaluación exhaustiva de la usabilidad y la experiencia del usuario.
- ✔ Documentación completa del proyecto, incluyendo código y el proceso de implementación.

Este enfoque metodológico garantizó un desarrollo sistemático y una entrega exitosa, cumpliendo con los requisitos iniciales y proporcionando una solución completa y funcional.

Capítulo 7

Metodología Aplicada

La metodología aplicada en este proyecto sigue el enfoque ágil de desarrollo de software, específicamente el marco de trabajo Scrum. Scrum es una metodología iterativa e incremental que permite abordar proyectos complejos de manera flexible y adaptativa. La estructura de Scrum se basa en roles definidos, eventos programados y artefactos clave que facilitan la colaboración y la entrega continua de valor.

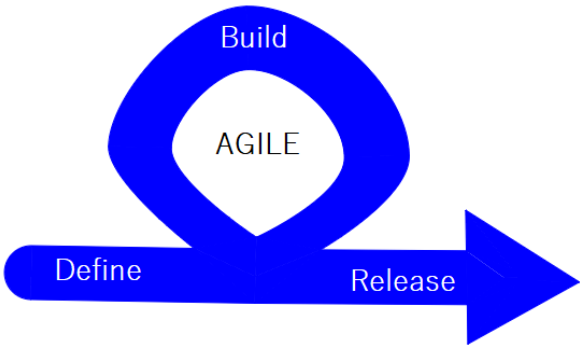


Ilustración 7.1: Ejemplo figura Metodología Ágil

7.1. Roles en Scrum

En el contexto de este proyecto, se definieron los siguientes roles Scrum:

- ✔ **Product Owner (Propietario del Producto):** Luis Alvarez León.
- ✔ **Scrum Master (Facilitador):** Nelson Monzón López.
- ✔ **Equipo de Desarrollo:** Un único miembro, responsable de la implementación del producto.

7.2. Eventos de Scrum

Los eventos de Scrum están diseñados para crear una cadencia y oportunidades de inspección y adaptación. En este proyecto, los eventos adaptados fueron:

- ✔ Sprint Planning (Planificación del Sprint): Reunión al inicio de cada sprint donde se definía el trabajo a realizar. Luis Alvarez León, como Product Owner, definía las tareas prioritarias.
- ✔ **Sprint Review (Revisión del Sprint):** Sesión al final de cada sprint para revisar y demostrar los incrementos de trabajo. Se presentaban los avances y se recopilaba feedback.
- ✔ **Sprint Retrospective (Retrospectiva del Sprint):** Reunión después de la Sprint Review para reflexionar sobre el sprint pasado y realizar ajustes para mejoras futuras.

7.3. Artefactos de Scrum

Los artefactos Scrum utilizados fueron:

- ✔ Product Backlog (Lista de Producto): Luis Alvarez León, como Product Owner, mantenía una lista dinámica de requisitos y prioridades.
- ✔ **Sprint Backlog (Lista del Sprint):** Selección de elementos del Product Backlog comprometidos para el sprint actual.
- ✔ **Incremento:** La suma de los elementos completados durante un sprint, con entregas potencialmente listas.

7.4. Uso de Scrum en el Proyecto

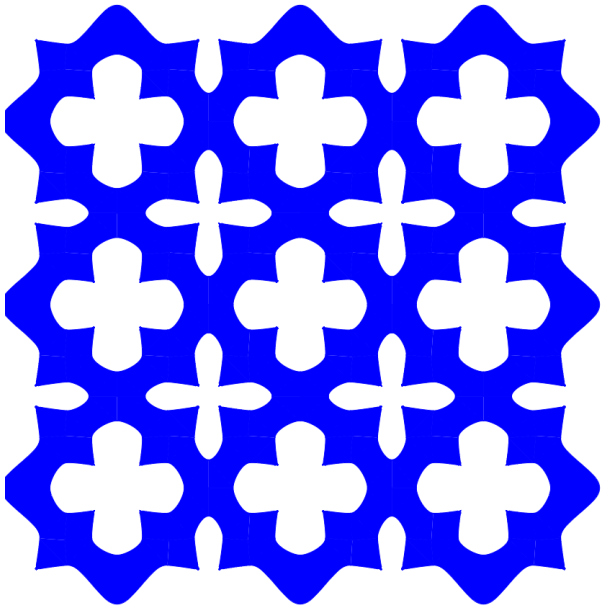
La aplicación de Scrum en este proyecto permitió una gestión ágil del desarrollo, con entregas iterativas y continuas mejoras. Cada sprint, con una duración típica de dos semanas, se iniciaba con una Sprint Planning donde se seleccionaban y priorizaban las tareas. Durante el sprint, como único miembro del equipo, trabajabas en la implementación planificada.

La Sprint Review proporcionaba una oportunidad para mostrar los resultados a Luis Alvarez León y recibir feedback. La Sprint Retrospective permitía reflexionar sobre el proceso y realizar ajustes para mejorar en los sprints futuros.

Este enfoque ágil y adaptativo garantizó una colaboración efectiva y una entrega continua de valor durante el desarrollo del proyecto.

Capítulo 8

Implementación



8.1. Sprint 0: Configuración Inicial

Inicialmente, disponemos de un archivo shape que define la configuración inicial de nuestra figura. Este archivo contiene los primeros $(N \times 3) + 1$ valores, donde N es el número de nodos en nuestra figura. Por ejemplo:

```
4
516.034241 749.759521
516.034241 150.000000
754.762512 600.000000
269.237518 584.855225
150.000000
100.000000
```

```
100.000000
100.000000
1.000000
1.000000
1.000000
1.000000
```

En este ejemplo, el primer dato indica el número de círculos que conforman la figura. Los siguientes N valores representan las coordenadas X e Y de cada uno de los círculos. Los N valores siguientes corresponden al radio de cada círculo, mientras que los últimos N valores indican el factor de suavizado, que influye en los segmentos que conectan cada círculo con los demás.

Este formato proporciona la información necesaria para entender la disposición inicial de los nodos, sus ubicaciones y propiedades específicas, como el radio y el factor de suavizado.

8.2. Configuración del Entorno de Desarrollo

La fase inicial del proyecto se centró en la configuración del entorno de desarrollo utilizando WebStorm, una poderosa herramienta que proporciona un entorno integrado para el desarrollo web. Durante este proceso, se creó una página básica en HTML, junto con su respectivo estilo en CSS y funcionalidades en JavaScript.

Creación de la Página Básica en HTML: Se diseñó una estructura HTML sencilla que sirviera como el punto de partida para la aplicación. La página incluye un simple `<div>` destinado a alojar un elemento `canvas` y un `<textarea>` donde se puede cargar un shape por defecto. Esta disposición básica proporciona el lienzo para la visualización de la figura generada y un área para la interacción del usuario.

Estilo en CSS: Se implementó un archivo de estilo CSS (`style.css`) para mejorar la presentación y disposición de los elementos en la página. Este archivo se encarga de definir estilos para el `<div>` contenedor, el `<textarea>`, y otros elementos según sea necesario.

Funcionalidades en JavaScript: El archivo `tools.js` se encargó de gestionar las interacciones y funcionalidades en la página. Esto incluyó la manipulación dinámica del `canvas` en respuesta a las acciones del usuario y la carga inicial de un shape por defecto en el `textarea`.

```
window.addEventListener("load", function () {
    // Crear una nueva solicitud XMLHttpRequest
    let solicitud = new XMLHttpRequest();

    // Configurar la solicitud
    solicitud.open("GET", "./shape.txt", true);
    solicitud.onreadystatechange = function () {
        if (solicitud.readyState === 4 && solicitud.status === 200) {
            // Obtener el contenido del archivo
```

```
            let contenido = solicitud.responseText;

            // Establecer el contenido en el textarea
            let textarea = document.getElementById("shapeInput");
            textarea.value = contenido;
        }
    };

    // Enviar la solicitud
    solicitud.send();
});
```

Esta configuración inicial proporciona una base sólida para el desarrollo posterior de la aplicación, permitiendo la interacción con el usuario y la visualización de las formas generadas en el canvas.

8.2.1. Integración de WebAssembly

La configuración del entorno de desarrollo es un paso crucial en el desarrollo de proyectos que involucran WebAssembly. En este contexto, se ha creado un script automatizado para simplificar este proceso.

```
#!/bin/bash
# Ejecutar emsdk_env.sh en la ruta de emsdk
cd /ruta/a/tu/emsdk
source emsdk_env.sh

# Ejecutar emcc en la ruta de BranchedShapesEdition
cd /ruta/a/tu/proyecto/BranchedShapesEdition
emcc mainBranchedShapesEdition.cpp triangulation.cpp -o blackbox.js --embed-file shape.txt
-s NO_EXIT_RUNTIME=1 -s "EXPORTED_RUNTIME_METHODS=['ccall',␣'Pointer_stringify',␣'_free']"
--shell-file index.html -s ALLOW_MEMORY_GROWTH=1

echo El script ha finalizado.
```

Algoritmo 8.1: Script de Configuración con WebAssembly

A todo esto cabe recalcar para su entendimiento posterior, que es Emscripten. Emscripten es una herramienta que permite la compilación de código fuente en lenguajes como C y C++ a JavaScript y WebAssembly, facilitando así la ejecución de software escrito en estos lenguajes en entornos web. Esta capacidad de convertir código nativo en código web amplía las posibilidades de desarrollo y ejecución de aplicaciones en el navegador, brindando a los desarrolladores la flexibilidad de utilizar lenguajes de programación de bajo nivel en el entorno web. Esta herramienta es la que estamos utilizando al usar el comando ”emcc”.

En el Listado 8.1, se destaca el script que automatiza la configuración del entorno y la compilación del proyecto con WebAssembly. Asegúrate de ajustar las rutas según la estructura específica de tus directorios y archivos. Este script simplifica el proceso de desarrollo, proporcionando una base sólida para proyectos que utilizan WebAssembly. El script realiza las siguientes tareas fundamentales:

1. **Configuración del Entorno Emscripten:** Inicia ejecutando el script `emsdk_env.sh` del Emscripten SDK. Este paso garantiza la correcta configuración del entorno de desarrollo y establece las variables necesarias para utilizar Emscripten.
2. **Compilación con Emscripten:** Posteriormente, el script se traslada al directorio del proyecto (**BranchedShapesEdition**) y utiliza el compilador Emscripten (**emcc**). La compilación implica la traducción del código fuente C++ de los ficheros **mainBranchedShapesEdition.cpp** y **triangulation.cpp** a WebAssembly y la generación de un archivo JavaScript (**blackbox.js**) y un archivo WASM (**blackbox.wasm**).
3. **Configuración de Opciones de Compilación:** Varias opciones de compilación se establecen para optimizar el rendimiento y la interoperabilidad entre el código C++ y JavaScript. Estas opciones incluyen la supresión de la salida del tiempo de ejecución de Emscripten al finalizar (`NO_EXIT_RUNTIME=1`), la definición de métodos de tiempo de ejecución exportados (`EXPORTED_RUNTIME_METHODS`), y la permitir el crecimiento dinámico de la memoria (`ALLOW_MEMORY_GROWTH`).

Este script automatizado optimiza el flujo de trabajo al eliminar la necesidad de configurar manualmente el entorno y compilar el código repetidamente. Facilita el desarrollo continuo al actualizar automáticamente los archivos JavaScript y WebAssembly en respuesta a cambios en el código fuente C++.

Respecto a los ficheros resultantes, no se deberá hacer nada con ellos, simplemente son necesarios para luego poder llamar a funciones del codigo C++, simplemente deberemos cargarlo en nuestro index.html.

8.2.2. Funcionamiento del entorno web

En esta fase, se implementaron las primeras funciones esenciales para que el entorno web funcione de manera efectiva. A continuación, se proporciona una descripción de las funciones clave:

✔ **computeShape:**

- **Propósito:** Calcula la forma llamando a un módulo C++.
- **Acciones:**
  1. Utiliza el módulo C++ mediante la función `_Z19ComputeSVGFromShapePc`.
  2. Actualiza el contenido en el área de salida SVG (`svgOutput`).
- **Resultado:** Facilita la generación dinámica de formas basadas en el input proporcionado en el área de entrada (`shapeInput`).

✔ **drawFigure:**

- **Propósito:** Parsea y dibuja el contenido SVG en el lienzo.
- **Acciones:**

1. Obtiene el contenido SVG desde el área de salida (`svgOutput`).
2. Parsea el contenido utilizando un `DOMParser`.
3. Dibuja el contenido SVG en el lienzo, reemplazando cualquier contenido previo si es necesario.
4. Resalta ciertos círculos en el lienzo.
5. Agrega eventos y funcionalidades adicionales al lienzo.

- **Resultado:** Permite la visualización dinámica de las formas generadas en el lienzo, con interactividad en ciertos elementos.

✔ **Carga de Contenido Inicial:**

- **Propósito:** Carga el contenido inicial desde un archivo `shape.txt`.
- **Acciones:**
  1. Crea una solicitud `XMLHttpRequest`.
  2. Configura la solicitud para obtener el contenido del archivo `shape.txt`.
  3. En el evento `onload`, actualiza el contenido en el área de entrada (`shapeInput`).
- **Resultado:** Facilita la carga automática de contenido inicial al cargar la página web.

Estas funciones forman la base del entorno web, permitiendo la generación, visualización y manipulación interactiva de formas SVG.

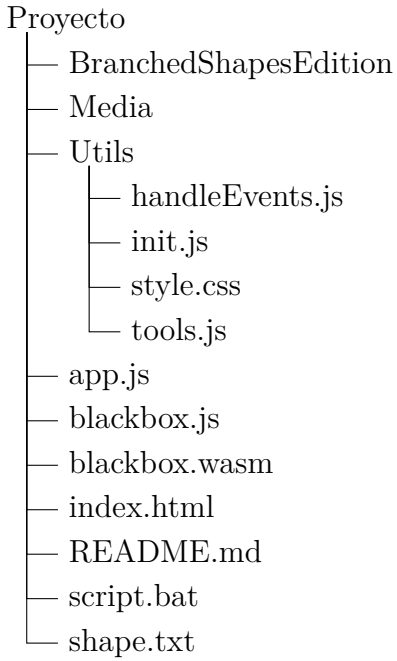


Ilustración 8.1: Estructura de archivos y carpetas

Como se puede visualizar en la Figura 8.1, tenemos la siguiente estructura. El directorio `BranchedShapesEdition` contiene todo el código C++, `Media` almacena las imágenes, `Utils` contiene varios archivos JavaScript (Uno para eventos, otro para la inicialización, un archivo CSS y otro con funcionalidades). En el directorio raíz, encontramos `app.js`, que es el archivo JavaScript encargado de llamar a las funciones de C++, los archivos `blackbox` que son la salida del WebAssembly, nuestro `index`, el shape de ejemplo y el script para compilar con WebAssembly, junto con un `README` con información del programa.

Para que la página web funcione, no son necesarios ni la carpeta `BranchedShapesEdition`, ni el `README`, ni el archivo Script.

8.3. Sprint 1 y 2: Primera Release

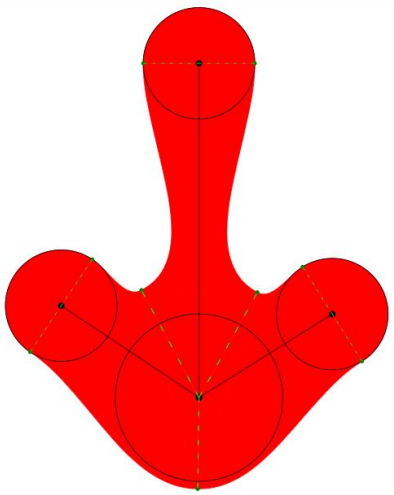


Ilustración 8.2: Svg de ejemplo

Tras nuestro Sprint 0, el resultado que obtenemos es el que podemos vislumbrar en la figura 8.2, una mera imagen, por lo que en este sprint los esfuerzos se han visto centrados en añadir funcionalidades y en poder modificar el SVG en tiempo real.

✔ **Implementación de D3.js:**

- **Propósito:** Habilita la modificación en tiempo real e interacción con los círculos en el elemento SVG mediante D3.js.
- **Acciones:**
  1. Selecciona todos los círculos interactivos en el lienzo con la línea de código: `circulitos = d3.selectAll("#canvas_svg circle.draggable-circle");`.
  2. Implementa tres funciones clave (`dragStarted`, `dragged`, y `dragEnd`) para la interacción en tiempo real con los círculos.

- **Resultado:** Permite la manipulación dinámica de los círculos y la actualización en tiempo real de la posición de los nodos.

✔ **Funciones de Arrastre (`dragStarted`, `dragged`, `dragEnd`):**

- **Propósito:** Facilitan la selección y arrastre en tiempo real de los círculos dentro del lienzo.
- **Acciones:**
  1. `dragStarted`: Inicia el arrastre al hacer clic en un círculo, marcando el círculo activo.
  2. `dragged`: Actualiza la posición del círculo mientras se arrastra, permitiendo la interactividad en tiempo real.
  3. `dragEnd`: Finaliza el arrastre al soltar el mouse, actualiza la figura y realiza acciones de limpieza.
- **Resultado:** Permite la modificación dinámica de la posición de los círculos y la actualización automática de la figura en el lienzo.

✔ **Función de Transformación (`transform`):**

- **Propósito:** Permite la modificación del shape mediante código C++.
- **Acciones:**
  1. Utiliza la función C++ `transform` para ajustar el zoom, desplazamiento y otras transformaciones.
  2. Actualiza el contenido en el área de entrada (`shapeInput`) con el resultado de la transformación.
- **Resultado:** Facilita la manipulación del shape mediante acciones específicas de transformación.

8.3.1. Creación y eliminación de círculos/segmentos

En esta sección, se detalla la interacción entre el código C++ y el entorno web, mostrando con mayor claridad cómo se gestionan la creación y eliminación de círculos y segmentos.

✔ **Función de Inserción de Círculo en C++ (`insert_point`):**

- **Propósito:** Inserta un nuevo círculo en la representación interna de la forma.
- **Acciones:**
  1. Calcula el radio promedio y el factor de suavizado si no se proporcionan.
  2. Agrega la posición (`point2d`), radio y factor de suavizado a las listas internas (`n_`, `r_`, `s_`).

- **Resultado:** Añade un nuevo círculo a la representación interna de la forma con la posición, radio y suavizado especificados.

✔ **Función de Inserción de Círculo Exportada a JavaScript (`insertpoint`):**

- **Propósito:** Interconecta la inserción de un círculo desde JavaScript a través de WebAssembly con la función C++.
- **Acciones:**
  1. Llama a la función C++ `insert_point` con las coordenadas proporcionadas.
  2. Genera un nuevo archivo SVG (`shape_new.svg`) y lo actualiza en el área de salida.
- **Resultado:** Permite la inserción de un nuevo círculo en la forma desde JavaScript y actualiza la visualización.

✔ **Función de Eliminación de Círculo en JavaScript (`handleCircleClickForErase`):**

- **Propósito:** Maneja la eliminación de un círculo en respuesta a la interacción del usuario.
- **Acciones:**
  1. Identifica el índice del círculo seleccionado.
  2. Llama a la función C++ `erasePoint` con el índice del círculo a eliminar.
  3. Recalcula la forma y actualiza la visualización.
- **Resultado:** Permite la eliminación de un círculo seleccionado y actualiza la representación visual.

8.3.2. Mejoras en la Modificación de Figuras

En los hitos anteriores, se logró desarrollar un software capaz de realizar modificaciones básicas en la figura, permitiendo la manipulación de círculos, transformación de la figura, así como la adición o eliminación de círculos y segmentos.

Esto proporciona la capacidad de crear figuras más complejas, como se explorará en el siguiente apartado, donde se generan figuras de manera aleatoria.

La versatilidad actual de tener círculos de diversos tamaños y conexiones ofrece la transición desde la figura inicial (ver Figura 8.2) hacia composiciones más elaboradas, como se muestra en la Figura 8.3, aprovechando las funcionalidades previamente implementadas, por ejemplo, aquí tras añadir varios círculos, todos unidos por segmentos a un circulo inicial, se ha podido crear una especie de estrella.

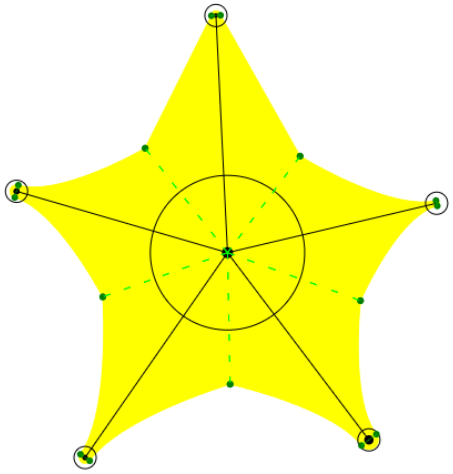


Ilustración 8.3: Estrella

8.3.3. Generación de figuras

El siguiente paso en el desarrollo del proyecto fue la implementación de la generación de nuevas figuras, tanto de manera aleatoria como a partir de la figura existente. Para ello, se crearon las siguientes funciones en JavaScript:

En la figura 8.4, se puede apreciar una figura generada a través de la función aleatoria, donde muchos de los valores del *shape* se han puesto al azar. Esto permite mostrar formas más o menos complejas, desde figuras creadas por dos círculos hasta más de un centenar.

Ilustración 8.4: Figura aleatoria

En la figura 8.5, se muestra una figura generada con parámetros muy similares a los de la figura aleatoria. Aunque se ha introducido cierta aleatoriedad en la imagen, se mantiene la esencia de la figura original, tanto en el número de círculos como en sus conexiones.

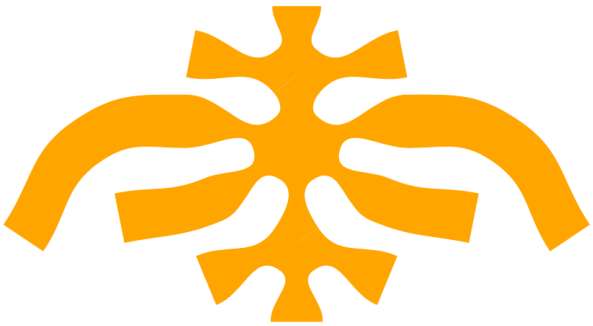


Ilustración 8.5: Figura similar

Ambas funciones utilizan el módulo `Free` para liberar la memoria del SVG anterior antes de generar uno nuevo. Luego, modifican el valor de `shapeInput.value` con la salida del módulo, que representa la figura generada mediante los métodos aleatorio o similar en C++. Posteriormente, se vuelve a calcular el shape para generar la nueva imagen. Cabe destacar que se utiliza la función `fixit()` para corregir posibles errores de índices de círculos en casos de SVGs muy grandes, llamando a la función `transform`, sin realizar cambios significativos. Se optó por esta solución debido a su bajo costo computacional, ya que el proceso de regenerar la imagen por segunda vez es insignificante en comparación con la frecuencia de cálculo al mover nodos.

8.3.4. Carga y descarga del SVG/Shape

Para cargar y descargar el shape, lo cual nos sirve para poder guardar nuestro avance en una figura o simplemente una figura que nos haya gustado que se generará aleatoriamente, podemos guardar el campo de texto dentro del textarea, es decir el shape, lo que vienen a ser los datos que generan la figura en un formato .txt, a si mismo podemos cargar un fichero .txt que contenga un shape y pasar los datos al textarea para generar la figura.

Luego tenemos la siguiente función:

```
function downloadsvg() {
    var aux = Module.ccall("_Z11downloadsvgv", "string", ["number"], []);

    var enlaceDescarga = document.createElement("a");
    enlaceDescarga.setAttribute(
        "href",
        "data:image/svg+xml;charset=utf-8," + encodeURIComponent(aux),
    );
    enlaceDescarga.setAttribute("download", "shape.svg");
    enlaceDescarga.style.display = "none";
```

```
    document.body.appendChild(enlaceDescarga);
    enlaceDescarga.click();
    document.body.removeChild(enlaceDescarga);
}
```

Explicación:

La función `downloadsvg()` tiene como objetivo descargar el SVG generado en la página web como un archivo con extensión `.svg`. Veamos una explicación detallada de cada paso en la función:

1. **Llamada a la Función C++**: La función comienza con la llamada a una función en C++ a través de WebAssembly utilizando `Module.ccall`. La función de C++ que se llama es `_Z11downloadsvgv`.
2. **Creación del Enlace de Descarga**: Se crea un elemento `<a>` (hipervínculo) en el documento. Este elemento se utiliza para simular el comportamiento de hacer clic en un enlace y descargar un archivo.
3. **Configuración del Enlace de Descarga**:
   - ✔ Se establece el atributo `href` del enlace con una cadena que representa los datos SVG. Esto se logra mediante la codificación URI del SVG generado.
   - ✔ Se establece el atributo `download` con el nombre del archivo que se descargará, en este caso, ”shape.svg”.
4. **Manejo de la Descarga**:
   - ✔ Se establece el estilo del enlace para que no sea visible en la página (`style.display = "none"`).
   - ✔ Se agrega el enlace al cuerpo del documento.
   - ✔ Se simula el clic en el enlace utilizando `enlaceDescarga.click()`.
   - ✔ Se elimina el enlace del cuerpo del documento después de la descarga.

En resumen, la función descarga el SVG generado en la página web como un archivo SVG llamado ”shape.svg”. La implementación incluye manipulaciones del DOM (Document Object Model) para lograr la descarga simulada del archivo.

8.4. Sprint 3 y 4: Segunda Release

En esta fase, tras culminar la primera versión del proyecto, hemos logrado alcanzar una versión funcional con características mínimas viables. Como resultado de las iteraciones y ajustes anteriores, la página web ha adquirido la capacidad de interactuar de manera intuitiva con la figura generada. Los usuarios tienen la capacidad de desplazar la figura y realizar zoom de manera eficaz. Además, la plataforma permite la manipulación en tiempo real de la

configuración de la figura, mediante el desplazamiento de los nodos y la adaptación simultánea de los segmentos correspondientes.

Esta implementación no solo facilita la modificación de la geometría existente, sino que también proporciona la funcionalidad de añadir o eliminar nodos, otorgando flexibilidad y versatilidad al usuario. De manera complementaria, se ha incorporado la capacidad de generar figuras completamente nuevas, brindando a los usuarios la posibilidad de explorar diferentes configuraciones geométricas de manera dinámica. Este hito representa un paso significativo hacia el desarrollo de una plataforma interactiva y robusta, sentando las bases para futuras mejoras y ampliaciones en el marco de este proyecto de investigación.

8.4.1. Modificación de parámetros de los círculos

El primer cambio a implementar para la segunda release fue la opción de poder cambiar parámetros tanto de un círculo como de varios al mismo tiempo. Comenzando por los cambios individuales a un círculo, tenemos:

✔ **Función de Mostrar Popup (`showPopup`):**

- **Propósito:** Muestra un cuadro emergente con la información actual del círculo seleccionado.
- **Acciones:**
  1. Obtiene la información actual del círculo seleccionado, como su radio, posición (`X` y `Y`), y factor de suavizado.
  2. Utiliza el contenido del área de entrada (`shapeInput`) y el índice del círculo para extraer la información relevante.
- **Resultado:** Prepara la información actual del círculo seleccionado para su visualización en un cuadro emergente.

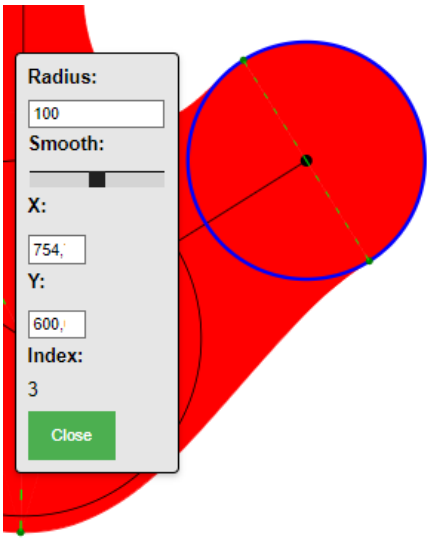


Ilustración 8.6: PopUp individual

Tras la implementación del popup, se agregó la capacidad de modificar no solo el radio y el factor de suavizado de círculos individuales, sino también de todos los círculos simultáneamente. El popup se activa al hacer clic derecho fuera de la figura pero dentro del lienzo (canvas). Este popup permite no solo ajustar parámetros globales, sino también rotar la imagen. De manera predeterminada, la imagen rota alrededor del centro del lienzo, pero se pueden agregar las coordenadas X e Y deseadas como el nuevo eje de rotación.

Como se puede ver en la figura 8.7, el popup nos permite modificar todos estos parámetros. También se puede apreciar cómo la imagen ha sido rotada; el slider va desde los 0 hasta los 360 grados.

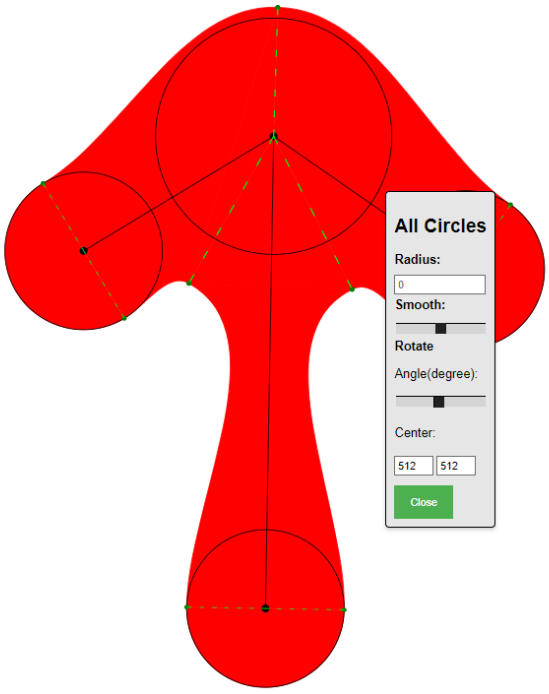


Ilustración 8.7: PopUp Global

A continuación se muestra un ejemplo de como simplemente modificando el smooth global, podemos tener figuras totalmente diferentes, teniendo en cuenta que también se pueden modificar parámetros individuales de cada circulo. En las siguientes figuras podemos vislumbrar esta diferencia, teniendo de base, los mismos círculos, en las mismas posiciones y con el resto de parámetros iguales.

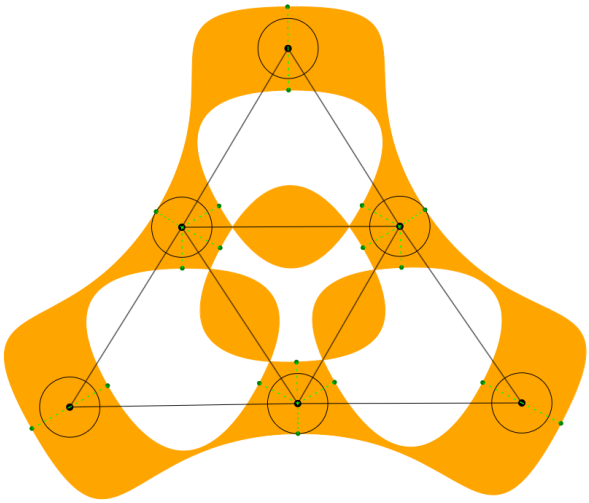


Ilustración 8.8: Smooth al máximo

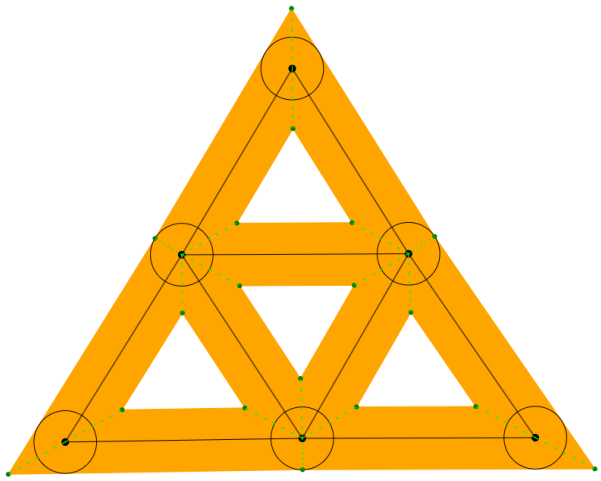


Ilustración 8.9: Smooth al minimo

8.4.2. Mejoras de usuario

Se incorporaron los botones 'Help' y 'Advance Info' para mejorar la interfaz de usuario y proporcionar información detallada y funcionalidades adicionales.

El botón 'Help' despliega información sobre todos los botones de la página, facilitando la comprensión de su propósito y funcionalidad. Mientras tanto, el botón 'Advance Info' revela un área de texto (textarea) donde se puede modificar manualmente la forma (shape). Además, se incluyeron los botones 'Compute Shape', 'Export Shape' y 'Upload Shape' en esta área para consolidar funciones relacionadas y ofrecer una interfaz más limpia.

En la figura 8.10, se muestra la distribución de estos elementos y la descripción de ayuda para una mejor comprensión del usuario.

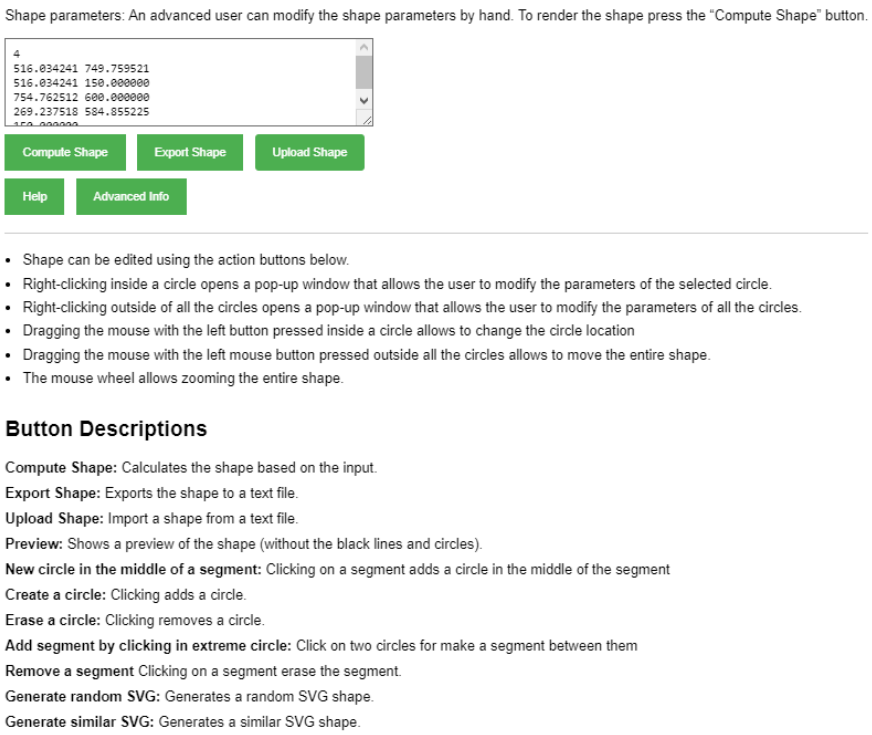


Ilustración 8.10: Ayuda (Help)

Se introdujeron dos nuevas funcionalidades derivadas de las ya existentes: "Añadir Múltiples Círculos" y "Eliminar Múltiples Círculos". Estas mejoras contribuyeron a dar más facilidad al usuario, ya que al utilizar estos botones, el evento no se elimina tras añadir o quitar un solo círculo. En su lugar, el usuario puede agregar o eliminar varios círculos de una vez, sin la necesidad de mantener presionado el botón continuamente.

La implementación de estas funciones llevó a la inclusión de un botón adicional para descartar las acciones pendientes. Este botón es crucial cuando se decide no añadir o eliminar más círculos, o si se cambia de opinión y se desea anular la acción previamente seleccionada.

Estas mejoras aumentan la eficiencia y la comodidad en la manipulación de la figura, proporcionando una experiencia más fluida y amigable para el usuario.

8.4.3. Mejoras en la Interactividad: Añadir y Eliminar Múltiples Círculos

La implementación de estas funciones llevó a la inclusión de un botón adicional para descartar las acciones pendientes. Este botón es crucial cuando se decide no añadir o eliminar más círculos, o si se cambia de opinión y se desea anular la acción previamente seleccionada.

Estas mejoras aumentan la eficiencia y la comodidad en la manipulación de la figura, proporcionando una experiencia más fluida y amigable para el usuario.

8.4.4. Deshacer o rehacer cambios

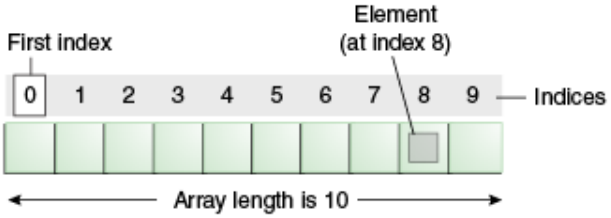


Ilustración 8.11: Ejemplo de array

Como se muestra en la figura 8.11, se dispone de un array con 10 espacios para almacenar formas (shapes), que representan las figuras en la aplicación. Cada vez que se realiza un cambio, como mover figuras (aunque solo se guarda al soltar el ratón), se almacena la forma actual. Si el array está lleno, se realiza un desplazamiento hacia la izquierda, eliminando la forma más antigua (en el primer índice) para dejar espacio al nuevo cambio.

Ilustración 8.12: Gestión de Formas (Undo/Redo)

En la figura 8.12, se presentan los botones 'Deshacer' y 'Rehacer'. Al hacer clic en la flecha hacia atrás (Deshacer), el puntero que apunta a la última posición del array se mueve a la penúltima, mostrando el penúltimo SVG guardado en el lienzo. De esta manera, se puede deshacer hasta llegar al índice 0, es decir, retroceder 9 cambios. Por otro lado, la flecha hacia adelante (Rehacer) simplemente avanza el puntero hacia adelante.

Cuando nos encontramos en un índice menor al máximo existente y realizamos modificaciones, se eliminan las formas restantes del array por encima de la posición actual. Por

ejemplo, si estamos en la posición 7, se borrarán los índices 8 y 9, y la última modificación se convertirá en la posición 8. Esto sucede porque al retroceder y realizar un cambio, se deshacen todos los cambios posteriores a ese punto.

8.4.5. Interacción con Segmentos

Se incorporaron varias funciones para interactuar con los segmentos, realizando operaciones principalmente en código C++. En JavaScript, la función simplemente envía los índices de los círculos involucrados.

Añadir un Círculo en el Medio de un Segmento Esta función agrega un círculo en el centro de un segmento, de manera unida, es decir, el círculo está conectado por segmentos a los dos círculos que conformaban el segmento.

```
EMSCRIPTEN_KEEPALIVE
char* insertpointmiddle(double x, double y) {
    int i = 0;
    int j = 0;
    nT.segment_distance(x, y, i, j);
    nT.insert_point((nT.n_[i] + nT.n_[j]) * 0.5,
                    (nT.r_[i] + nT.r_[j]) * 0.5,
                    (nT.s_[i] + nT.s_[j]) * 0.5, i, j);
    nT.svg_generation("shape_new.svg", true, globalColor);
    return allocateAndExtract();
}
```

Eliminar un Segmento Esta función elimina un segmento al hacer clic en él.

```
EMSCRIPTEN_KEEPALIVE
char* disconnectnodes(double x, double y) {
    int i = 0;
    int j = 0;
    nT.segment_distance(x, y, i, j);
    nT.disconnect_nodes(i, j);
    nT.svg_generation("shape_new.svg", true, globalColor);
    return allocateAndExtract();
}
```

Añadir un Segmento Conectando Dos Círculos Diferentes Esta función une dos círculos mediante un segmento al hacer clic en dos círculos diferentes.

```
EMSCRIPTEN_KEEPALIVE
char* connectnodes(int argc, int argv) {
```

```
    nT.connect_nodes(argc, argv);
    nT.svg_generation("shape_new.svg", true, globalColor);
    return allocateAndExtract();
}
```

8.5. Sprint 5: Cierre y Validación

8.5.1. Refactorización y Optimización

Luego de alcanzar la última versión, se llevó a cabo un proceso de refactorización del código con el objetivo de mejorar su legibilidad, facilitando así el mantenimiento y la escalabilidad del mismo.

Entre las acciones realizadas se encuentran:

- ✔ Cambio de Nombres de Variables: Se realizaron ajustes en los nombres de las variables para hacer el código más comprensible y coherente.
- ✔ Reorganización del Código: Se trasladaron secciones de código a diferentes archivos para una mejor organización y estructuración del proyecto.
- ✔ Optimización de Cargas de JavaScript: Se realizaron mejoras en la eficiencia de la carga del código JavaScript, buscando un rendimiento más óptimo.
- ✔ Eliminación de Llamadas Innecesarias: Se identificaron y eliminaron llamadas a funciones en momentos no críticos, contribuyendo a la eficiencia general del código.
- ✔ Mejora de la Función de Animación: La función de animación fue optimizada para lograr un movimiento más suave y fluido en la interfaz.

Estas acciones no solo contribuyen a la legibilidad del código, sino que también buscan mejorar el rendimiento y la mantenibilidad del sistema, sentando las bases para futuras actualizaciones y expansiones.

8.5.2. Pruebas de Rendimiento, Validación y Corrección de Errores

Se llevaron a cabo pruebas exhaustivas de funcionamiento y rendimiento en diversos dispositivos y navegadores para identificar posibles errores y optimizar el rendimiento de la página web.

Ilustración 8.13: Error ”stackSave”

Durante estas pruebas, se detectó un error particular, representado en la figura 8.13, denominado ”stackSave”, que surgía al llamar a métodos de ”blackbox.js” antes de que estuviera completamente cargado, especialmente al invocar métodos de WebAssembly. Para abordar este problema, se implementó un pequeño retardo en la llamada a los primeros métodos al inicio de la página, permitiendo que ”blackbox.js” se cargara completamente.

Además, se ajustó la asignación de memoria de WebAssembly, aumentando la memoria mínima y permitiendo una expansión automática solo cuando fuera necesario, como en el caso de figuras muy grandes.

Durante las pruebas, se observó que interactuar con círculos de radio muy pequeño resultaba complicado. Para mejorar la usabilidad, se implementó la visualización de estos círculos con un radio mínimo de 5 para facilitar la interacción del usuario, mostrándose visualmente con un círculo negro.

Se introdujo el método ”preview”, que permite visualizar la imagen final sin las líneas negras ni círculos oscuros, como se presenta en la figura 8.14. Esta funcionalidad ayuda a los usuarios a obtener una vista previa de la imagen tal como se exportaría.

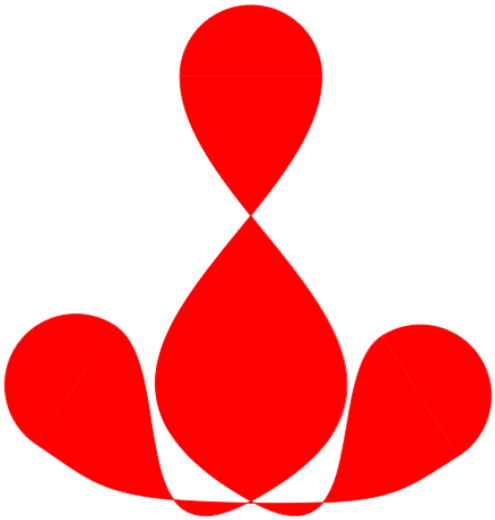


Ilustración 8.14: Vista previa (Preview)

Se llevaron a cabo correcciones de errores menores y se implementaron cambios para mejorar la experiencia del usuario. Algunas de las modificaciones realizadas incluyen:

- ✔ Reiniciar valores al valor por defecto al cerrar los popups.
- ✔ Mejora en la precisión de los clics y al arrastrar el ratón para una interacción más precisa.
- ✔ Permite cerrar los popups al hacer clic fuera de ellos, mejorando la usabilidad.
- ✔ Mejoras en el apartado de ayuda para una comprensión más clara por parte del usuario.
- ✔ Ahora, la imagen descargada tiene un nombre predefinido para una mejor organización.
- ✔ Solución a errores al superponer círculos, evitando problemas visuales y de interacción.
- ✔ Eliminación de errores que podrían causar el bloqueo o cierre inesperado de la página.

Estas modificaciones contribuyen a una mayor estabilidad, precisión y facilidad de uso en la aplicación web.

8.5.3. Evaluación de la Usabilidad y Experiencia del Usuario

Con el fin de evaluar la usabilidad y recabar comentarios sobre la experiencia del usuario, se llevaron a cabo pruebas con personas sin conocimientos previos de la página web. Estas pruebas se centraron en evaluar la facilidad de uso y comprensión de las funcionalidades disponibles.

Los participantes proporcionaron valiosos comentarios sobre la navegación, la disposición de los elementos y la comprensión de las funciones. A raíz de estas evaluaciones, se identificaron áreas de mejora en la interfaz gráfica. Para abordar estas mejoras, se decidió realizar cambios significativos.

Una de las mejoras destacadas fue la reorganización de los botones. En lugar de tener botones dispersos en la interfaz, se optó por agruparlos en una barra flotante superpuesta sobre el lienzo. Los botones se organizaron por funcionalidades, y al hacer clic en una categoría específica, se revelaron los botones correspondientes. Esta modificación proporcionó una interfaz más amigable, cómoda e intuitiva, simplificando el acceso a las diversas funciones disponibles.

Este proceso de evaluación y mejora continuará, permitiendo la evolución constante de la interfaz para adaptarse a las necesidades y expectativas de los usuarios, asegurando así una experiencia óptima en futuras versiones.

8.5.4. Documentación y Proceso de Implementación

La documentación del proyecto se encuentra de una manera breve en el repositorio oficial en GitHub: NestorManeiro/TFG. En el archivo `README.md`. Además, se destaca que la página web en funcionamiento puede ser visitada en https://nestormaneiro.github.io/TFG/.

Para una comprensión más profunda de los avances y detalles del proyecto, se ha mantenido un documento en Google Docs: Documentación detallada del TFG. Este documento proporciona información adicional sobre el desarrollo, los desafíos enfrentados y los avances realizados durante el proceso.

La implementación y desarrollo del proyecto han seguido un enfoque iterativo, con avances continuos y mejoras graduales. Se han utilizado las funcionalidades de GitHub para gestionar y organizar el código, el seguimiento de cambios a lo largo del tiempo y facilitando gracias a GitHub Pages la comprobación del código en diferentes dispositivos.

8.6. Sprint 6: Evolución Continua

Después de completar todas las fases requeridas para el trabajo de investigación, junto con su validación correspondiente, se determinó integrar algunas de las mejoras obtenidas a través de las pruebas llevadas a cabo con diversos usuarios, de cara al TFG. Estas mejoras se centran principalmente en la optimización de la interfaz, la implementación de una versión móvil y la realización de ajustes menores.

8.6.1. Mejoras graficas

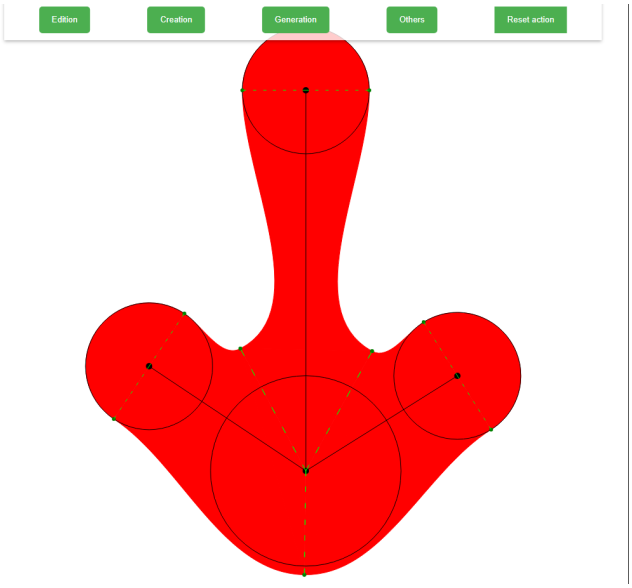


Ilustración 8.15: Barra flotante

Como se evidencia en la Figura 8.15, se incorporó una barra flotante superpuesta a la imagen con el propósito de mejorar la comodidad y proporcionar una interfaz más limpia. En esta barra, se organizaron los botones en cuatro categorías. ('Edición', 'Creación', 'Generación' y 'Otros'). Al hacer clic en uno de estos botones, se despliegan las acciones correspondientes, resultando en una interfaz más ordenada, clara y amigable.

8.6.2. Adaptación a Dispositivos Móviles

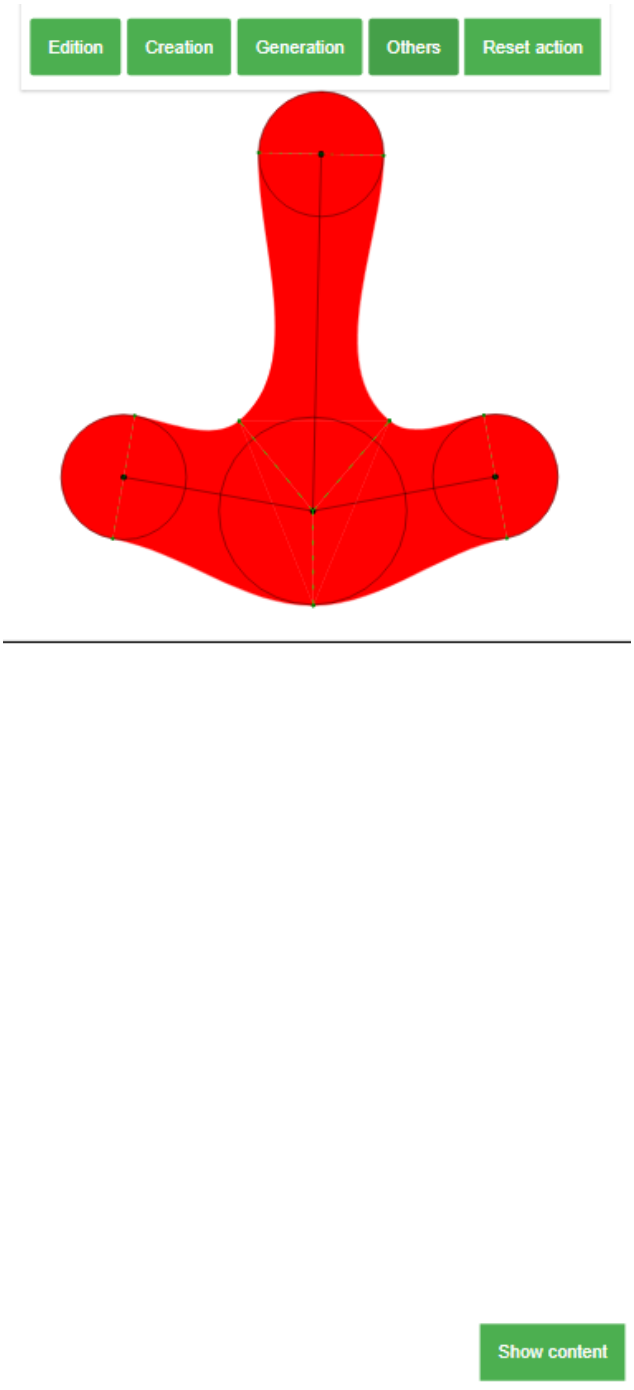


Ilustración 8.16: Interfaz en Dispositivos Móviles

La adaptación a dispositivos móviles no solo implicó ajustes visuales, sino también modificaciones en la interacción dada la naturaleza táctil de estos dispositivos. Algunas de las diferencias clave entre eventos táctiles y eventos de ratón son:

- ✔ **Multiples Toques:** Los dispositivos móviles permiten múltiples toques simultáneos, lo que requiere el manejo de eventos táctiles de manera diferente. Se implementaron funciones específicas, como `touchStarted`, para gestionar esta característica.
- ✔ **Gestos Táctiles:** Los gestos táctiles, como pellizcar para hacer zoom o deslizar para desplazarse, son comunes en dispositivos móviles. Estos gestos fueron incorporados en las funciones táctiles para mejorar la experiencia del usuario.
- ✔ **Sin Clic Derecho:** Dado que no hay clic derecho en dispositivos táctiles, ciertas interacciones, como las que se realizaban con clic derecho en eventos de ratón, fueron adaptadas. Por ejemplo, la función `rightClick` fue reemplazada por una lógica diferente para manejar eventos táctiles.
- ✔ **PreventDefault:** Se utilizó la función `preventDefault` para evitar comportamientos predeterminados en gestos táctiles, evitando problemas como el desplazamiento involuntario de la página al tocar y arrastrar.
- ✔ **Eventos de Movimiento:** Los eventos táctiles, como `touchmove`, fueron empleados para rastrear el movimiento del dedo en la pantalla y realizar acciones correspondientes, como el arrastre de elementos.

Estas adaptaciones permitieron una transición fluida entre la interacción con ratón y táctil, garantizando una experiencia coherente y eficiente en dispositivos móviles. La Figura 8.16 muestra la interfaz resultante después de estos ajustes.

Como se observa en la Figura 8.16, en esta versión se ha introducido un botón denominado 'Mostrar contenido'. Dada la naturaleza más compacta en dispositivos móviles, fue necesario reorganizar ciertos elementos para evitar sobrecargar la página. Este botón mencionado anteriormente permite acceder a información avanzada y ayuda, proporcionando explicaciones detalladas de todas las funciones.

8.6.3. Modificar radio de los círculos arrastrando

Se ha introducido la capacidad de ajustar el radio directamente al crear un círculo. Al crear un círculo de manera individual, si se mantiene presionado el botón del ratón y se arrastra, el radio del círculo se incrementará o disminuirá.

8.6.4. Colores y plantilla en blanco

Se han agregado dos botones: uno que permite cambiar el color de la figura y otro que facilita la creación de una plantilla vacía donde solo hay un círculo en el centro.

8.7. Validación

Posterior a estas modificaciones, se llevó a cabo una validación exhaustiva, tanto a nivel de código como mostrándola a otros usuarios. El objetivo fue verificar que los usuarios se sintieran más cómodos con la nueva versión, que resultara más comprensible y para evaluar el desempeño de la versión móvil.

Capítulo 9

Conclusiones y Trabajo Futuro

9.1. Conclusiones

El desarrollo del proyecto ha permitido alcanzar varios objetivos clave. La creación de una aplicación web interactiva para generar y modificar figuras SVG ha sido exitosa, proporcionando una interfaz amigable y funcional. Se logró una integración efectiva de tecnologías como JavaScript, WebAssembly y C++ para implementar las funciones esenciales de generación y manipulación de figuras.

La refactorización y optimización del código han mejorado significativamente la legibilidad y mantenibilidad del mismo. Se realizaron ajustes en la usabilidad y la experiencia del usuario, incorporando cambios sugeridos por evaluadores externos sin conocimiento previo. La introducción de nuevas funciones, como la inserción y eliminación de puntos en segmentos, ha ampliado las capacidades de la aplicación.

Durante las pruebas de rendimiento, se identificaron y corrigieron errores, como el problema de carga temprana de WebAssembly, la necesidad de limpiar su cache y expandirla.

9.2. Trabajo Futuro

A pesar de los logros alcanzados, hay oportunidades para expandir y mejorar el proyecto en el futuro. Algunas áreas de enfoque incluyen:

- ✔ **Más Funciones de Edición:** Introducir funciones de edición avanzadas, como la posibilidad de cortar, copiar y pegar figuras.
- ✔ **Integración con Sistemas Externos:** Investigar la posibilidad de integrar la aplicación con sistemas externos o plataformas de diseño gráfico.
- ✔ **Permitir diferentes patrones de colores:** Posibilitar que cada segmento de la figura pueda tener colores distintos, brindando mayores opciones creativas.

- ✔ **Funcionalidad de Guardado en el Entorno Web:** Implementar la capacidad de guardar varias figuras en el entorno web, permitiendo así la salvaguarda del estado de alguna figura para su recuperación en el futuro. Esta funcionalidad también facilitaría la comparación entre diferentes figuras.
- ✔ **Añadir funciones visuales:** Incorporar nuevas funciones visuales, como la posibilidad de añadir bordes a las figuras y permitir que las figuras independientes tengan diferentes colores para una mayor personalización estética.

Estos puntos representan solo algunas de las áreas donde se podría continuar trabajando para mejorar y hacer evolucionar la aplicación en el futuro. El proyecto proporciona una base sólida para futuras iteraciones y desarrollo continuo.

Capítulo 10

Manual de usuario

Fuera de las agrupaciones, se encuentran los botones de:

- ✔ **Reset action:** Reinicia la acción que está en espera, anulando todo lo que está pendiente de suceder.
- ✔ **Help:** Muestra una breve explicación de cada botón.
- ✔ **Advance Info:** Muestra el área de texto donde se ubica el shape, permitiendo su modificación manual. También incluye los botones **Compute shape**, **Export shape**, y **Upload shape**.
- ✔ **Compute shape:** Envía el shape del área de texto para ser ejecutado por el programa y redibuja el SVG.
- ✔ **Export shape:** Exporta el shape en formato .txt.
- ✔ **Upload shape:** Permite cargar un shape dentro del área de texto.

10.1. Edición

- ✔ **New circle in the middle of a segment:** Al hacer clic en un segmento, añade un círculo en el centro del segmento, unido a los dos círculos a los que pertenecía el segmento. Esto crea dos nuevos segmentos, como si se agregara un 'codo' al segmento.
- ✔ **Erase a circle:** Elimina un círculo al hacer clic sobre él.
- ✔ **Erase multiple circles:** Elimina un círculo al hacer clic sobre él, pero la acción no se completa hasta hacer clic en **Reset action**.
- ✔ **Remove a segment:** Elimina un segmento al hacer clic sobre él.

10.2. Creación

- ✔ **Create a circle:** Crea un círculo al hacer clic.
- ✔ **Create multiple circles:** Crea un círculo al hacer clic, pero la acción no se completa hasta hacer clic en **Reset action**.
- ✔ **Add segment by clicking in extreme circles:** Tras pulsar el botón, el usuario puede crear un segmento entre dos círculos, haciendo clic en uno y luego en otro.

10.3. Generación

- ✔ **Generate random shape:** Crea una figura totalmente aleatoria.
- ✔ **Generate similar shape:** Crea otra figura aleatoria, pero con datos del shape similares. Por lo tanto, no varía en número de círculos, segmentos, etc.

10.4. Otros

- ✔ **Redo y Undo:** Botones con flechas hacia adelante y hacia atrás que permiten deshacer y rehacer cambios.
- ✔ **Download SVG:** Descarga la figura en formato .svg.
- ✔ **Blank template:** Deja la plantilla vacía, excepto por un círculo en el centro.
- ✔ **Preview:** Permite visualizar cómo resultaría la figura final, sin marcas, líneas o bordes visibles.
- ✔ **Colours:** La rueda con diferentes colores despliega un menú para cambiar el color del SVG.

Bibliografía

[1] D3.js (2023). D3.js documentation. https://d3js.org/getting-started.

[2] Emscripten (2023). Emscripten - optimizing code. https://emscripten.org/docs/optimizing/Optimizing-Code.html.

[3] freeCodeCamp (2023). Qué es svg - explicación de etiquetas e imágenes. https://www.freecodecamp.org/espanol/news/que-es-svg-explicacion-de-etiquetas-imagenes/#:~:text=SVG%20o%20GrÃąficos%20Vectoriales%20Escalables,dentro%20de%20una%20ventana%20grÃąfica.

[4] Git SCM (2023). Git - documentation. https://git-scm.com/doc.

[5] GitHub (2023a). Github docs. https://docs.github.com/es.

[6] GitHub (2023b). Github pages. https://docs.github.com/es/pages.

[7] Javascript.info (2023). Curva de bézier. https://es.javascript.info/bezier-curve.

[8] JavaScript.info (2023). Curva de bézier - javascript.info. https://es.javascript.info/bezier-curve.

[9] JetBrains (2023). Getting started with webstorm. https://www.jetbrains.com/help/webstorm/getting-started-with-webstorm.html.

[10] López, N. M. (2024). Interactive visualization of ipol demos.

[11] math.hws.edu (2023). Bezier curve demos. https://math.hws.edu/eck/cs424/notes2013/canvas/bezier.html.

[12] Mozilla Developer Network (2023a). Cache-control - mozilla developer network. https://developer.mozilla.org/es/docs/Web/HTTP/Headers/Cache-Control.

[13] Mozilla Developer Network (2023b). Webassembly guide - mozilla developer network. https://developer.mozilla.org/en-US/docs/WebAssembly/C_to_wasm.

[14] Overleaf (2023). Overleaf documentation. https://www.overleaf.com/learn.

[15] W3SchoolsCPP (2023). W3schools - c++ tutorial. https://www.w3schools.com/cpp/.

[16] W3SchoolsCSS (2023). W3schools - css tutorial. https://www.w3schools.com/css/.

[17] W3SchoolsHTML (2023). W3schools - html tutorial. https://www.w3schools.com/html/.

[18] W3SchoolsJS (2023). W3schools - javascript tutorial. https://www.w3schools.com/js/.