UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

AMBIENTE GRÁFICO PARA LEVANTAMIENTOS ARQUITECTÓNICOS EN WINDOWS RT CON .NET 4.5.1 BASADOS EN DISTANCIÓMETROS INALÁMBRICOS BT

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A N

Luis Alejandro Aguilar Díaz

Miguel Ángel Alanis Montes

DIRECTOR DE TESIS

M.I. Gabriel Mauricio Álvarez Medina

Ciudad Universitaria, Cd. Mx., 2016

Índice General

Índice de Ilustraciones

1. Introducción

Un levantamiento se define como un planteamiento arquitectónico que implica cualquier cambio (que puede ser desde una remodelación hasta una construcción desde cero) en un espacio geográfico basado en un proceso de investigación [1]. En la actualidad el objetivo final de un levantamiento es recolectar información del espacio para contar con el conocimiento necesario del estado actual del proyecto a tratar.

El éxito dependerá de la definición de los trazos y de las mediciones descritas en el planteamiento arquitectónico representado por un dibujo a mano alzada.

El proceso de levantamiento es tardado y requiere experiencia por lo que surge la necesidad de optimizar los tiempos.

Debido al auge en los dispositivos móviles, actualmente se cuenta con las herramientas que permiten resolver estos tipos de problemas de manera más rápida y eficiente.

En el mercado actual existen distintas soluciones, algunas tienen la precisión necesaria, pero carecen de naturalidad al momento de dibujar, mientras que las que son más intuitivas al usuario, tienen deficiencia en sus procesos de medición. La precisión necesaria dentro de la aplicación es de ±1 mm y se definió con base en la información obtenida de los arquitectos entrevistados.

Entre las aplicaciones tenemos las que se muestran en la tabla **1.1**:

Tabla de aplicaciones		
Aplicación	**Precisión necesaria**	**Intuitivas al usuario**
AutoCAD® *360*	✓	✗
Topography App	✗	✗
Surveying®	✗	✓
CAD Touch Free	✗	✓

Tabla 1.1 Tabla de aplicaciones para realizar levantamientos

1.1. Objetivo

Los objetivos propuestos son:

- Acelerar los tiempos del proceso de levantamiento sin disminuir la precisión de las mediciones
- Recopilar la información en un formato para que pueda ser leído por *AutoCAD®*, el cual es uno de los softwares más utilizados en el área de Diseño Asistido por Computadora con un porcentaje cercano al 70% mundial
- Habilitar la conectividad de un distanciómetro *Bluetooth®* para obtener mejores mediciones
- Aprovechar las características del dispositivo móvil y sacar ventaja de la portabilidad de la información

1.2. Justificación

Con base en los objetivos, se planteó este tema como una ayuda a aquellas personas que se dedican al proceso de levantamientos, dejando como resultado final una herramienta que les permita hacer su trabajo de forma más eficiente, así como permitir que aquellos que no cuenten con demasiada experiencia puedan realizarlo sin la necesidad de invertir en un especialista.

En el ámbito de nuestro desarrollo personal el tema de tesis nos permitirá ampliar nuestros conocimientos hacia el área de *Modelado de Información de Construcción (BIM)* (**J1**), así como mejorar nuestras técnicas de programación en .NET.

1.3. Metodología

Con base en la investigación realizada, se separó el proceso de levantamientos arquitectónicos en módulos de programación. Cada uno de estos utiliza un conjunto de clases con tareas específicas (**VER ILUSTRACIÓN 1**), las cuales fueron implementadas usando el patrón de arquitectura *Modelo Vista Controlador (MVC)* (**J3**). En los casos necesarios se tuvieron que implementar algoritmos abordados en las materias de computación gráfica, geometría analítica y métodos numéricos.

El flujo de la aplicación (**VER ILUSTRACIÓN 1.2**) comienza con el módulo de proyectos, en el cual se declara la información del proyecto de levantamiento, incluyendo datos del cliente y los materiales que se usarán en el proceso de cuantificación. También en este proceso, el usuario define el número de niveles, así como el número de espacios por nivel que tendrá el proyecto arquitectónico. Por otro lado, se permite seleccionar el croquis del lugar mediante los servicios de *Bing Maps* (**VER** ERROR! REFERENCE SOURCE NOT FOUND.), así como especificar la dirección del mismo.

Posteriormente en el módulo de levantamientos, se definen las funciones de trazo, dimensionamiento, creación de arcos y la asignación de muros, puertas y ventanas.

Finalmente, en el módulo de materiales, se realizan las funciones de asignación y cuantificación de materiales, así como la generación de reportes sencillos y la exportación a AutoCAD.

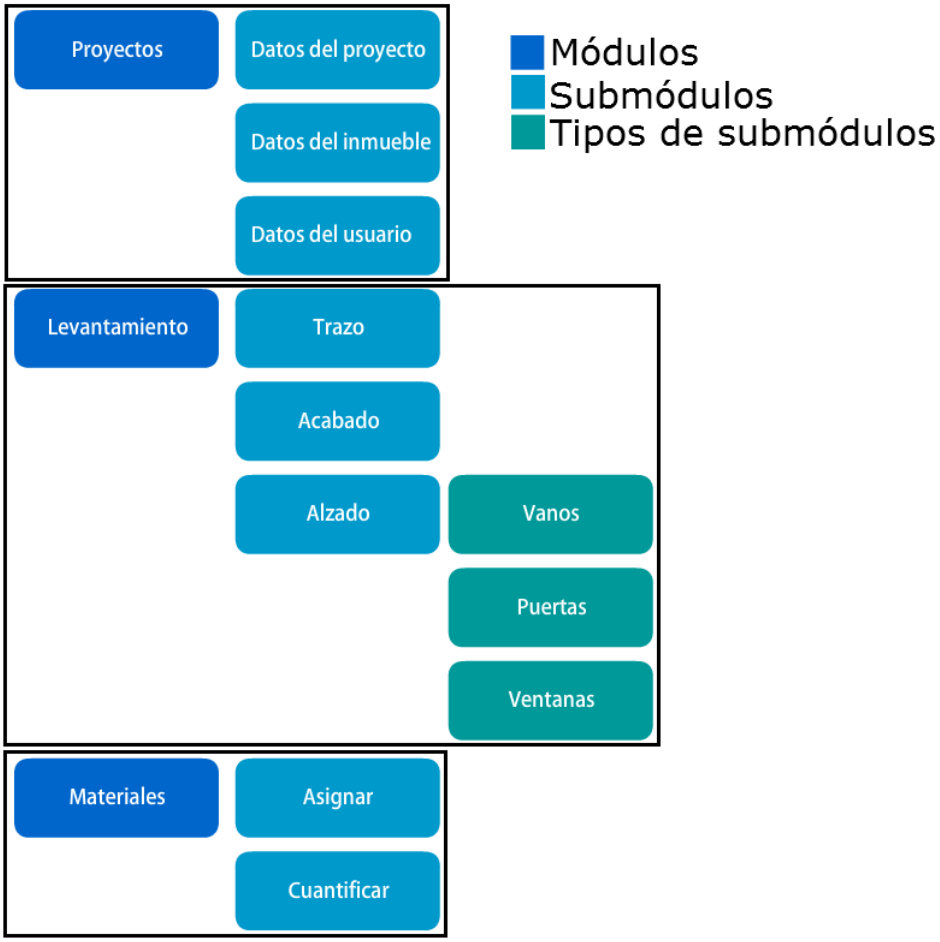


Ilustración 1.1 Diagrama de módulos.

1.4. Narrativa por capítulos

- **Motor gráfico**

  Definimos el motor gráfico 2D de la aplicación. A este motor le dedicamos la mayor parte de nuestro tiempo debido a la naturaleza de la aplicación y a la limitación del API de *.NET®* para las aplicaciones de la tienda de *Windows®*.

- **Análisis geométrico**

  Optimizamos el proceso de los cálculos geométricos con algoritmos concisos para obtener de forma accesible y consistente los datos que serán usados posteriormente en las cuantificaciones.

- **Diseño de la geometría**

  Desarrollamos la metodología para acotar los segmentos de muros con etiquetas que describen los principales componentes del dibujo como el proceso de creación de curvas en la geometría de los muros, y la unión de los mismos.

- **Modelo de la aplicación**

  Definimos la estructura básica de la aplicación y la implementación de las características del proyecto.

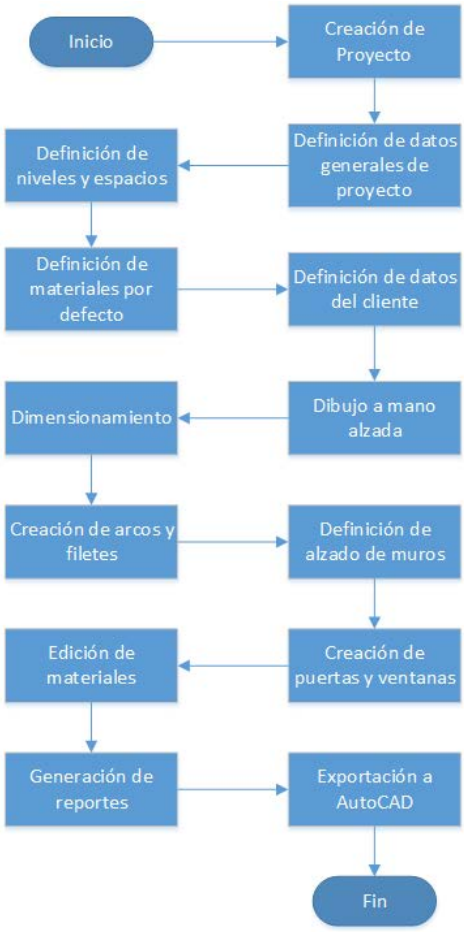


Ilustración 1.2. Flujo de la aplicación

- **Periféricos y sensores**

  Agregamos la compatibilidad con los periféricos del dispositivo para las funciones específicas de la aplicación, tales como la cámara, el GPS, la brújula y la comunicación a través de *Bluetooth®*.

- **Cuantificación**

  Creamos una estructura para asociar una clave a un elemento en el plano y poder cuantificarlo.

- **Interfaz gráfica**

  Diseñamos los controles necesarios para que la aplicación tenga un funcionamiento más intuitivo al usuario, así como complementar la vista de otros módulos.

- **Pruebas y conclusiones de la aplicación**

  Describimos las pruebas que realizamos durante el desarrollo de la aplicación y presentamos nuestra conclusión.

2. Motor gráfico

En este capítulo se explican las principales herramientas que se desarrollaron para la creación del trazo a mano alzada y los algoritmos que se utilizaron para cada una de estas herramientas. El trazo es la base de todas las funcionalidades de la aplicación. Por último, se detallan los procesos de desplazamiento y acercamiento, los cuales, aunque no sean parte del funcionamiento principal, son características fundamentales del motor gráfico.

2.1. Estructura básica

2.1.1. Canvas (Lienzo) [2]

Como la aplicación se ejecuta en un entorno de *Windows® Store Application* se buscó un control que tuviera la capacidad de dibujar en él, que facilitará la asociación de elementos y que pudiera simular un entorno 2D. El control que cumple con estas características fue el *Canvas*, que pertenece al espacio de nombre *Windows.UI.Xaml.Controls*.

En la documentación, el control *Canvas* queda definido como un área en el que se agregan elementos de tipo *FrameworkElement* con coordenadas cartesianas, las cuales son relativas al área del *Canvas*.

El *Canvas* tiene un sistema de coordenadas de referencia en donde el origen se centra en la esquina superior izquierda del control. Las coordenadas en X hacen referencia a la propiedad *Left* del Objeto insertado en el *Canvas* y las coordenadas en Y hacen referencia a la propiedad Top (**VER ILUSTRACIÓN 2.1**).

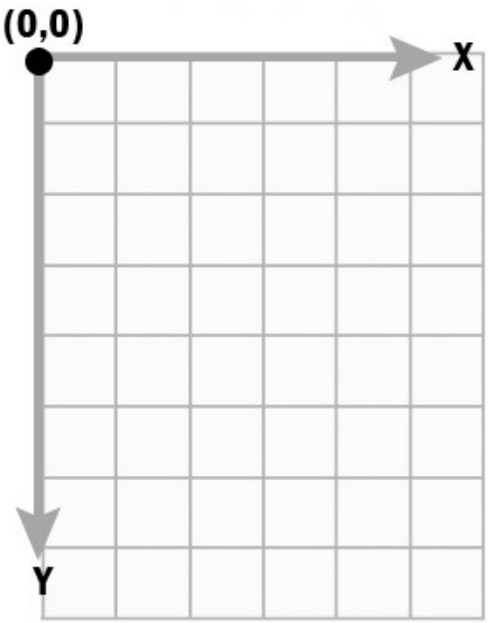


Ilustración 2.1. Sistema de coordenadas del Canvas

El *Canvas* guarda los elementos en su propiedad *Children*, la cual es de tipo *UIElementCollection* y es heredada de la clase Panel (**VER ILUSTRACIÓN 2.2**).

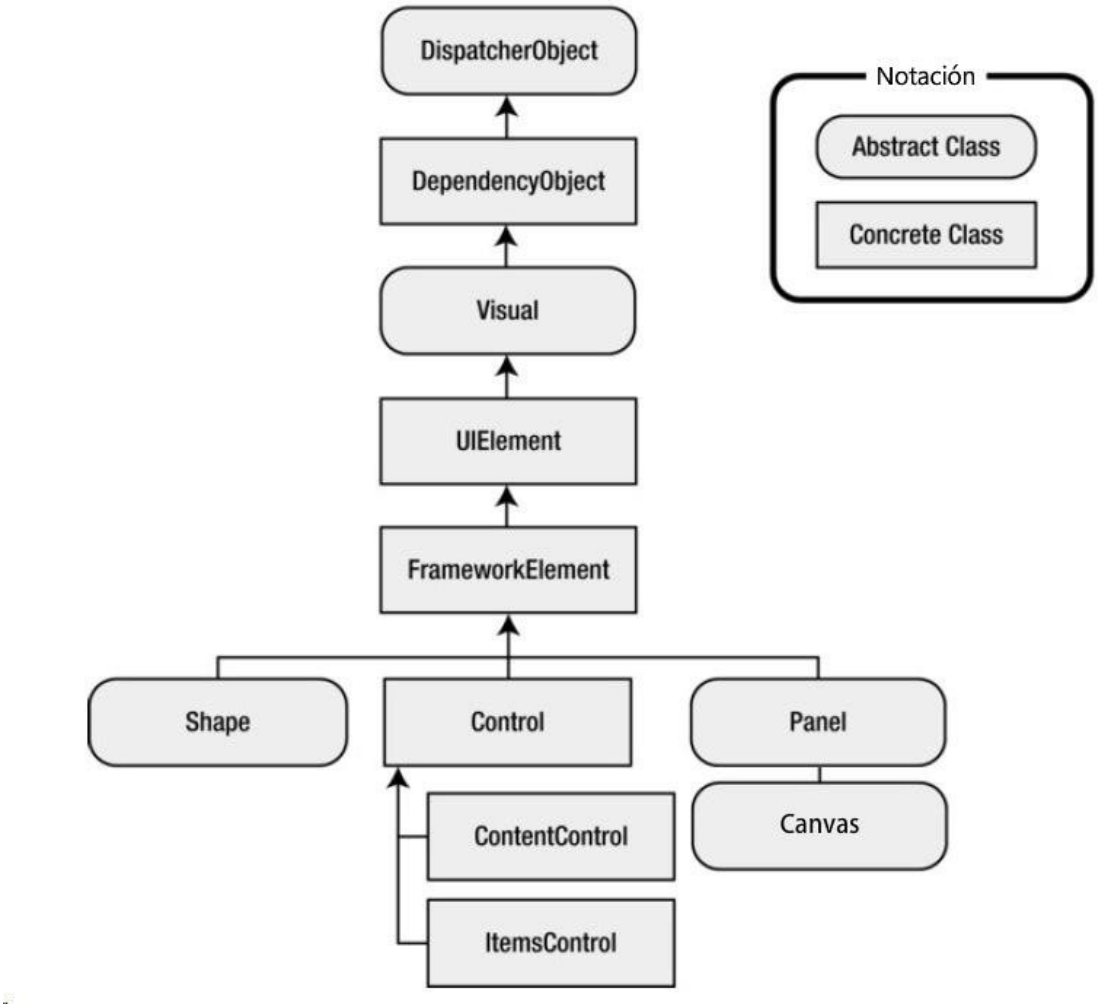


Ilustración 2.2. Diagrama de clases de los elementos WPF [2]

Para nuestra aplicación los elementos gráficos son de tipo *Path*, debido a que este objeto permite la agrupación de distintas geometrías, las cuales se forman con segmentos de línea, o arcos.

Para cada elemento gráfico se definieron las siguientes operaciones.

- **Dibujado**

  Agrega el elemento al *Canvas*, en caso de que ya exista y tuviera su propiedad de visibilidad con valor *Collapsed*, este método lo hará visible nuevamente
- **Borrado**

  Elimina el elemento de la lista de los hijos del *Canvas*
- **Ocultar**

  Cambia la propiedad de visibilidad de los elementos contenidos en el *Canvas* a *Collapsed*, que es similar hacerlos invisibles

- **Actualizar**
  La información es actualizada y si cambio la geometría se encarga de regenerar el *Path*
- **Transformar**
  En caso de que exista alguna transformación geométrica en el plano, este método se encarga de ejecutarla. Después de aplicar la geometría se llamará al método *Update*
- **Regenerar**
  Se utiliza para cambios de estilo de los elementos contenidos en el *Canvas*, como cambiar el color, el grosor de la línea o estilo de letra en caso de los Textos

Para mantener el *Canvas* con una resolución constante se utilizó un tamaño fijo de 925 x 520 pixeles. Este tamaño tiene una proporción 16:9 la cual es estándar en la mayoría de las pantallas como la *Windows Surface RT* (que fue la que utilizamos inicialmente en nuestro proyecto). Pero debido a que los diferentes modelos tienen distintas resoluciones encapsulamos el *Canvas* dentro del control *ViewBox*, y con eso logramos manejar una resolución fija en un espacio variable.

Por ejemplo, la Surface 1 RT tiene una resolución de 1366 x 768, y la Surface 2 una resolución 1920 x 1080 (VER ILUSTRACIÓN 2.3), lo que significa que el *Canvas* ocuparía distintos tamaños en la pantalla. Al utilizar el *ViewBox* se aplica una transformada geométrica que mantiene un aspecto equilibrado sin importar la resolución de la pantalla.

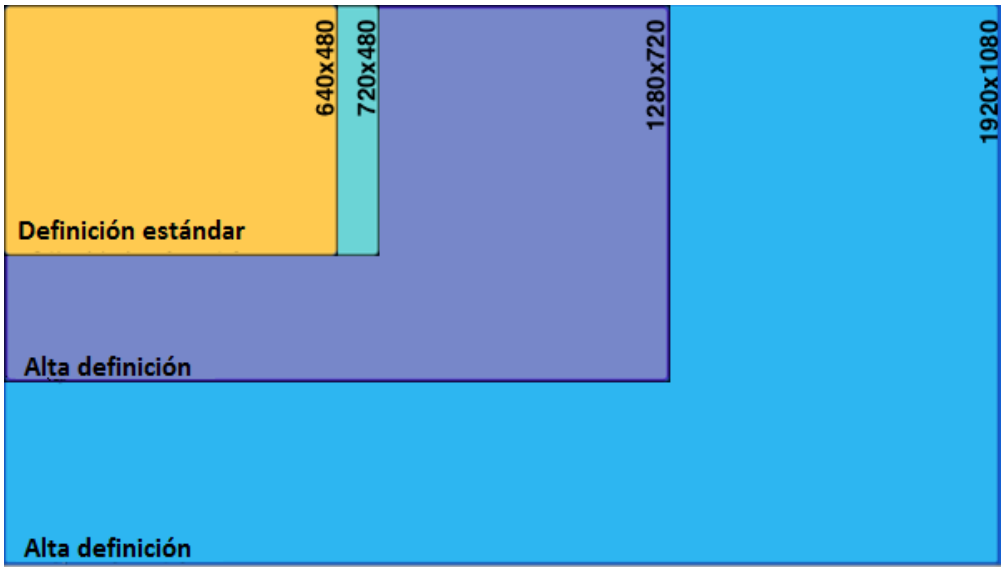


Ilustración 2.3. Aspectos de pantalla

Los elementos de la aplicación utilizan un conjunto de unidades que se definen a una escala 1:100, donde una unidad de estas equivale a 100 pixeles. Estas unidades son llamadas unidades gráficas (VER ILUSTRACIÓN 2.4).

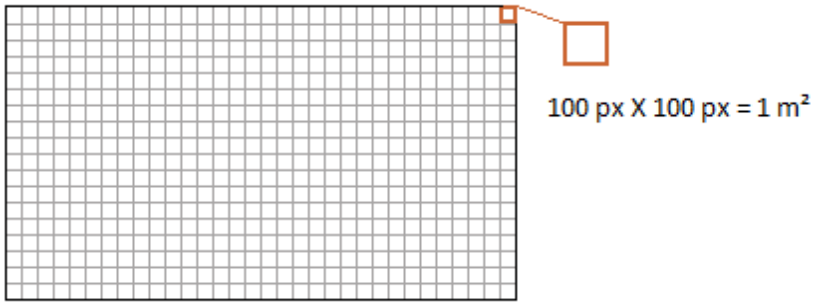


Ilustración 2.4. Unidades gráficas

2.1.2. Puntos (Point y Point2D)

Debido a la naturaleza grafica de la aplicación, la unidad mínima que consideramos es un vértice, el cual se representa por medio de un punto. Para almacenar y manipular estos datos utilizamos dos estructuras: *Windows.Foundation.Point* y *Point2D*. La primera de ellas consiste en un arreglo de dos dimensiones las cuales se aproximan a la distribución de pixeles en la pantalla, de aquí la necesidad de crear un nuevo punto, ya que *Point* no podía ser referenciado y no proporcionaba la precisión requerida.

La estructura de *Point2D* se define de la siguiente manera;

- Una componente de X y de Y al igual que *Point* de tipo Doble
- Un enumerador (*GeometryStatus*) que permite definir el estado en el que se encuentra el punto, esta parte se utiliza en la parte de dimensionamiento
- Un método para referenciar el punto, para que en lugar de usar las coordenadas de la estructura utilice las guardadas en la estructura de bloque de la aplicación, en los siguientes temas se hablará de que es un bloque

La funcionalidad de la clase puede revisarse en el apéndice de funcionalidad de puntos (**APÉNDICE A**).

2.1.3. Segmentos y tipos de segmentos (LineSegment, Segment2D y Diagonal2D) [2]

Los segmentos de recta se encuentran dibujados dentro de la clase *Figure*, la cual pertenece a la colección de figuras del *Path* (**VER ILUSTRACIÓN 2.5**). Para nuestros segmentos la colección de figuras contendrá solo una recta.

El objeto *Figure* se define con un punto de inserción y una colección de segmentos, al ser una línea recta usamos la clase *LineSegment*, que requiere solo de un punto final para su creación. El punto inicial será el punto de inserción de la figura.

Se decidió dibujar los segmentos por separado para facilitar la selección individual de los mismos, así como la asignación de estilos de línea.

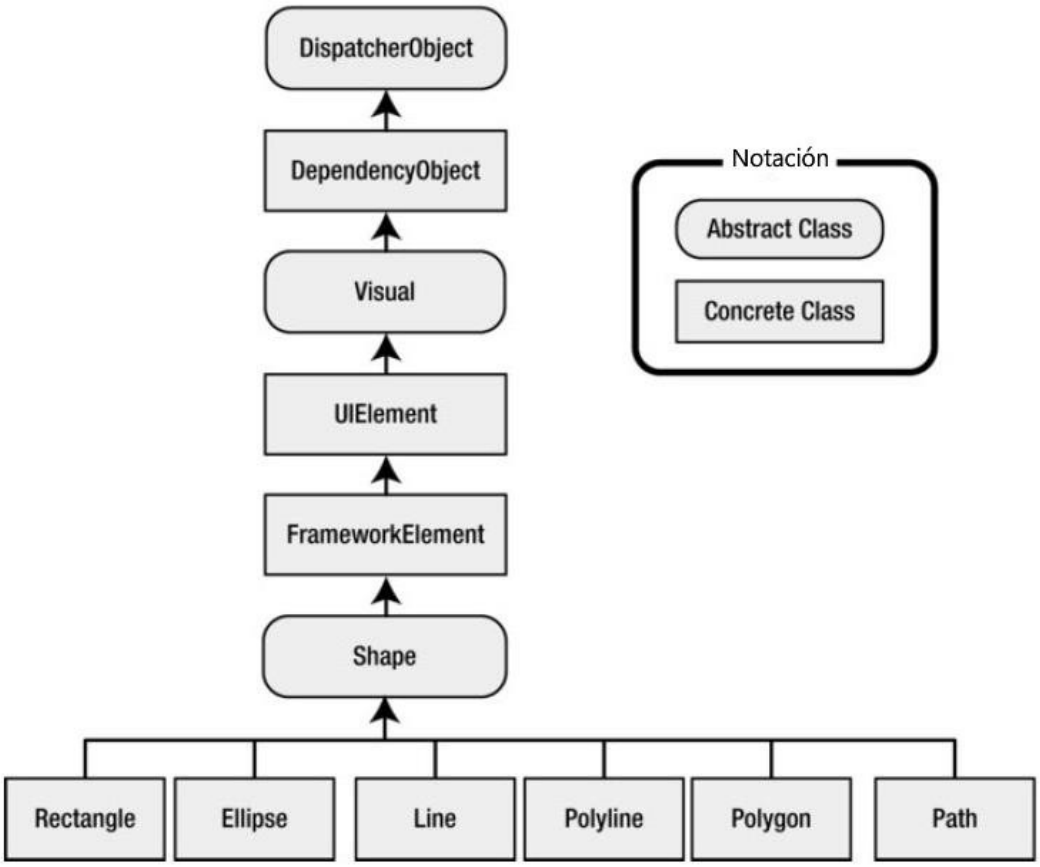


Ilustración 2.5. Diagrama jerárquico de la clase Path [2]

La geometría se define con un *Path* y se administra con la clase *Segment2D* la cual utiliza unidades gráficas, el *Segment2D* lo definimos de la siguiente manera.

- **Punto inicial**
  Define el punto donde comienza el segmento de línea
- **Punto final**
  Define el punto donde finaliza el segmento de línea
- **Punto medio**
  Define el centro de la línea. Se utilizaron los métodos de *Point2D* (**A2**)
- **Etiquetas (Tags)**
  Define el nombre y la dimensión de manera visual, la etiqueta de forma predeterminada se inserta en el punto medio del segmento de línea

Al ser un elemento gráfico contará con todos los demás métodos descritos en el tema **2.1.2** *Diagonal2D* es una sub-clase de *Segment2D* que nos permite diferenciar las diagonales internas de la figura, las cuales nos permitirán la funcionalidad de dimensionar.

2.1.4. Polyline2D y AcabadoPolyline2D

Existen dos tipos de polilínea, la primera es llamada polilínea de trazo y se encuentra encapsulada en la clase *Polyline2D*. También llamada de trazo por que conforma la geometría que genera el usuario mediante el dibujo a mano alzada. Se compone de una colección de segmentos, triángulos y diagonales.

Como las geometrías dibujadas pasan por un proceso de ortogonalización y optimización no se permiten segmentos curvos en el trazo del usuario.

En un principio, solo se guardarán los segmentos, pero mientras avanza el proceso de la aplicación se agregarán las diagonales y posteriormente los triángulos. Estos dos últimos se generan con el proceso de triangulación y se utilizan para el proceso de dimensionamiento por lo que se detallarán a fondo en la sección de triangulación. Entre las funciones y propiedades asignadas a esta clase tenemos

- **Cálculo del centroide**

  Se usa para asignar la posición de la etiqueta de área. Se calcula de la siguiente forma como se muestra en la **ECUACIÓN 2.1**:

$$\text{Centroide}(x, y) = \frac{\sum \text{Centroide}_{triángulo}}{\text{número de triángulos}} \qquad (2.1)$$

  Donde $c_{triángulo}$ es el Centroide de cada triangulo y $n_{triángulos}$ es el número total de triángulos.

- **Segmento Inicial**

  Regresa el primer segmento.

- **Segmento final**

  Regresa el último segmento.

- **Triangular**

  Utilizando diversos algoritmos que posteriormente se detallarán, divide la geometría en triángulos, los cuales se requieren durante el proceso de dimensionamiento.

- **Exportar geometría**

  Genera la geometría real basada en las unidades definidas por el usuario.

- **Vértices**

  Una estructura de tipo *Point2D* con la diferencia de tener un doble apuntador en donde un apuntador apunta al vértice anterior y el otro al siguiente. Si el vértice es el último vértice de la polilínea apuntará al primer elemento y si el primero apuntará al último. Además, cada vértice guarda la normal de rotación del respecto al segmento anterior y el segmento siguiente.

- **Zoom extendido**

  Genera un puerto de vista de la caja de colisión expandida al 120 % de la figura y la

centra en la pantalla.

La segunda polilínea es llamada polilínea de acabado. Esta se dibuja en sentido de las manecillas del reloj y hace referencia a la geometría real del lugar. Es generada al terminar el proceso de dimensionamiento. Los segmentos se convierten en muros y pueden ser de dos tipos de geometría.

- **Muros rectos**
- **Muros curvos**
    - **Arco de filete**

      Es un arco con ángulo menor o igual a 90 grados y pueden ser definidos en sentido horario o anti-horario
    - **Medio arco**

      Son dibujados siempre en sentido horario y con un ángulo de 180 grados

Esta polilínea guardará la geometría en planta y la colección de muros.

2.2. Rejilla de dibujo (Grid)

La rejilla o *Grid* en inglés, es un arreglo de puntos con un espacio constante entre ellos, que tiene como objetivo facilitar la proporcionalidad del espacio real con el dibujo realizado a mano alzada.

Se cuenta con dos tipos de rejilla: La primera de ellas con una separación de 1 metro en escala 1:1 que equivale a 100 pixeles dentro del *Canvas*. La otra rejilla tiene una separación de medio metro también en escala 1:1 lo que equivale a un espacio de 50 pixeles. Otra diferencia respecto a la primera es la tonalidad que es más clara (**VER ILUSTRACIÓN 2.6**).

Cuando inicia un proyecto en blanco, se tiene un área de dibujo de 9.25m por 5.25m. Al aplicar un acercamiento, las rejillas tienen una limitante en su visibilidad. Si el espacio alcanza un tamaño muy grande, el *Grid* se hace invisible.

Los criterios son los siguientes:

- La rejilla de medio metro desaparece a los 28m por 16 m
- La rejilla de metro desaparece a los 40m por 23m

La visibilidad de la rejilla es controlada con las capas de rejilla. Una capa se encuentra en el modo de trazo y la otra en el modo de acabado.

Para dibujar la rejilla, primero se calcula el tamaño del cuadrado y se divide el arreglo del *Canvas* con este tamaño obteniendo un límite en X y un límite en Y. Posteriormente, se inserta un punto en cada coordenada que satisfaga la **ECUACIÓN 2.2**:

$$\{(\blacksquare\blacksquare, \blacksquare\blacksquare) \mid 0 \leq \blacksquare\blacksquare \leq \frac{\blacksquare_{\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare}}{\blacksquare_{\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare}} + 1, 0 \leq \blacksquare\blacksquare \leq \blacksquare_{\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare}$$

$+ 1\}$ (2.2)

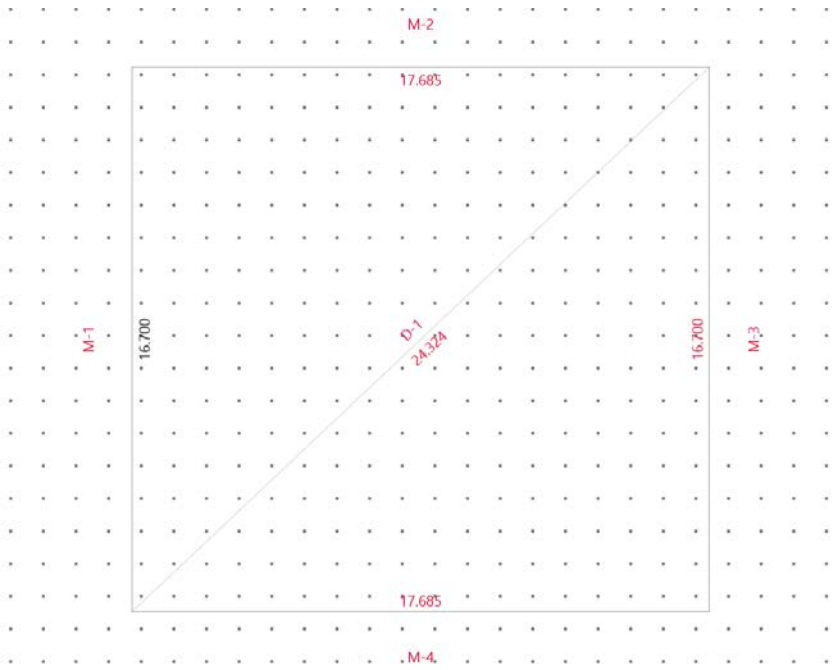


Ilustración 2.6. Rejilla de la aplicación sin los puntos intermedios

Donde (�,) son las coordenadas de cada punto de la rejilla, y son las dimensiones del lienzo de dibujo y son las dimensiones de la rejilla.

El punto es calculado utilizando la Ecuaciones 2.3, 2.4:

$$\text{[illegible]} = \text{[illegible]} * \text{[illegible]} \quad (2.3)$$

$$\text{[illegible]} = \text{[illegible]} * \text{[illegible]} \quad (2.4)$$

Debido a que no se puede dibujar un punto en el *Canvas*, en un *Path* generamos un arreglo de líneas con un tamaño de dos pixeles.

La rejilla está sujeta a las acciones de desplazamiento y acercamiento, por lo que directamente recibe un cambio en el renderizado con base en la transformada global.

2.3. Captura de puntos [2]

Para la captura de puntos y las interacciones con el *Canvas* se diseñó un flujo de trabajo que está basado en tres eventos del *Canvas*:

- Pointer Pressed
- Pointer Moved
- Pointer Released

2.3.1. Pointer Pressed

Este evento consiste en detectar cuando se entra en contacto con la pantalla ya sea con el puntero del mouse o con el toque táctil. Existen ciertas banderas que limitan el funcionamiento de este evento, entre las que tenemos:

- **IsInTagEvents**
  Esta bandera evita que el funcionamiento de este evento se realice cuando se presiona una etiqueta
- **Draw**
  Esta bandera indica que se puede dibujar y da inicio al proceso de captura de puntos
- **Multi_Contact**
  Esta bandera se activa cuando la pantalla detecta más de un contacto simultáneo

2.3.2. Pointer Moved

Este evento consiste en detectar el movimiento y la velocidad del puntero utilizando el cambio de coordenadas mientras el contacto en la pantalla no sea interrumpido. También contiene ciertas banderas entre las que tenemos:

- **IsInTagEvents**
  Esta bandera evita que el funcionamiento de este evento se realice cuando se esté presionando una etiqueta
- **Drawing**
  Al estar activada, crea los vértices utilizando las coordenadas por las que pasa el puntero dibujando el trazo tal y como lo realizó el usuario
- **Zoom**
  En caso que *Multi_Contact* haya sido activado y la distancia de los punteros presente una diferencia contra la distancia original, se generará un evento de *PinchIn* o *PinchOut*. Estos eventos serán definidos durante el proceso de Zoom (**APÉNDICE G4**)
- **Pan**
  En caso de tener una geometría dibujada, permite el desplazamiento en la pantalla mediante el toque de un dedo

2.3.3. Pointer Released

Este evento consiste en detectar el momento en que el usuario termina el contacto con la pantalla, interrumpiendo así el evento anterior e iniciando el procesamiento de los puntos capturados. Además, resetea el número de contactos permitiendo que se pueda volver a iniciar cualquier otro proceso. Este evento cuenta con las siguientes banderas:

- **IsInTagEvents**: Si esta bandera está activa y el puntero se encuentra sobre una etiqueta se ejecuta el evento que tenga asignada la etiqueta
- **Drew**: Si se encuentra activa inicia el proceso de creación de la geometría

2.3.4. Configuración de la captura de puntos (*CanvasSettings*)

El objetivo de esta clase es guardar la información necesaria para controlar las acciones que se realizan sobre el *Canvas* en el transcurso de los tres eventos descritos anteriormente. Entre las propiedades de la clase encontramos:

- **CanvasTimer**
  Se encarga de definir el tiempo mínimo que debe existir entre la captura de dos puntos que en nuestro caso equivale a 80 ms. Se eligió este tiempo debido a que disminuía la carga del sistema durante el proceso de captura sin afectar la precisión requerida**.**
- **PointerDeviceType**
  Describe si el puntero es de tipo *Touch*, *Mouse* o *Pen.*
- **LastDrawnPoint**
  Guarda el último punto dibujado como *Point2D.*
- **PreviouslyDrawnPoint**
  Guarda el punto anterior como *Point2D.*
- **DiscretePoints**
  Contiene una lista de los puntos discretizados.
- **NumOfContacts**
  Número de punteros registrados en el proceso actual. Solo está disponible si el puntero es de tipo *Touch.*

2.3.5. Lectura de trazo del usuario

Esta lectura se utiliza únicamente para dibujar el trazo a mano alzada del usuario y consiste en la creación de un segmento entre el *PreviouslyDrawnPoint* y *LastDrawnPoint*. Cada segmento es dibujado en el *Canvas* como un hijo independiente.

Fue necesario para los cálculos vectoriales definir un sentido de trazo. Debíamos dar una dirección al dibujo, ya fuese en sentido o en contra de las manecillas del reloj. Optamos por el sentido horario ya que la mayoría de los trazos comenzaban a dibujarse hacia esa dirección.

El trazo real es medido directamente del objeto *PointerPoint*, el cual es una propiedad de los argumentos que reciben los eventos del *Canvas*. Como los puntos que guardamos son de tipo *Point2D* creamos un constructor en la clase *Point2D* que recibe un Point de *Windows*®.

2.3.6. Discretización de puntos

El objetivo de este proceso es reducir el número de puntos leídos sin cambiar la geometría dibujada por el usuario. Esto fue necesario debido a que al capturar todos los puntos que se detectaban, se alentaba demasiado la tarea de vectorización.

Usando *PreviouslyDrawnPoint* y *LastDrawnPoint* se generó un método que limita el número de puntos a trabajar en el proceso de vectorización.

La primera condición consiste en que se superen los 80 ms entre las lecturas del *PreviouslyDrawnPoint* y *LastDrawnPoint*. Una vez que el tiempo fue superado, se realiza la siguiente condición, la cual se define como una comparación usando la función de similitud descrita en el punto 1.1.2 con una constante de error de $4_{\diamond\diamond\diamond\diamond\diamond\diamond\diamond}$

^{2}.

Posteriormente, se analiza si el punto siguiente se encuentra en esta área, en cuyo caso no se actualiza el *PreviouslyDrawnPoint* y se calcula el nuevo valor de *LastDrawnPoint*. En caso contrario *LastDrawnPoint* se guarda en la colección de puntos *DiscretePoints*, y se le asigna el valor de último punto guardado a *PreviouslyDrawnPoint*.

Por ejemplo, al dibujar la figura (**VER ILUSTRACIÓN 2.7**), el trazo original contenía 192 puntos y al aplicar el proceso de discretización los puntos fueron reducidos a 26.

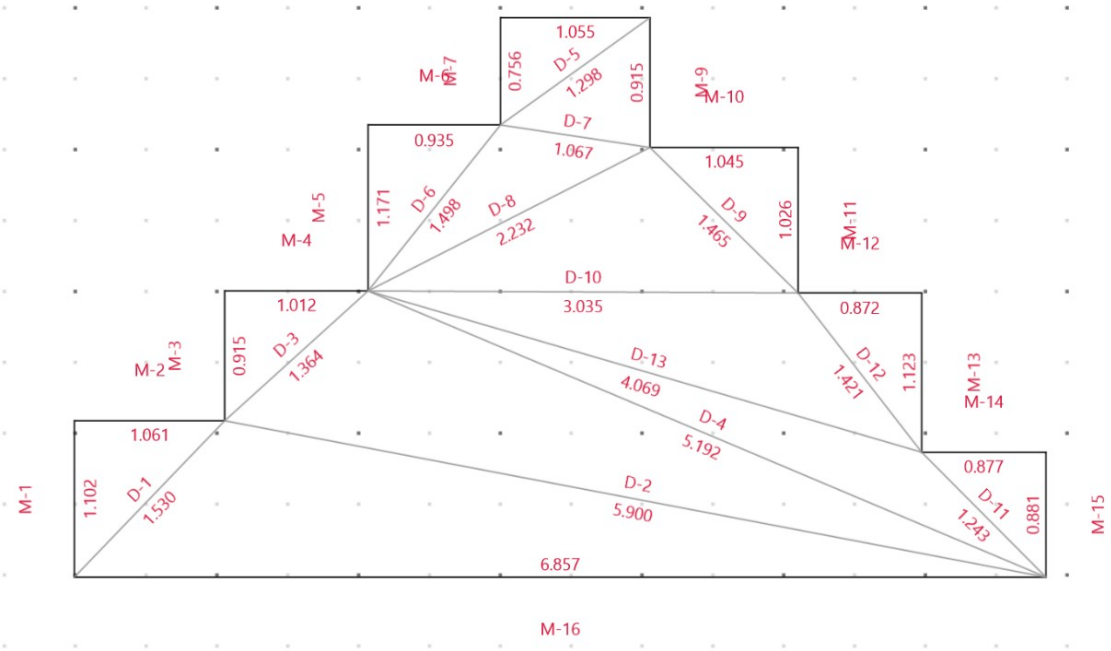


Ilustración 2.7. Ejemplo de trazo después del proceso de discretización

2.4. Desplazamiento (Panning)

Se define como desplazamiento, al cambio del puerto de vista mediante una transformada de translación (**APÉNDICE E6**). El proceso da inicio al presionar la pantalla si la bandera de desplazamiento está habilitada. El primer paso del proceso, es registrar el punto de contacto inicial. Se mantiene el contacto con el dispositivo (*Evento PointerMoved*) y la densidad de puntos se controla con la clase *CanvasTimer*.

La acción consiste en registrar el punto final y generar un vector de desplazamiento que va del punto inicial al último punto registrado. Este vector se usa para generar la transformada de desplazamiento que modifica el puerto de vista. Una vez terminada la transformada, el

punto inicial se convierte al último punto registrado y se espera a que el *CanvasTimer* repita la acción (**VER ILUSTRACIÓN 2.8**).

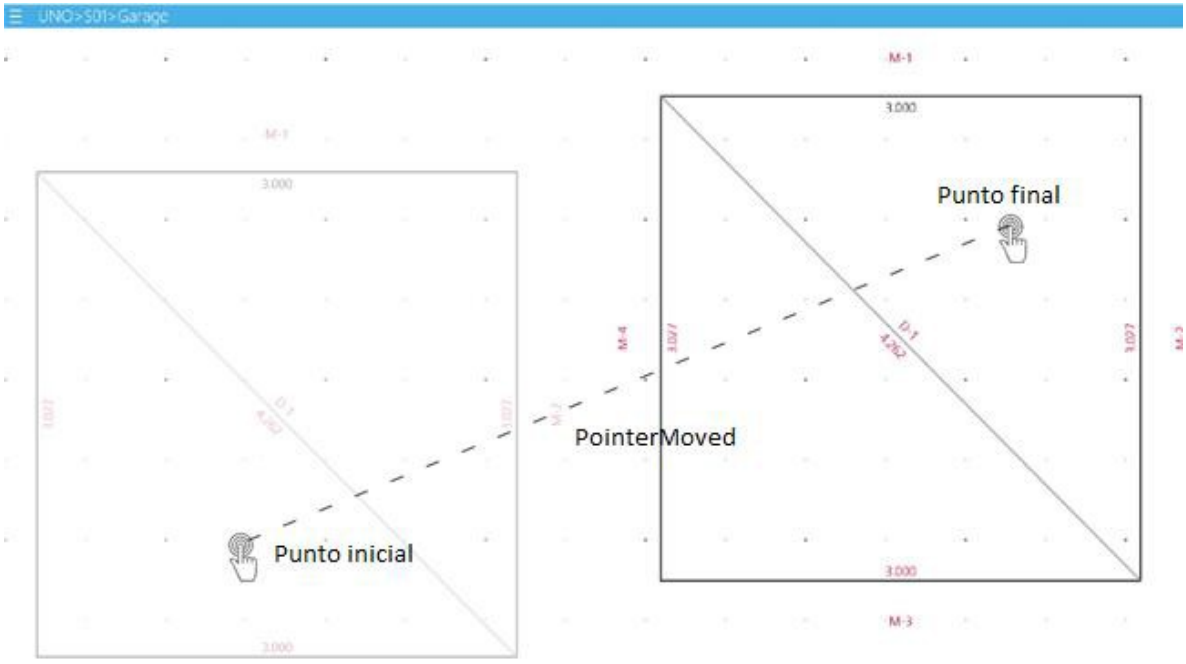


Ilustración 2.8. Desplazamiento en la aplicación

2.5. Acercamiento (Zoom)

El acercamiento requirió definir un gesto de tipo pellizco (**APÉNDICE G4**) ya que no se contaba con uno de manera nativa que actuara sobre el control *Canvas*.

2.5.1. Evento Pinch

El *pinch event* requiere que solo existan dos punteros en la pantalla los cuales se registran en el evento de *PointerMoved* del *Canvas*.

La lectura de los punteros se controló con el *CanvasTimer* y cuando la distancia entre los punteros superaba la constante *PINCH_DELTA*, se ejecutaba un evento de tipo pellizco.

$$\text{PINCH\_DELTA} = 15$$

El algoritmo fue el siguiente:

1. Se captura la posición inicial de los punteros cuando se tienen presionados dos de ellos
2. Se calcula la distancia de los punteros (Δ_0)
3. Si existe un desplazamiento de los punteros capturados y la distancia actual (Δ_{+1}) es diferente a la inicial, se ejecuta el evento de ZoomOut si el valor $\Delta_{+1} - \Delta_{}$ es mayor a 0 o el ZoomIn si es menor a 0
   La Δ_{+1} se vuelve la $\Delta_{}$ y el proceso se repite mientras se tengan dos punteros

El diagrama de flujo del algoritmo de Pinch se muestra en la ilustración 2.9:

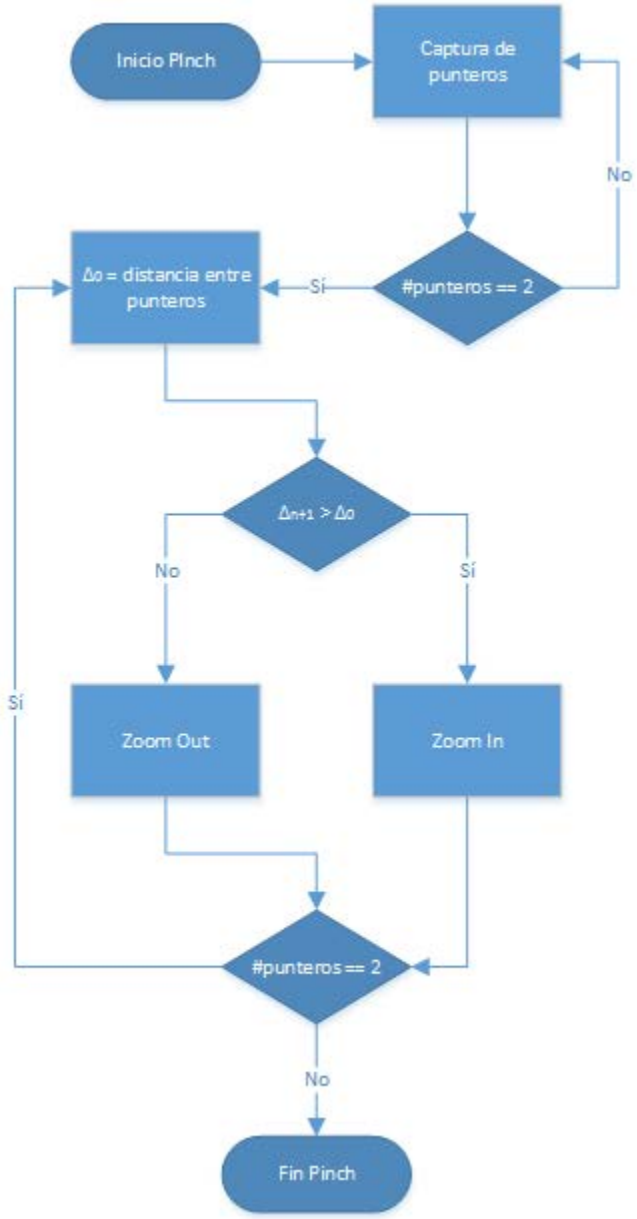


Ilustración 2.9.Diagrama de flujo del evento pinch

La aplicación tiene un factor de escalamiento que va del 16.7% hasta 200% del tamaño del puerto de vista actual. La navegación de estos factores de escala se realiza mediante el incremento o decremento de la variable global *ZoomIndex*. Esta variable es de tipo entero y tiene un valor que va desde uno hasta doce.

Se decidió contar con número finito de acercamientos para optimizar el proceso de renderizado y por qué no tenía caso encoger o agrandar tanto la figura ya que perdía definición y no se podía trabajar.

Mediante prueba y error se llegó a la conclusión de que el puerto de vista solo debía reducirse seis veces de su tamaño original y agrandarse hasta dos veces más. Las transformaciones se dan en doce pasos, dejando como valor inicial el *ZoomIndex* en seis que indica el tamaño original.

Al ejecutar el evento *ZoomIn* se decrementa la variable global *ZoomIndex* y con el *ZoomOut* se incrementa la variable.

Se calcula la matriz de transformación con el *ZoomIndex* actual. A esta matriz se le agrega un desfasamiento basado en la posición del punto medio de los punteros. (**VER MATRIZ 2.5**)

$$\text{[illegible]} = \begin{bmatrix} \frac{\text{ZoomIndex}}{6} & 0 & 0 \\ 0 & \frac{\text{ZoomIndex}}{6} & 0 \\ [\text{illegible}] & [\text{illegible}] & 1 \end{bmatrix} \quad (2.5)$$

Esta transformada se anexa a la transformada global y se actualiza la vista.

2.6. Creación de segmentos

El proceso de segmentación tiene como objetivo vectorizar la figura y generar los vértices mínimos de la geometría. La entrada de este proceso se inicia con los puntos generados en la discretización de puntos.

Un segmento se define con dos puntos consecutivos y el último segmento se forma por el punto inicial y el punto final.

2.6.1. Discretización de segmentos

Los segmentos se discretizaron definiendo un tamaño mínimo y un ángulo menor a cierta tolerancia. Además, la cantidad de segmentos varía si se usa una pluma táctil o si se toca con el dedo (Debido a la cantidad de puntos capturados por uno u otro medio). Se realizó un proceso de experimentación, el cual tenía como objetivo disminuir el número de segmentos en función al ángulo de giro entre dos puntos.

Tras analizar los distintos resultados, se concluyó que las **CONSTANTES 2.6 Y 2.7** generaban el menor número de segmentos para ambos casos de trazo.

$$\text{ANGLE\_TOLERANCE\_PEN} = \frac{\pi}{30}\text{[illegible]} \quad (2.6)$$

$$\text{ANGLE\_TOLERANCE\_TOUCH} = \frac{\pi}{22.5}\text{[illegible]} \quad (2.7)$$

Un segmento menor a 20 pixeles se considera demasiado pequeño para entrar en el rango de precisión de un levantamiento arquitectónico debido a que la escala de dibujo se definió de 100 pixeles a un metro.

Esta longitud mínima de segmento actúa como una segunda constante (**VER CONSTANTE 2.8**) en el proceso de discretización y se define de la siguiente manera:

$$\text{SEGMENT\_MIN\_LENGTH} = 20\text{pixeles} \quad (2.8)$$

El proceso de discretización realiza una iteración en la colección de segmentos. En cada iteración se toman dos segmentos, se calcula el ángulo del segmento con la dirección vectorial y se mide la magnitud del segmento (PROPIEDADES).

El segmento es válido y es agregado a la lista de segmentos si cumple las siguientes condiciones:

- Su inclinación es menor a la constante de giro
- Su longitud es mayor a la constante de longitud mínima

En caso de que la regla no se cumpla, el segmento se mezcla siempre con el anterior. La excepción es el primero que siempre se mezcla con el siguiente.

Nota: La figura mínima con la que se puede trabajar es un triángulo. Si la discretización de segmentos no genera mínimo tres, el proceso de levantamientos regresa a un estado inicial.

2.6.2. Vectorización

Este proceso se encarga de:

- Simplificar el dibujo del usuario
- Generar un aspecto ortogonalizado
- Guardar la figura con objetos de la clase *Segment2D*

Las características de una figura vectorizada son:

- Es un polígono simple
- El primer segmento siempre tiene un ángulo de 90° o 180°
- Tiene por lo menos tres lados
- La pendiente entre el segmento anterior al segmento siguiente debe ser distinta
- Ángulo mínimo de inclinación 15°

El proceso de vectorización se ejemplifica en la ilustración 2.10.

2.6.3. Validación de intersecciones

Este proceso tiene los siguientes objetivos:

- Cortar los excesos (VER ILUSTRACIÓN 2.11)
- Cerrar los segmentos en la intersección (VER ILUSTRACIÓN 2.12)
- Eliminar figuras secundarias que puedan generar un polígono complejo (VER ILUSTRACIÓN 2.13)

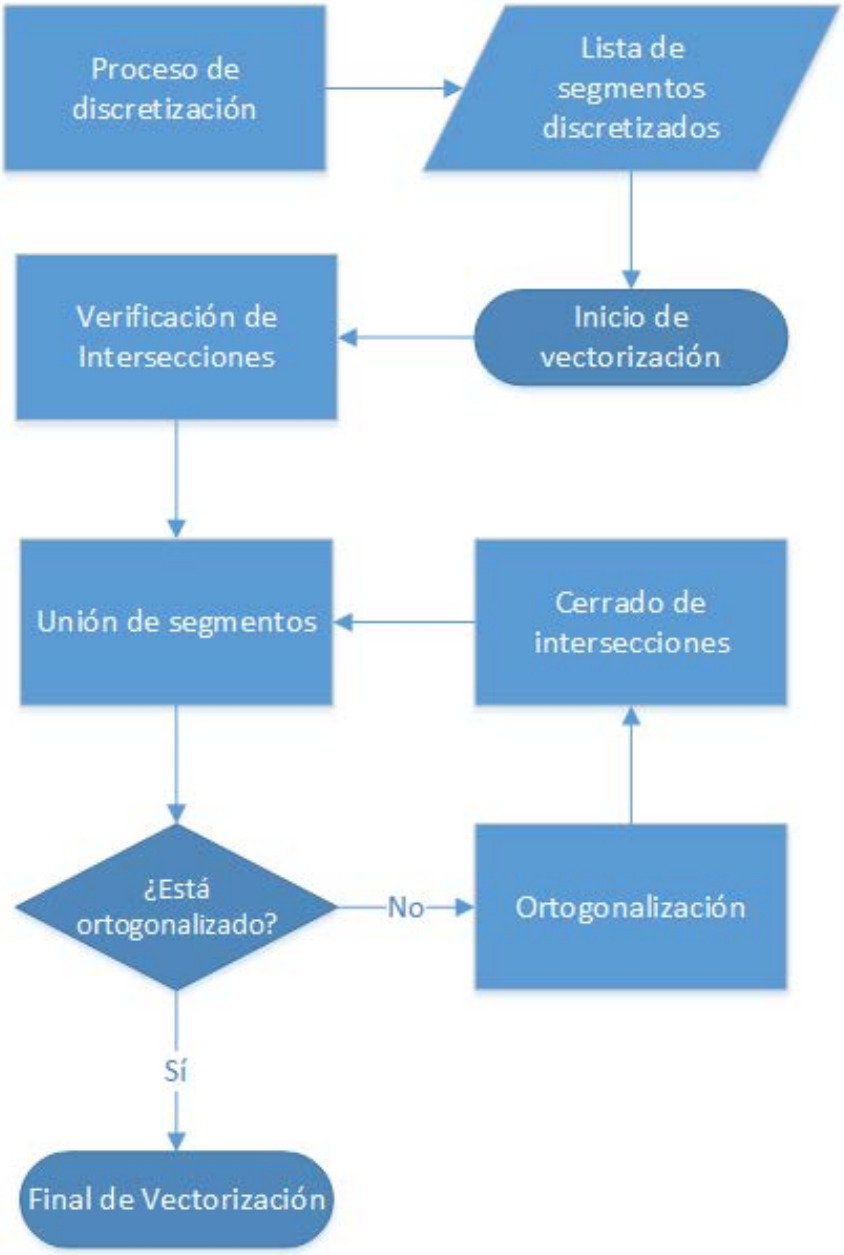


Ilustración 2.10. Diagrama del proceso de vectorización

El primer paso es verificar la intersección de cada segmento contra todos los segmentos generados. Si existe una intersección, se identifican los segmentos entre los que ocurre. Posteriormente se realiza un corte eliminando los segmentos que se encuentren antes o después de la intersección, logrando así mantener la figura original.

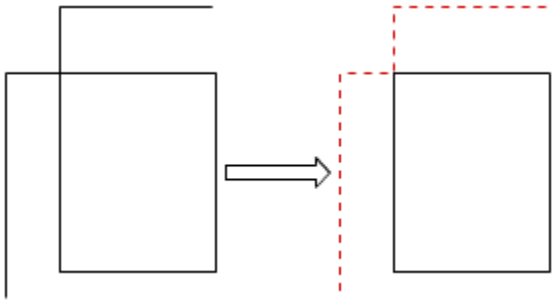


Ilustración 2.11. Eliminación de excesos del trazo

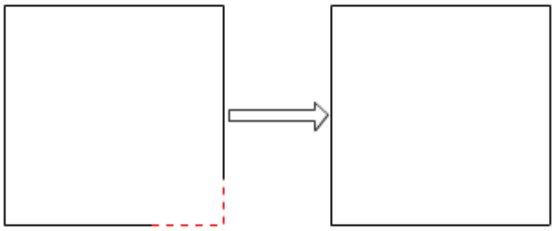


Ilustración 2.12. Cerrar los segmentos en una intersección

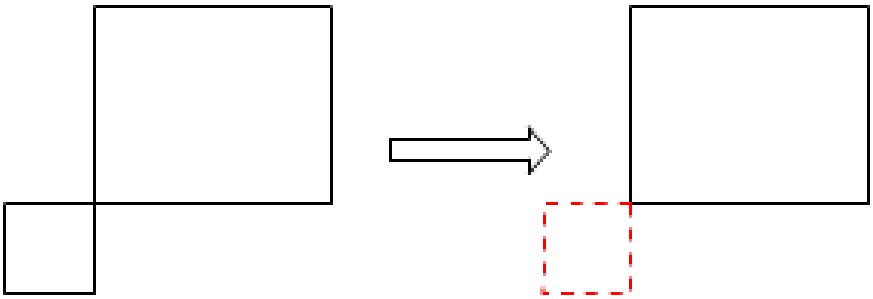


Ilustración 2.13. Conversión de polígono complejo a polígono simple

2.6.4. Unión de segmentos

Este proceso tiene como objetivo juntar los segmentos consecutivos que tengan una pendiente similar o igual. Se observó que cuando la diferencia entre los ángulos de los segmentos es muy pequeña, estos asemejan un arco. Este caso se demuestra usando un polígono regular. Sabiendo que los ángulos que van del centro a los vértices suman 360°, se buscó el polígono regular que se acercará más a una circunferencia.

El resultado fue un octodecágono (**VER ILUSTRACIÓN 2.12**).

$$\text{ANGLE\_TOLERANCE\_PEN} = \frac{\pi}{30}\text{rad} \qquad (2.9)$$

Entre las razones por las que se escogió, fue porque se tiene una buena aproximación de un arco, en grados es un número entero lo cual nos da precisión en los cálculos y en radianes es expresada como una fracción fácil de manejar.

$$\text{ANGLE\_SEGMENT\_MERGE} = \frac{\pi}{9}\text{rad} \qquad (2.10)$$

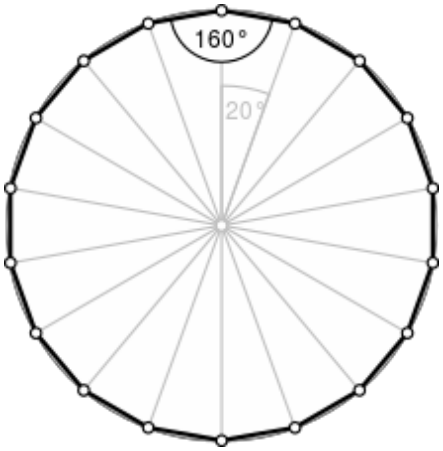


Ilustración 2.14. Octodecágono, similar a una circunferencia

Nota.

Tome en cuenta que la figura final tenderá a ser ortogonal.

El algoritmo funciona de la siguiente manera

- El primer segmento se elige como pivote, y el siguiente segmento como actual
- Se compara el pivote con el segmento actual
- Si no hay cambio en la pendiente de los segmentos el segmento actual se vuelve el segmento siguiente

Si cambia el punto final del segmento pivote se convierte el punto inicial del segmento actual. Los segmentos intermedios son eliminados y el nuevo pivote se vuelve el segmento actual y el segmento actual el siguiente hasta que el pivote no sea el último segmento.

El diagrama de flujo del algoritmo de unión de segmentos se muestra en la ILUSTRACIÓN 2.15

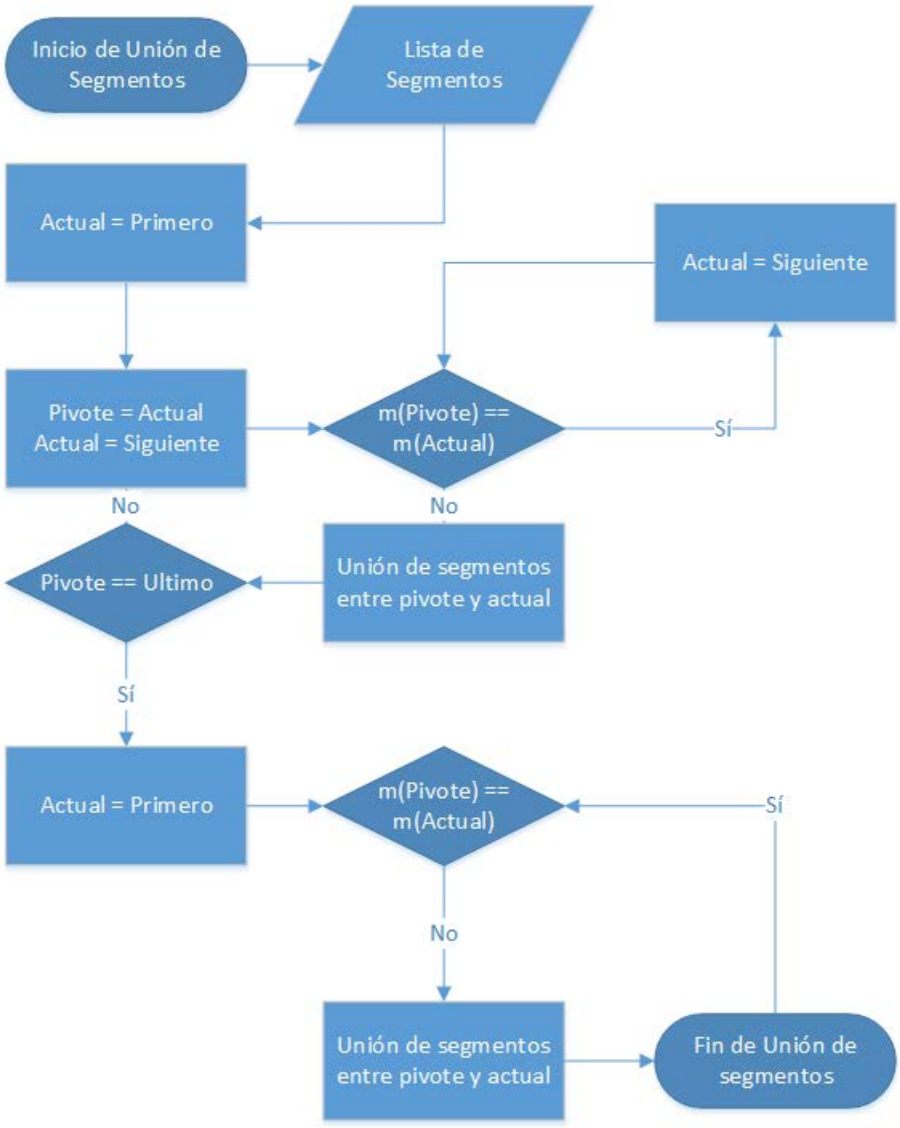


Ilustración 2.15. Diagrama del proceso de unión de segmentos

2.6.5. Cerrado de intersecciones

En este punto el proceso se encarga de cerrar los segmentos mediante un proceso de proyección, el cual requiere el método de intersección de dos rectas (APÉNDICE E4).

La distancia mínima entre dos segmentos la llamamos distancia de cerrado. Al realizar la proyección de los segmentos se requiere que la distancia de cerrado sea menor a la CONSTANTE 2.11:

$$\text{SEGMENT\_MAXIMUM\_PIXEL\_CLOSURE\_LENGTH} = 100\text{��} \qquad (2.11)$$

El algoritmo de cerrado de intersecciones funciona de la siguiente manera:

1. Recorremos los segmentos en sentido horario.
2. Validamos si el segmento actual y el segmento siguiente comparten un punto en común. Si el punto final del segmento inicial es idéntico al punto inicial del segmento siguiente, el segmento actual se vuelve el siguiente y obtenemos el siguiente segmento.
3. En caso de que no se cumpla la regla verificamos si los segmentos tienen distinto ángulo. Si tienen el mismo ángulo, los segmentos son paralelos, la figura está abierta y mandamos un error de dibujado.
4. Se realiza el proceso de proyección siempre y cuando la distancia de cerrado sea menor a la constante de cerrado. En caso de que sea mayor mandamos un error de dibujado.
5. Se repite el proceso hasta recorrer todos los segmentos. Si el segmento es el último se compara con el primero y ahí termina el proceso.

Ejemplo:

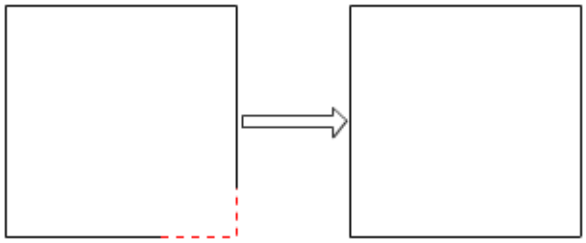


Ilustración 2.16. Los segmentos se proyectan y se cierra la figura

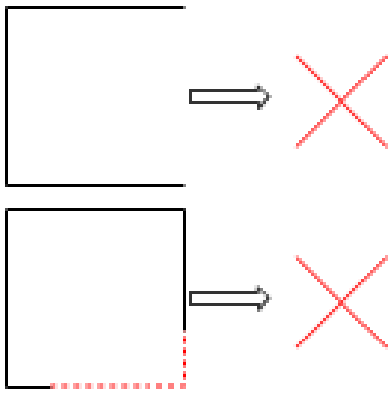


Ilustración 2.17. El primer caso ilustra segmentos paralelos y el segmento caso segmentos con una distancia mayor a la distancia mínima de cerrado. En ambos casos no se puede cerrar la figura

2.6.6. Ortogonalización

Este proceso tiene como objetivo que no existan segmentos con ángulos pequeños respecto al eje de las abscisas o al eje de las ordenadas, por lo que si no se supera la tolerancia definida (VER CONSTANTE 2.12) convierte estos segmentos en horizontales o verticales según sea el caso.

$$\text{▯▯▯▯▯▯▯\_▯▯▯▯▯\_▯▯▯▯▯\_▯▯▯▯▯▯▯▯▯▯} \quad \frac{\pi}{12}\text{▯▯▯} \qquad (2.12)$$

$$=$$

Para este proceso se utiliza el siguiente algoritmo:

1. Utilizando las diferenciales (VER ECUACIÓN 2.13) sobre las coordenadas del primer segmento se modifican las coordenadas para que este se vuelva horizontal o vertical

   Siendo:

$$\begin{aligned} \text{▯▯} &= |\text{▯}_0 - \text{▯}_1| \\ \text{▯▯} &= |\text{▯}_0 - \text{▯}_1| \end{aligned} \qquad (2.13)$$

▯▯ ▯▯ > ▯▯ ▯▯ ▯▯▯▯▯▯▯▯▯ ▯▯ ▯▯▯▯▯▯ h▯▯▯▯▯▯▯▯▯ ▯▯ ▯▯▯▯ ▯▯▯▯ ▯▯

▯▯▯▯▯▯ ▯▯▯▯▯▯▯▯

2. El segmento a analizar pasa a ser el siguiente segmento
3. Se analiza el segmento y se verifica que este dentro de la tolerancia
4. Si se encuentra dentro se transforma a horizontal o vertical según sea el caso, de lo contrario se deja tal y como esta. Al modificar el segmento actual, también se modifica el punto inicial del siguiente segmento

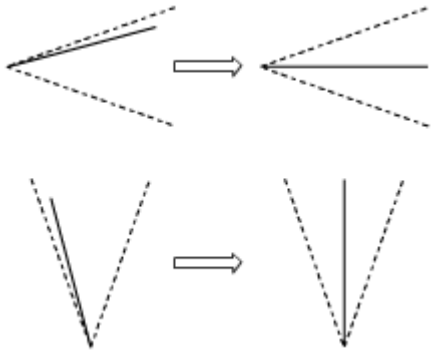


Ilustración 2.18. El ángulo mínimo es representado con las líneas punteadas. Del lado izquierdo se tiene el segmento antes de ser ortogonalizado y del lado derecho el resultado

5. Si el segmento actual no es el último, regresar a paso tres. De lo contrario se analiza de la misma forma, pero en este caso se utiliza como el punto inicial del primer segmento

2.6.7. Creación de la polilínea

La colección de segmentos se guarda en una estructura llamada *polilínea*. En la creación de la polilínea se etiquetan los segmentos con el siguiente proceso:

- El formato para nombrar los segmentos es *M-{n}*, donde n es el índice del segmento y M representa la palabra muro
- Debajo del índice del segmento se escribe la dimensión del segmento con los siguientes formatos:

- **Metros**
  Redondeado a tres decimales

1.123

- **Centímetros**
  Redondeado a un decimal

$$112.3$$

- **Milímetros**
  Sin decimales

$$1123$$

- **Pulgadas**
  Con resolución mínima de $\frac{1}{32}$ ��

$$13" \frac{5}{32}$$

- **Pies**
  La misma resolución de las pulgadas.

$$1' 1" \frac{5}{32}$$

Ilustración 2.19. Dimensión del segmento uno en metros

Se usó un código de colores para indicar el estado de la etiqueta, los estados son:

- **Inicial**
  El nombre y la dimensión en rojo
- **Dimensionado**
  La dimensión en negro y el segmento en rojo
- **Finalizado**
  El nombre y la dimensión en negro

El punto de inserción de la etiqueta por default es el punto medio del segmento (**APÉNDICE A2**). Existe la posibilidad de cambiar la posición de la etiqueta presionándola y desplazándola sobre el segmento (**VER ILUSTRACIÓN 2.20**).

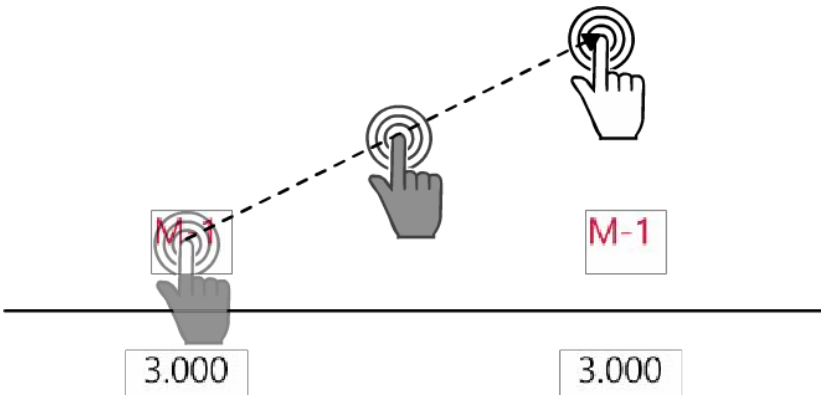


Ilustración 2.20. Desplazamiento de la etiqueta mediante una proyección vectorial

La orientación de la etiqueta se basa en aplicar una transformada de rotación usando como parámetro el ángulo del vector que define el segmento.

3. Análisis geométrico

En este capítulo se explican los algoritmos que se utilizan para procesar la geometría creada por el trazo del usuario con el fin de obtener una geometría que represente el levantamiento de una forma fiel y precisa. Este análisis consta de dos fases: Triangulación y Dimensionamiento. Cada una de estas fases tiene diferentes algoritmos que se utilizaran dependiendo del tipo de geometría requerida.

3.1. Triangulación [3]

El proceso de triangulación tiene como objetivo simplificar un polígono simple (**APÉNDICE E1**) en triángulos. El polígono debe cumplir con las siguientes características:

- El polígono es simple y cerrado
- Toda triangulación de un polígono simple con n vértices tiene n-2 triángulos
- El polígono se dibujó en sentido horario
- El segmento $\blacklozenge_0 = [\blacklozenge_0, \blacklozenge_1] \perp \text{eje}_\blacklozenge$ o $\perp \text{eje}_\blacklozenge$

El algoritmo requirió crear una clase de tipo vértice que tuviera las características de un vértice de un polígono simple (**APÉNDICE C**).

El algoritmo de triangulación se basa en tres algoritmos diferentes que se diseñaron para esta aplicación los cuales están basados en algoritmos clásicos (**APÉNDICE F**).

3.1.1. Triangulación convexa

Este método se aplica cuando todos los vértices del polígono simple P son convexos.

- Los vectores formados entre los vértices del polígono tienen un ángulo interior menor a 180°.
- El producto cruz entre el vector formado del segmento actual al segmento siguiente (**VER ILUSTRACIÓN 3.1**), siempre da como resultado una normal con dirección positiva si es medido en sentido anti horario.

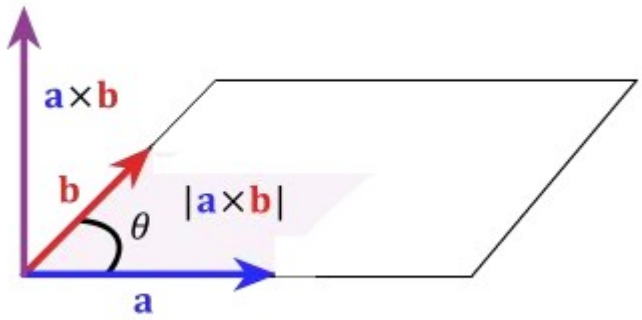


Ilustración 3.1 Producto cruz entre dos segmentos del polígono

El algoritmo se detalla a continuación (**VER ILUSTRACIÓN 3.2**):

1. Se escoge como pivote el $\blacklozenge_0$
2. Se recorren los vértices en sentido horario desde $\blacklozenge_{0+2}$ hasta $\blacklozenge_{0-2}$
3. Se trazan diagonales que van de $D_\blacklozenge = [\blacklozenge_0, \blacklozenge_\blacklozenge]$ y se crea el triángulo $\blacklozenge_\blacklozenge = [\blacklozenge_0, \blacklozenge_{\blacklozenge-1}, \blacklozenge_\blacklozenge]$

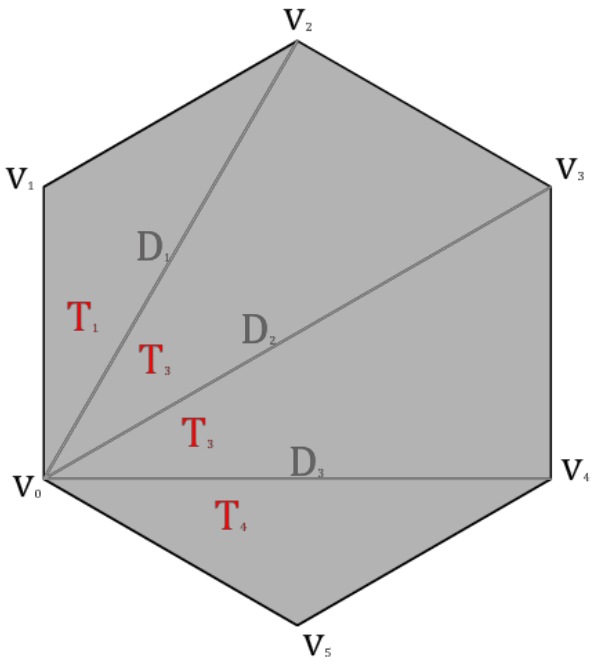


Ilustración 3.2. Triangulación de un polígono simple convexo

3.1.2. Triangulación estrella

La triangulación de estrella se utiliza cuando se cumple que los vértices del polígono se intercalan entre Convexo y Cóncavo (**VER ILUSTRACIÓN 3.4**), ya sea de la forma Convexo-Cóncavo-Convexo o Cóncavo-Convexo-Cóncavo, además de que el número de vértices convexos es el mismo al número de vértices cóncavos.

A diferencia de la triangulación convexa en esta triangulación podemos encontrar normales negativas como se muestran en la ilustración **3.3**.

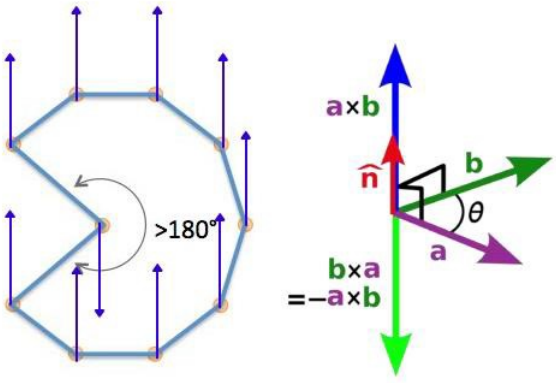


Ilustración 3.3. El sentido de la normal con base en el tipo de vértice

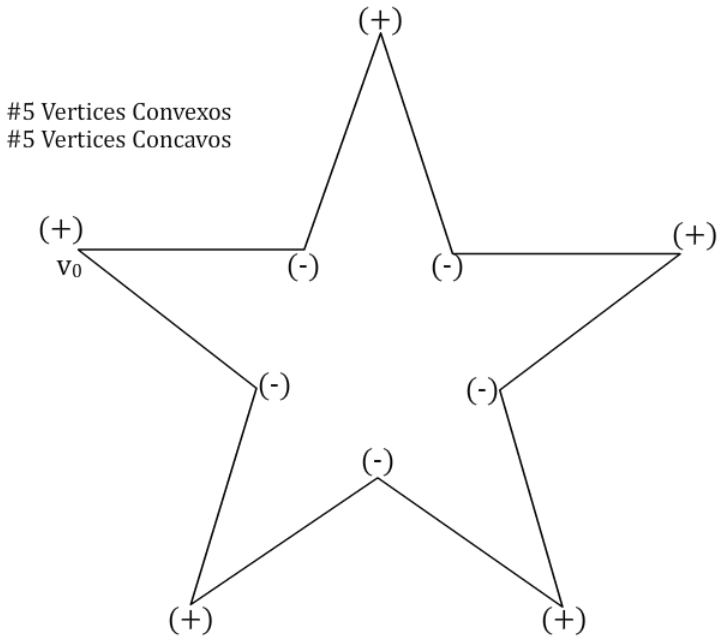


Ilustración 3.4. Análisis de normales de la estrella

El algoritmo de estrella, basado en el algoritmo *Ear Clipping* (**APÉNDICE F**) es el siguiente:

1. Se recorren los vértices en sentido horario desde v_0 hasta v_n
2. Se buscan las orejas, aquellos vértices que sean (+)
3. Se trazan diagonales que van de $D_v = [v_{v-1}, v_{v+1}]$ y se crea el triángulo $v_v \equiv [v_{v-1}, v_v, v_{v+1}]$
4. El resultado (**VER ILUSTRACIÓN 3.5**) a este proceso es un área convexa por lo cual se le aplica el algoritmo de triangulación convexa

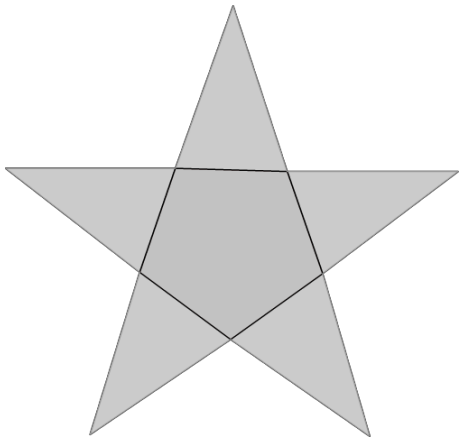


Ilustración 3.5. El resultado es un área convexa

3.1.3. Triangulación mixta

La triangulación mixta se utiliza cuando la figura no es convexa ni cumple con las condiciones de estrella (**VER ILUSTRACIÓN 3.6**), este algoritmo busca un vértice cóncavo en sentido horario.

Antes de crear un triángulo se válida si la diagonal intersecta un segmento existente. En caso de que no lo intersecte se ignora, y se busca otra boca.

El algoritmo también simplifica triángulos que estén definidos de la siguiente manera:

Cóncavo – Convexo – Convexo o Convexo – Convexo – Cóncavo

A diferencia de los demás algoritmos este no es codicioso, es decir, el proceso termina al encontrar un triángulo, ya sea hacía adelante o atrás.

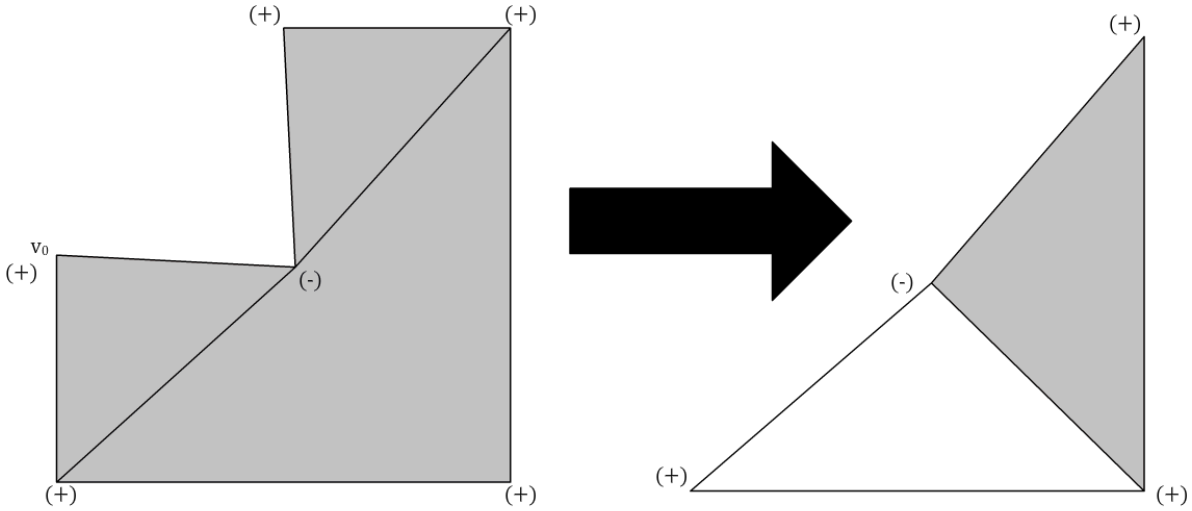


Ilustración 3.6. Ejemplo de triangulación mixta

1. Se recorren los vértices en sentido horario desde v_0 hasta v_n
2. Se buscan las bocas (aquellos vértices que sean (-))
3. Se verifica si hay un triángulo válido hacia adelante y hacia atrás
4. En caso de encontrar un triángulo hacia adelante se traza la siguiente diagonal $D_n = [v_n, v_{n+2}]$ y se crea el triángulo $v_n = [v_n, v_{n+1}, v_{n+2}]$
5. En caso de encontrar un triángulo hacia atrás se traza la siguiente diagonal $D_n = [v_n, v_{n-2}]$ y se crea el triángulo $v_n = [v_{n-2}, v_{n-1}, v_n]$

El resultado puede ser un área mixta, convexa o estrella.

3.1.4. Proceso de triangulación

Basado en los algoritmos definidos en los puntos 3.1.1 a 3.1.3 se creó un proceso que realiza

la triangulación de la siguiente manera:

1. Se crean los vértices de la figura
2. Se define el tipo de figura: convexa, estrella o mixta
3. Si es convexa se realiza la triangulación convexa y se termina el proceso
4. Si es estrella se realiza la triangulación de estrella y se va al paso 6
5. Si es mixta se realiza la triangulación mixta y se va al paso 6
6. Se reducen los vértices de la figura removiendo el segundo vértice del triángulo $\mathbf{\Delta} = [\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2]$ que forma la diagonal $\mathrm{D} = [\mathbf{v}_0, \mathbf{v}_2]$
7. Si la reducción redujo a la figura a tres vértices hemos terminado si no volvemos al paso dos

El diagrama de flujo del algoritmo de triangulación se muestra en la Ilustración 3.7

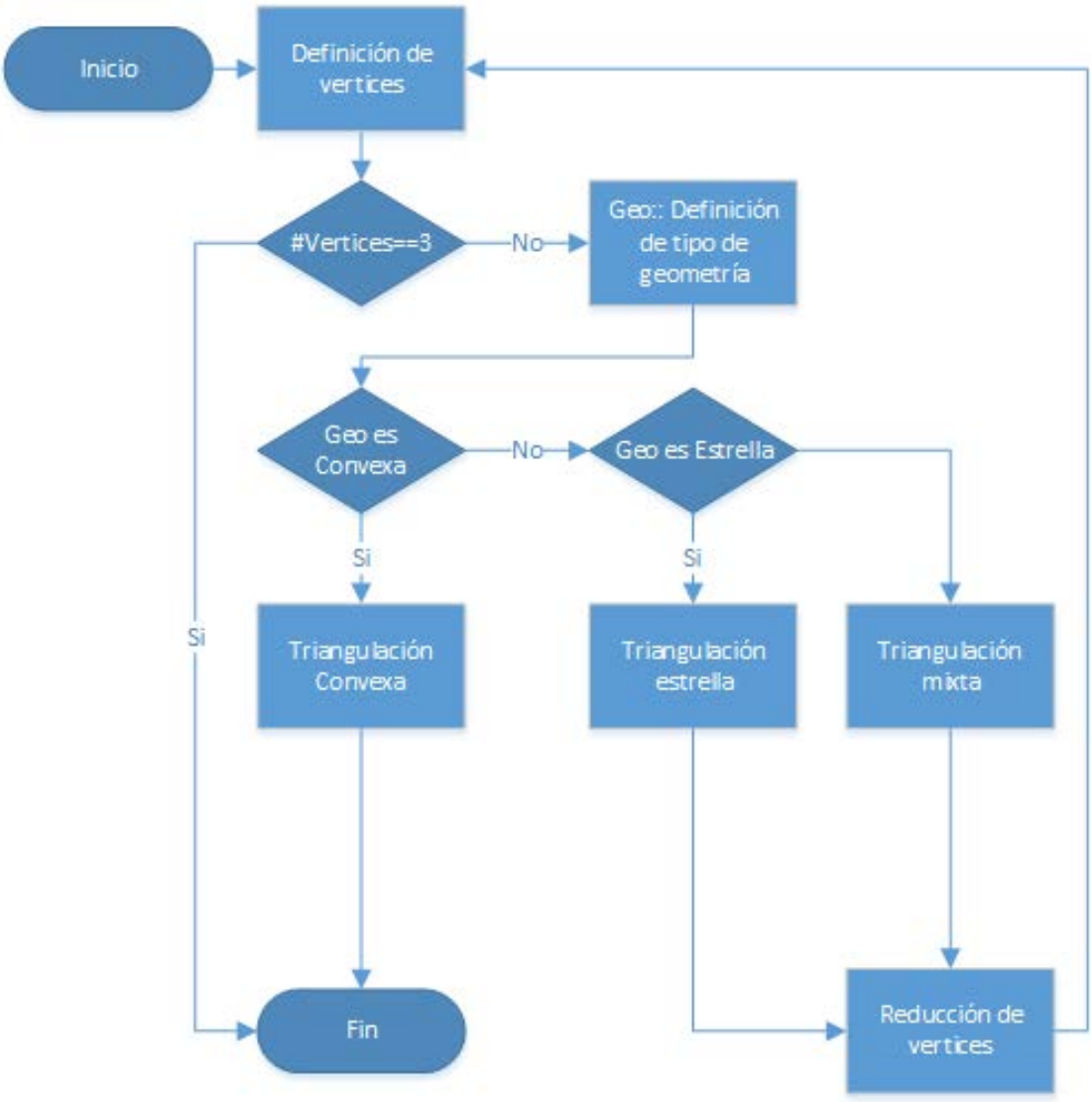


Ilustración 3.7 Diagrama de flujo del proceso de triangulación

El resultado del proceso es una lista de diagonales y una colección de triángulos que son guardados en la polilínea.

3.1.5. Intercambio de diagonales

Una *diagonal* es un segmento que divide la región interior del polígono en un conjunto de triángulos. En algunos casos la diagonal puede ser difícil de medir, para lo cual se creó la opción de intercambio de diagonal

El intercambio de diagonal solo se puede realizar en regiones convexas que sean de cuatro vértices. (VER ILUSTRACIÓN 3.8)

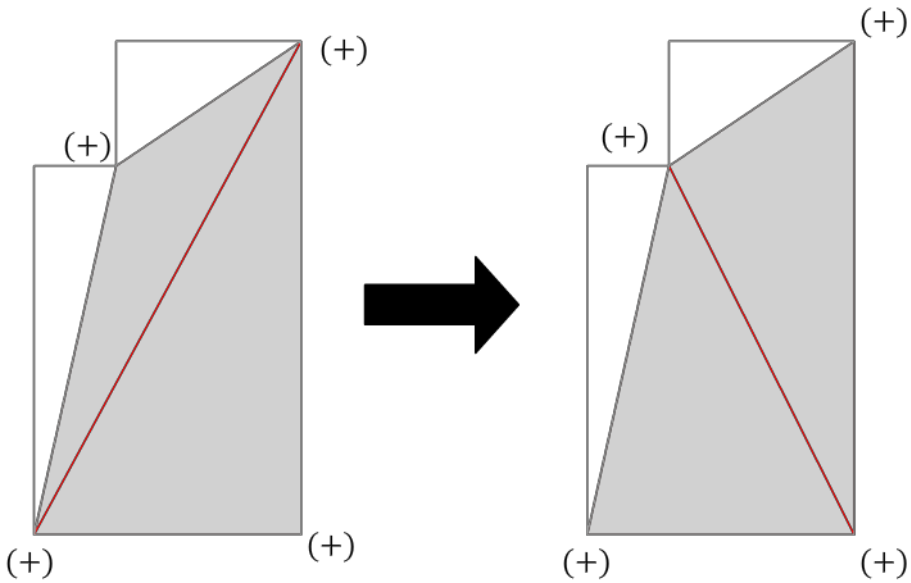


Ilustración 3.8. Intercambio de diagonal

El proceso es el siguiente:

1. Se escoge una diagonal resultante del proceso de triangulación
2. Se encuentran los dos triángulos que definen a la diagonal y se crea una figura de cuatro lados usando los vértices de los triángulos seleccionados. No se admiten vértices repetidos
3. Se verifica si la figura es convexa
4. Si es convexa la diagonal $D = [\diamond_{\diamond}, \diamond_{\diamond}]$ se cambia por la diagonal $D = [\diamond_{\diamond+1}, \diamond_{\diamond+1}]$
5. En caso de no ser convexa no se puede cambiar la diagonal porque en la boca la diagonal iría por afuera de la figura

3.2. Dimensionamiento

Este proceso tiene como objetivo usar las dimensiones reales en el trazo previamente dibujado y finalmente crea una geometría a escala que permitirá cambios de nivel, definición de segmentos curvos e inserción de vanos. A esta geometría nos referiremos como de *acabados*. Utilizando las diagonales que se crearon en el proceso de triangulación, el algoritmo elimina los errores que se pudieron dar durante el trazo a mano alzada y permite que la geometría final tenga las mismas proporciones que el espacio real. A grandes rasgos el proceso se llevará a cabo de dos fases:

- Captura de dimensiones reales
- Corrección de geometría por triángulo

El diagrama de flujo de dimensionamiento se muestra en la Ilustración **3.9**

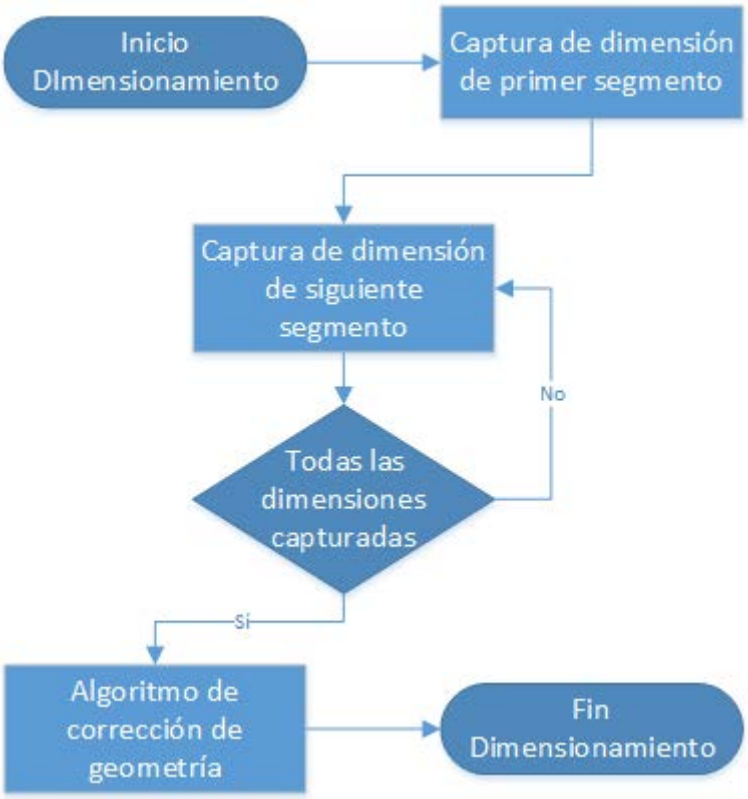


Ilustración 3.9. Diagrama de flujo de dimensionamiento

3.2.1. Captura de dimensiones reales

Para la realización de esta fase utilizaremos las etiquetas que se generaron en el proceso de dimensionamiento. Originalmente las etiquetas tendrán los textos en color rojo. Al presionar alguna de ellas, la aplicación permitirá editar la dimensión de la etiqueta y una vez que se presione en cualquier parte de la pantalla fuera de la etiqueta se guardará la dimensión introducida y el texto de la dimensión será de color negro. (**VER ILUSTRACIÓN 3.10**)

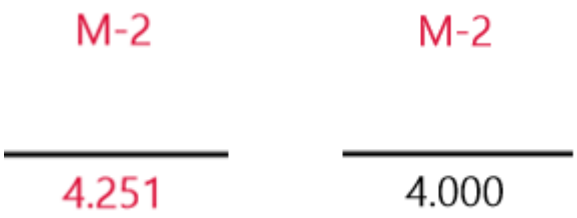


Ilustración 3.10. Cambio de estado en la etiqueta

Todas las etiquetas muestran este cambio al modificar la dimensión original, pero en el caso de la etiqueta correspondiente al primer segmento (etiquetado siempre como M-1) también se escala la figura modificando todas las dimensiones de la geometría y finalmente un acercamiento que permita que toda la figura se pueda visualizar dentro de la pantalla (**VER ILUSTRACIÓN 3.11**). Debido a esto se recomienda que la primera dimensión que se edite sea esta, de lo contrario se perderán las dimensiones introducidas y se tendría que repetir el proceso de introducción de dimensiones.

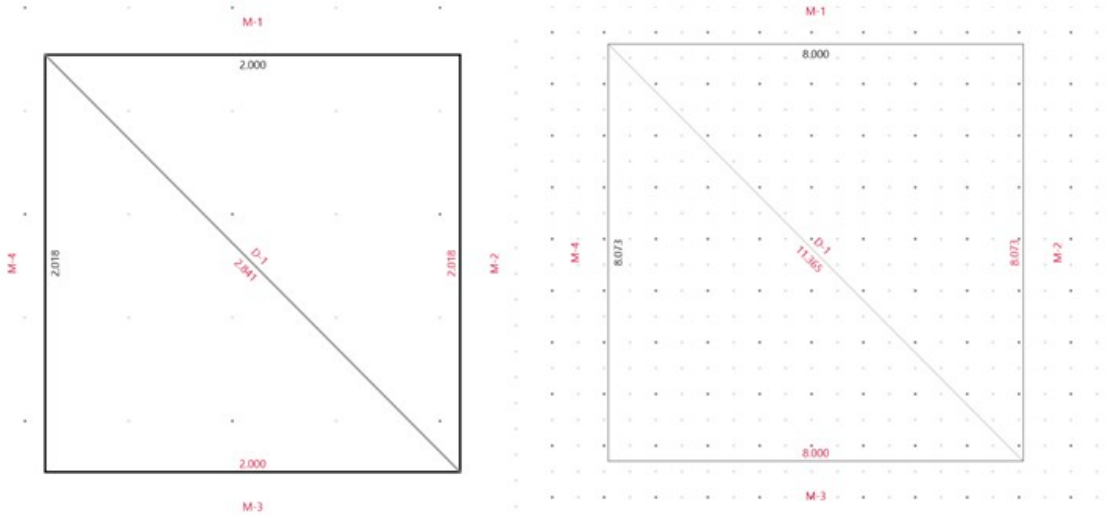


Ilustración 3.11. Escalamiento al modificar el primer segmento

Finalmente, cuando todas las dimensiones hayan sido editadas, incluyendo las diagonales, seguiremos a la siguiente fase donde se corregirá la figura y terminará el proceso de dimensionamiento.

3.2.2. Corrección de geometría por triángulo

Cuando se hayan definido todas las dimensiones se inicia la corrección de geometría. Para esto, utilizaremos los triángulos que se generaron en el proceso de triangulación y corregiremos la figura triángulo por triángulo hasta tener todos los puntos de la geometría en la posición adecuada.

Cada uno de los triángulos se corregirá de la siguiente manera:

- Se fija un segmento del triángulo al cual llamamos base

- Se generan dos ecuaciones de circunferencia donde los radios son las dimensiones definidas por el usuario y el centro es cada uno de los puntos extremos del segmento base
- Se calculan las intersecciones de ambas ecuaciones
- Se elige la intersección que mejor se adecue a la figura original y se utiliza como punto para los dos segmentos sobrantes

Un ejemplo se muestra en la ilustración **3.12**

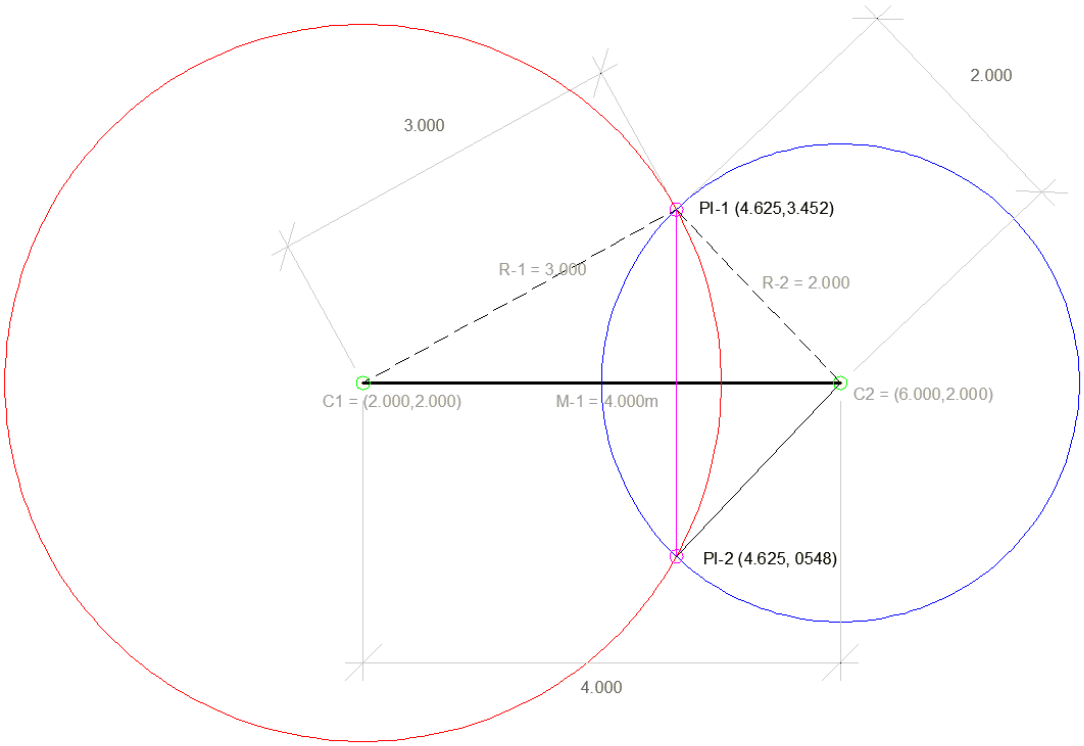


Ilustración 3.12. Intersección de segmento mediante trazos de circunferencias

Ejemplo:

Para un triángulo de dimensiones 4, 3 y 2, donde el segmento de dimensión 4 será el segmento base, utilizamos la ecuación general de la circunferencia (**VER ECUACIÓN 3.1**) [3]:

$$(\text{□} - h)^2 + (\text{□} - \text{□})^2 = \text{□}^2 \qquad (3.1)$$

Donde *r* es el radio y la coordenada *(h, k)* es el centro de la circunferencia. Por lo tanto, las ecuaciones de las dos circunferencias son **3.2.1** y **3.2.2**:

$$(\text{□} - 2)^2 + (\text{□} - 2)^2 \equiv 3^2 \qquad (3.2.1)$$

$$(\text{□} - 6)^2 + (\text{□} - 2)^2 \equiv 2^2 \qquad (3.2.2)$$

Resolviendo 3.2.1 y 3.2.2:

$$\text{□}^2 - 4\,\text{□} + 4 + \text{□}^2 - 4\text{□} + 4 = 9 \qquad (3.3.1)$$

$$\text{□}^2 - 12\,\text{□} + 36 + \text{□}^2 - 4\text{□} + 4 = 4 \qquad (3.3.2)$$

Agrupando términos en las ecuaciones 3.3.1 y 3.3.2:

$$\text{□}^2 + \text{□}^2 - 4\,\text{□} - 4\text{□} - 1 = 0 \qquad (3.4.1)$$

$$\text{□}^2 + \text{□}^2 - 12\text{□} - 4\text{□} + 36 = 0 \qquad (3.4.2)$$

Igualando las ecuaciones 3.4.1 y 3.4.2:

$$\text{□}^2 + \text{□}^2 - 4\,\text{□} - 4\text{□} - 1 = \text{□}^2 + \text{□}^2 - 12\text{□} - 4\text{□} +$$

$$36$$

$$8\text{□} - 37 = 0$$

$$\text{□} = \frac{37}{8} = 4.625 \qquad (3.5)$$

Sustituyendo 3.5 en la ecuación 3.2.1:

$$(4.625 - 2)^2 + (\text{□} - 2)^2 = 3^2 \qquad (3.6)$$

Resolviendo la ecuación 3.6:

$$6.890625 + \text{□}^2 - 4\text{□} + 4 = 9 \qquad (3.7)$$

$$\text{□}^2 - 4\text{□} + 1.890625 = 0$$

$$\text{□} = \frac{-(-4) \pm \sqrt{(-4)^2 - 4(1)(1.890625)}}{2(1)}$$

$$\text{□} = \frac{4 \pm 2.90473}{2}$$

$$\text{□} = 3.45236 \qquad (3.7.1)$$

$$\text{□} = 2.54763 \qquad (3.7.2)$$

Al obtener la *y* de ambas intersecciones, tenemos los puntos donde las dimensiones definidas por el usuario son válidas permitiendo que el ángulo sea correcto.

Este algoritmo se aplica a cada triángulo generado en el proceso de triangulación utilizando como base el primer segmento que se dibujó. (VER ILUSTRACIÓN 3.13)

Este algoritmo consiste en lo siguiente:

1. Se toman como puntos fijos los puntos correspondientes al primer segmento de la figura, con lo cual el triángulo que contiene este segmento es el único que tiene dos puntos resueltos. Esto asegura que siempre comencemos el proceso con este triángulo y que los triángulos adyacentes al mismo tengan un punto resuelto al menos.
2. Se ordena la lista de triángulos usando como criterio la cantidad de puntos resueltos en orden decreciente.
3. Se utiliza el algoritmo de corrección de triángulo para mover el punto faltante del primer triángulo que no tenga todos sus puntos resueltos al lugar correcto. Este proceso modifica todos los triángulos que contienen este punto.

4. Para seleccionar cuál de los dos puntos se utiliza, se hace una comparación con los ángulos del triángulo previo a la corrección, escogiendo el que tenga una diferencia menor.
5. Se calcula el desplazamiento que sufrió el punto y se le aplica al resto de la figura.
6. Se repite desde el segundo paso hasta que todos los puntos hayan sido resueltos.

Ilustración 3.13. Diagrama de dimensionamiento

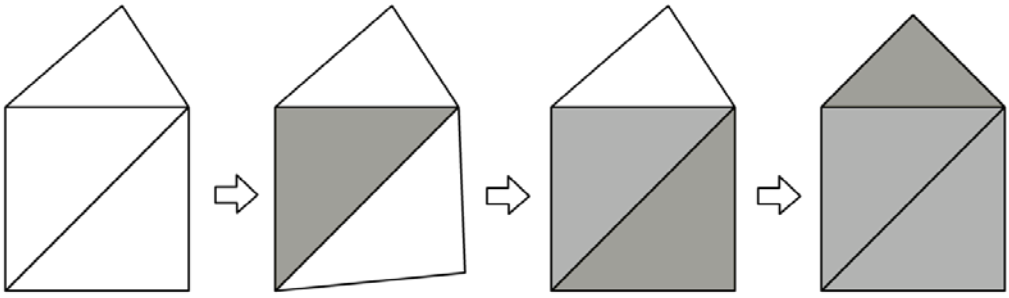


Ilustración 3.14. Proceso de dimensionamiento. Observe como cambia la geometría en el proceso

4. Diseño de la geometría

La geometría de la aplicación se define a partir del trazo del usuario, se divide en capas que manipula el usuario y se maneja en dos vistas.

La primera vista es la de planta, a esta geometría se llamó *Geometría Acabado*, la cual contiene la vista de vanos, puertas, ventanas, simbología y arcos dibujados por el usuario.

La segunda vista es la de alzado, a esta geometría se llamó *Geometría Alzado*, en esta vista se puede cambiar las dimensiones de muro, personalizarlo ya sea agregándole molduras o insertándole ventanas y puertas.

4.1. Geometría acabado

Al tener dimensionado el trazo, se crea una segunda geometría la cual tiene las dimensiones reales, pero no puede ser dimensionada. A diferencia de la geometría de trazo, a esta se le pueden definir curvas y será llamada *AcabadoPolyline2D*.

La geometría de acabado se define usando la geometría de trazo en forma vectorial. Se comienza a dibujar tomando como base el primer punto del segmento (**VER ILUSTRACIÓN 4.1**), de ahí utilizando coordenadas polares calculamos el siguiente punto. (**VER ECUACIONES 4.1 Y 4.2**) [3]

$$\text{[illegible]} = \text{[illegible]} + \text{[illegible]} \cos \Theta \qquad (4.1)$$

$$\text{[illegible]} = \text{[illegible]} + \text{[illegible]} \sin \Theta \qquad (4.2)$$

Donde la distancia d es el valor que ingreso el usuario en el proceso de dimensionamiento y el ángulo Θ es el ángulo del vector del segmento de trazo dimensionado.

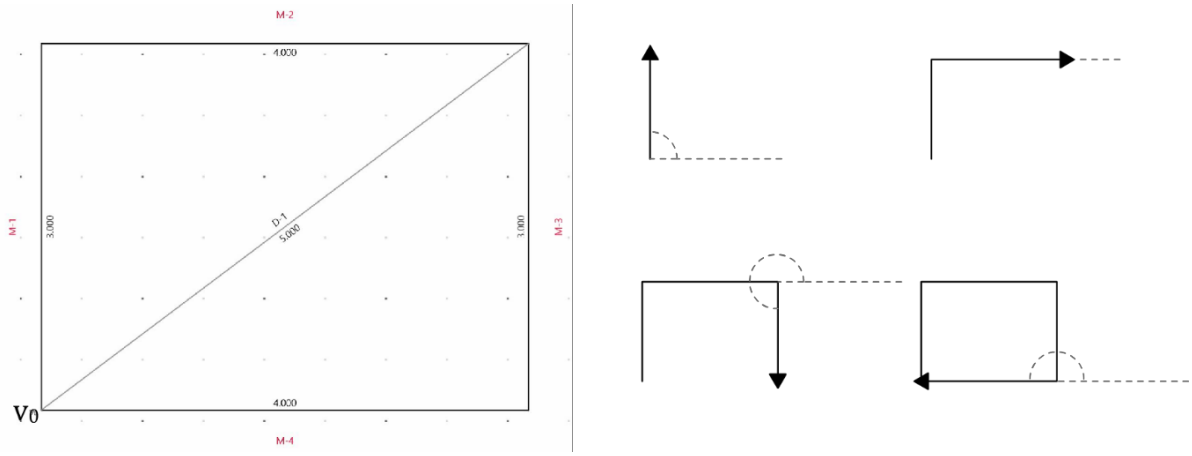


Ilustración 4.1. Dibujado de geometría de acabado. La geometría inicia en [illegible] ([illegible], [illegible]) y sigue la secuencia indicada por los vectores del lado derecho de la imagen

Con respecto a esto, la geometría solo necesita un punto inicial y una colección de muros. Un muro es un segmento de la polilínea y tiene las siguientes características:

- Punto final
- Magnitud (longitud de muro)
- Ángulo
- Vista de muro
- Etiqueta de muro
- Etiqueta de área

La etiqueta de muro permite realizar las siguientes acciones de muros:

- Convertir un muro recto a un muro con arco
- Convertir un vértice de muro en un filete
- Editar su geometría de alzado

Un muro también es un contenedor y puede guardar la definición de una puerta, una ventana o un espacio vacío llamado vano. En la **ILUSTRACIÓN 4.2** se muestra la jerarquía de muros:

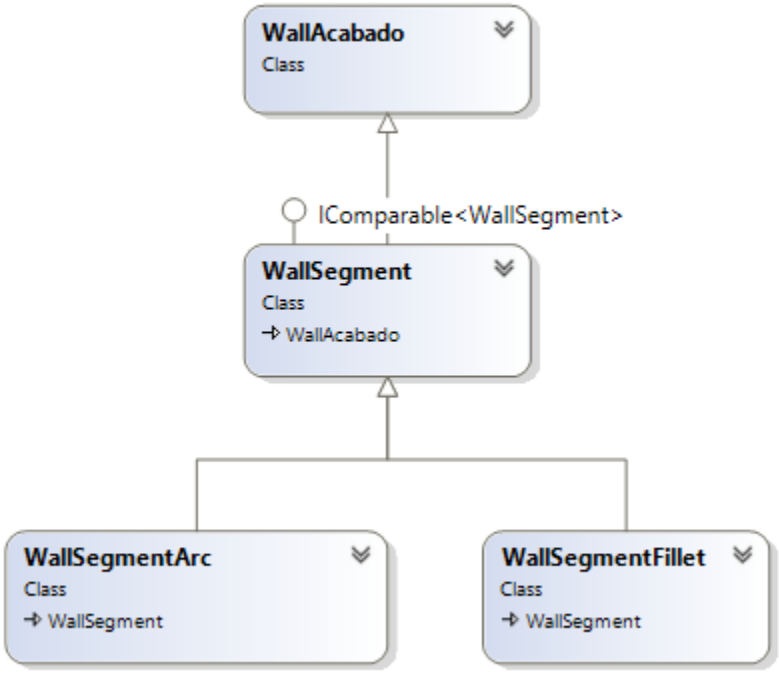


Ilustración 4.2. Diagrama de muros

4.2. Muro de arco

En la aplicación, un arco será definido como media circunferencia y dibujado en sentido de las manecillas del reloj. La geometría se define en el (**APÉNDICE E9**).

Para crear un muro de arco (*ArcSegment*) solo es necesario seleccionar el muro recto (*LineSegment*) y ejecutar el método de *LineSegment*.*CreateArc*(). El ángulo de este arco será siempre de 180° y su área se suma al área generada en el proceso de trazo.

4.3. Muro de filete

El muro de filete es un arco definido por tres puntos y a diferencia del arco anterior que solo requiere dos.

Se define a partir de la geometría de acabado y su ángulo puede ser agudo u obtuso, pero siempre menor a 180°.

Su ángulo es calculado con respecto al radio que ingresa el usuario de la siguiente forma:

1. Se seleccionan dos muros contiguos de la geometría de acabado
2. Se proporciona la longitud del radio

3. Se realiza una validación para saber si el filete es posible. La primera condición es que la longitud de los segmentos sea mayor al valor del filete
4. Se calcula el ángulo del vértice B con los vectores creados del muro anterior y del muro siguiente

La ilustración **4.3** describe los componentes del filete:

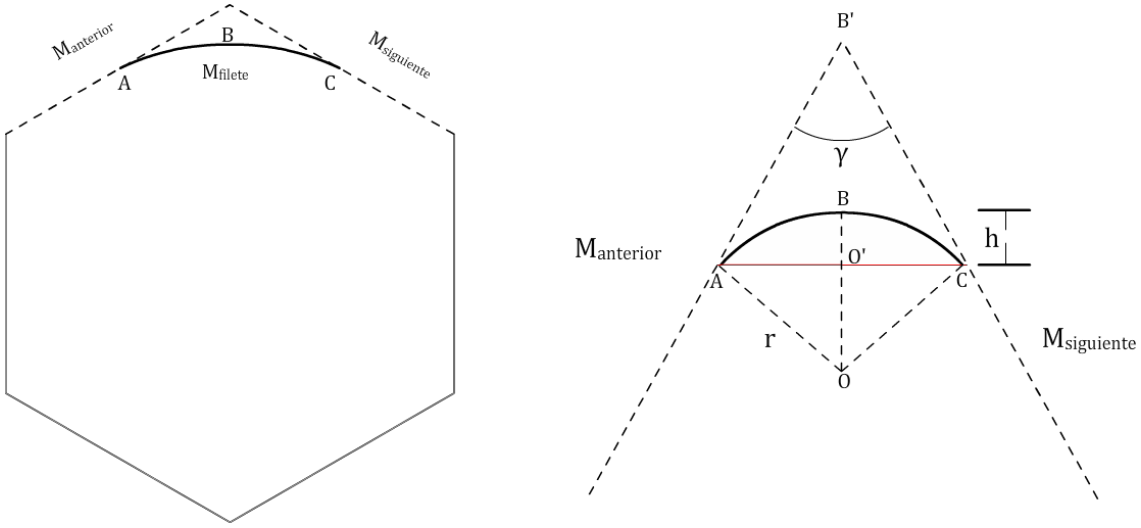


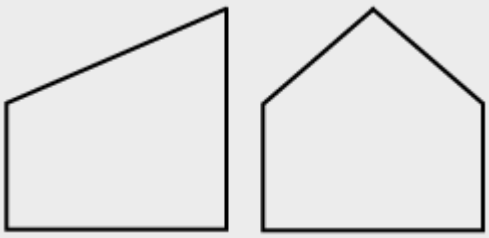
Ilustración 4.3. Diagrama de arco de filete

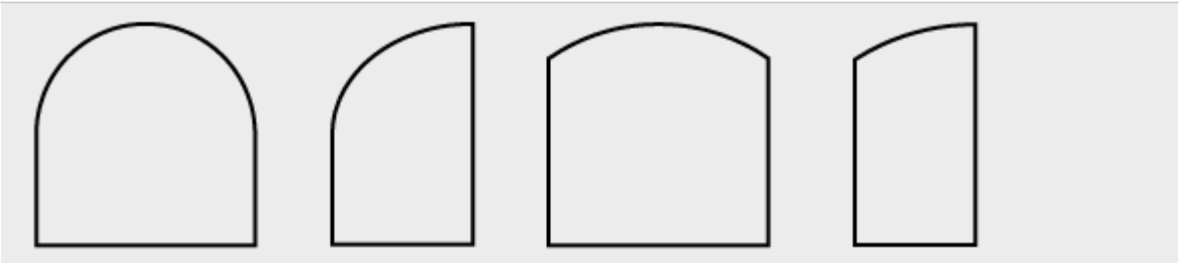
4.4. Geometría de alzado [4]

La geometría de alzado hace referencia a la vista de elevación del muro (*fachada*).

La aplicación define un catálogo de las fachadas más usadas las cuales se muestran en la tabla **4.1**:

Muro Sencillo

Muros Inclinados

Muros con arcos



Muros compuestos

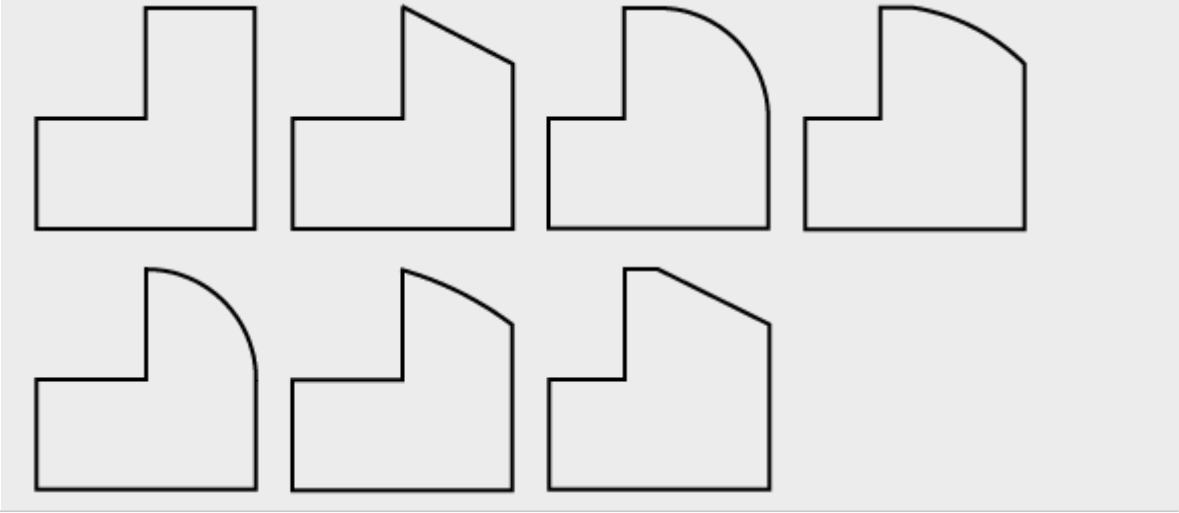


Tabla 4.1 Fachadas utilizadas en la aplicación

Para dibujar los muros de arcos existen dos casos:

- **Caso 1**
  Se usa media circunferencia
- **Caso 2**
  Se usa un arco de tres puntos

El muro de dos aguas y los muros compuestos se forman con dos niveles, por lo que la altura total incluye la altura del entrepiso. Esta altura será discutida en el siguiente capítulo.

4.4.1. Zoclos y molduras

Los zoclos o molduras son piezas que pueden ser colocadas en la base de los tabiques de los muros, en el intermedio del muro, o en la parte superior. Su función principal es decorativa, pero algunas sirven para proteger los muros de golpes o roces.

La aplicación define tres tipos de molduras (VER ILUSTRACIÓN 4.4):

- **Zoclos**

Definen las molduras inferiores que siguen la trayectoria de la base del muro. Su parámetro de definición es el ancho de moldura.

- **Moldura Intermedia**

Definen las molduras que se encuentran desfasadas a cierta distancia de la base y no superan el siguiente nivel del muro. También siguen la trayectoria de la base. Sus parámetros de definición son el ancho y la altura a piso.

- **Moldura superior**

Definen las molduras que se encuentran desfasadas a cierta distancia del nivel superior. A diferencia de las otras molduras siguen la trayectoria del techo, el cual, en el caso de los muros compuestos solo afecta al piso superior.

Todas las molduras tienen como parámetro adicional el espesor, el cual, a diferencia de las alturas no se verá reflejado gráficamente, pero tendrá importancia al momento de cuantificar los materiales necesarios del levantamiento.

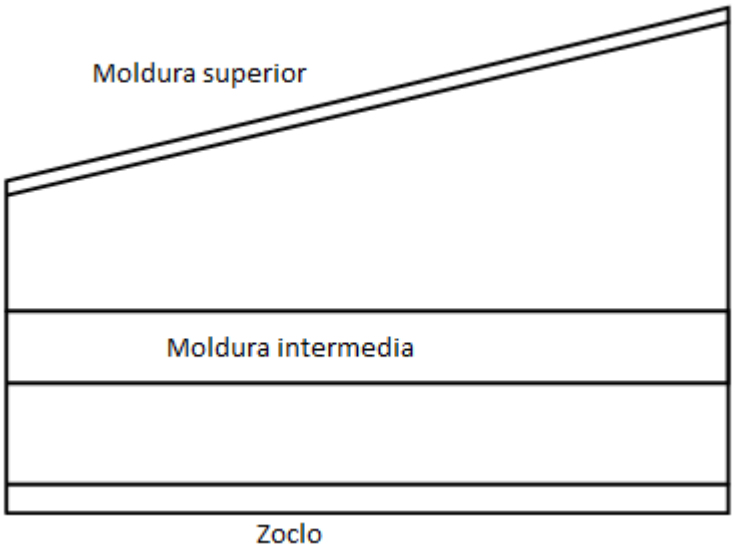


Ilustración 4.4. Molduras en muro inclinado

4.4.2. Dimensiones

El principal objetivo de la vista de elevación del muro es poder definir las mediciones reales de los muros, y al mismo tiempo, con el uso de una vista previa a escala, verificar si las mediciones son correctas. Con este fin se crearon las dimensiones o cotas, las cuales permiten al usuario ingresar las mediciones por medio del teclado o utilizando algún dispositivo compatible.

En la vista del alzado las dimensiones se componen de una parte gráfica y una parte lógica.

4.4.2.1. *Dimensión gráfica*

La dimensión gráfica se compone de una ilustración en forma de cota que se define entre dos puntos y puede ser insertada de manera recta en cualquier segmento del muro con un desfase (VER ILUSTRACIÓN 4.5).

```
<Ctrl_Dimension Lenght="80" Direction="Horizontal" Canvas.Left="x" Canvas.Top="y" />
```

Ilustración 4.5. Dimensión horizontal

Además de la ilustración se le asocia un *TextBox* que se ubica en el centro de la dimensión y contiene la medida real introducida por el usuario. En esta parte de la dimensión hay de 2 tipos:

- **Fijas**

  Son dimensiones que se definen por parámetros del proyecto o del cuarto correspondiente, por lo cual el usuario no las puede modificar
- **Libres**

  Estas dimensiones pueden modificarse libremente por el usuario

4.4.2.2. *Dimensión lógica*

En la parte lógica las dimensiones tienen una jerarquía, la cual define si afecta o no a otras dimensiones. Debido a esta jerarquía una dimensión solo puede modificar las dimensiones que se encuentren en el mismo nivel o en un nivel inferior, pero no pueden afectar a la dimensión padre.

Debido a esta regla hay dimensiones de dos tipos:

- **Dependientes**

  Son aquellas que dependen de otra dimensión, por lo tanto, tienen ciertas restricciones evitando que tengan valores muy elevados o nulos. Para la aplicación solo se permite una dependencia directa entre dos dimensiones
- **Independientes**

  Son aquellas que se encuentran en la jerarquía superior y por lo tanto no tienen ningún límite, pero afectan a las demás dimensiones

Como se mencionó la dependencia de los muros se basa en una estructura de árbol. Para los muros compuestos se llega a tener más de dos hijos por nodo y las cotas siempre tienen codependencia en pares, por lo que fue necesario crear un nodo agrupador el cual no entra como dimensión, pero incrementa el nivel de jerarquía de la dimensión.

Como regla adicional, una dimensión que se encuentre al mismo nivel de un grupo no podrá alterar las mediciones de los hijos del grupo, mientras que el nodo padre del grupo si podrá cambiar las mediciones de las dimensiones hijas del nodo grupo (VER ILUSTRACIÓN 4.6).

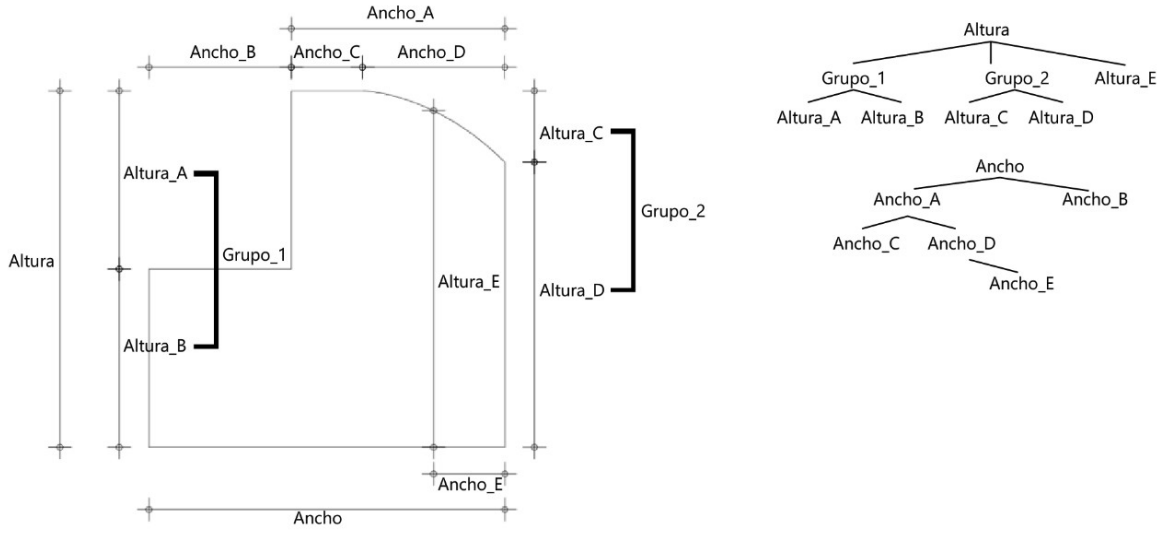


Ilustración 4.6. Diagrama de jerarquías de dimensiones

4.4.3. Notas y fotografías

Para el proceso de documentación se creó la funcionalidad de registrar notas que estuvieran referenciadas al muro.

Para las fotografías se creó un directorio en la carpeta de imágenes del usuario con el formato *Proyecto>Nivel>Espacio>Muro*. La información de notas se guardó en el nodo de *Geometry_Alzado* el cual presenta la estructura mostrada en la ilustración **4.7**:

```
<Geometry_Alzado>
<WallViews>
  <WallView Name="M-1" Type="0" Area="3.83295">
    <Moldura Type="0" Altura="0.1" Espesor="0.025" Apiso="0.9" Enabled="true" />
    <Moldura Type="1" Altura="0.1" Espesor="0.025" Apiso="0.9" Enabled="true" />
    <Moldura Type="2" Altura="0.1" Espesor="0.025" Apiso="0.9" Enabled="true" />
    <Dimensions>
      <Dimension Name="DimWidth" Value="2" />
      ...
      <Dimension Name="DimWidthE" Value="0.445" />
    </Dimensions>
    <Notes>
      <Note Name=".\2015-001 - UNO\Sótano 1\Garage\M-1\Carga.jpg" Content="Muro de Carga" />
    </Notes>
  </WallView>
</WallViews>
</Geometry_Alzado>
```

Ilustración 4.7. Nodo XML de geometría de alzado

4.4.4. Vanos, puertas y ventanas [4]

Para completar la vista de alzado se requirió implementar una herramienta capaz de insertar y editar vanos, los cuales serán definidos como cualquier abertura en un muro. Estos pueden permanecer vacíos o estar destinados a una puerta o ventana. Además, debido a la naturaleza de la aplicación estos elementos permiten crear las conexiones entre diferentes espacios y con esto poder generar una vista de planta la cual englobe todos los espacios que se encuentran dentro de un mismo nivel.

Esta herramienta funciona de manera similar a la edición del muro y se dividió en puertas y ventanas por facilidad. Se incluyeron las principales fachadas que podría tener una puerta para limitar las posibles dimensiones y así disminuir los errores durante el proceso (VER TABLAS 4.2 Y 4.3).

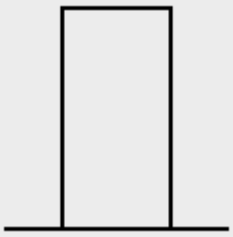
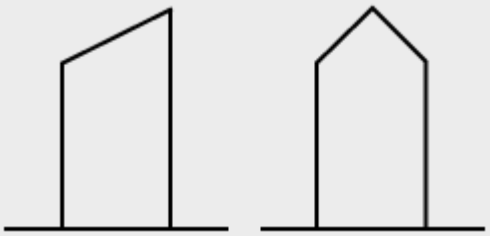
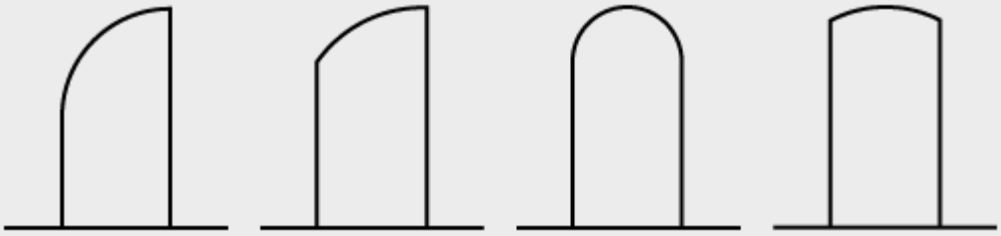
Puertas sencillas

Puertas inclinadas

Puertas con arco


Tabla 4.2 Fachadas de puertas

Ventanas sencillos

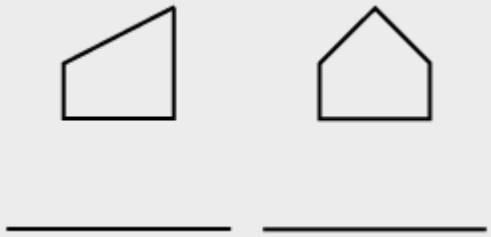
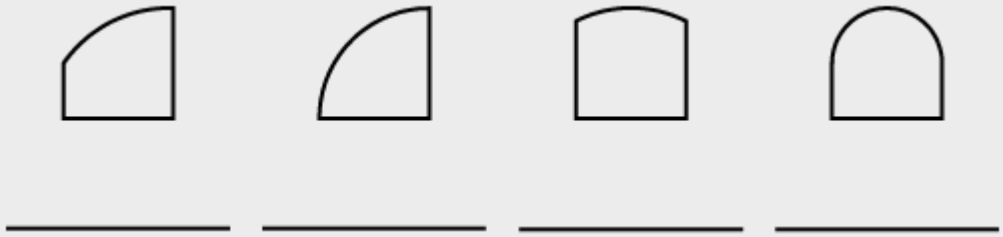
Ventanas inclinadas

Ventanas con arco


Tabla 4.3 Fachadas de ventanas

Al igual que los muros, utilizan las dimensiones para definirlas. Adicionalmente, dentro de este proceso se definirán las siguientes características:

- **Tipo de vano**

  Define si el vano se encuentra vacío o si contiene una puerta o ventana. Este se verá reflejado en la vista de planta insertando el bloque correspondiente en el sentido que la apertura y el abatimiento se hayan definido
- **Espesor de muro**

  Define el espesor del muro ya que la geometría de trazo corresponde a las dimensiones interiores del cuarto
- **Materiales**

  Define los materiales de la puerta o ventana en caso de que no esté vacío. En el caso de la puerta también se define el marco
- **Abatimiento**

  Define si la puerta abre hacia afuera o hacia adentro
- **Apertura**

  Define si la puerta abre del lado izquierdo o derecho

- **Fijo superior**
  En el caso de las puertas, define si contiene un bloque fijo en la parte superior y su altura
- **Espacio contiguo**
  Define el espacio que conecta el vano con el espacio actual
- **Puerta/Ventana existente**
  Define la puerta o ventana en caso de que se haya definido en el espacio contiguo. De ser así, automáticamente copia los datos de la ventana previamente definida

Una vez que se regresa a la vista de planta, toda la información de la puerta o ventana se guarda en el XML utilizando la estructura mostrada en las ILUSTRACIONES 4.8 Y 4.9.

```
<Doors>
  <Door Name="P-1" Type="1" Area="1.8900000000000001" VanoType="Puerta"
    Abatimiento="Interior" Apertura="Izquierdo" AcabadoType="-1" OwnerWall="M-2"
    Flip="1" AdjacentRoom="Garage" RelatedVano="P-1">
    <Dimensions>
      <Dimension Name="DimHeight" Value="2.3" />
      ...
      <Dimension Name="DimThickness" Value="0.18" />
    </Dimensions>
    <Materiales>
      <Material Clave="Madera tambor" Concept="" Value="1.89" UnitType="m2" Type="6" />
      <Material Clave="Estándar" Concept="" Value="1.89" UnitType="m2" Type="5" />
    </Materiales>
  </Door>
</Doors>
```

Ilustración 4.8. Nodo XML de dimensiones de puertas

```
<Windows>
  <Window Name="V-1" Type="1" Area="1.08" VanoType="Ventana"
    AcabadoType="-1" OwnerWall="M-3" WindowType="Abatible"
    Flip="1" AdjacentRoom="" RelatedVano="">
    <Dimensions>
      <Dimension Name="DimVanoWidth" Value="0.9" />
      ...
      <Dimension Name="DimThickness" Value="0.18" />
    </Dimensions>
  </Window>
</Windows>
```

Ilustración 4.9. Nodo XML de dimensiones de ventanas

4.5. Vanos [4]

Los vanos se definen en el área de la geometría de alzado, pero son visibles en la vista de planta. En la ilustración **4.10** se muestra el diagrama de flujo para la inserción de un vano.

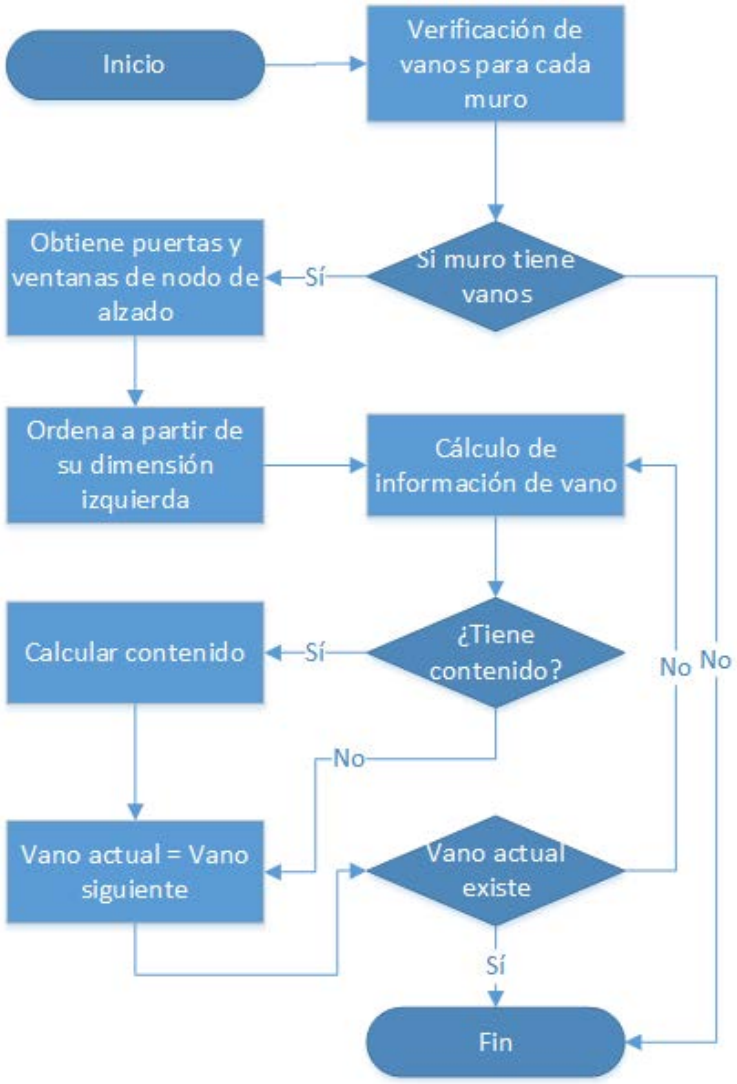


Ilustración 4.10. Diagrama de inserción de vanos

Para trabajar con los vanos se definió una estructura que es guardada en un segmento de muro con las siguientes propiedades (**VER ILUSTRACIONES 4.11 Y 4.12**):

- Dimensión izquierda
- Dimensión de ancho de vano
- Dimensión derecho
- Dimensión de ancho de muro
- Contenido
- Rotación
- Punto de inserción

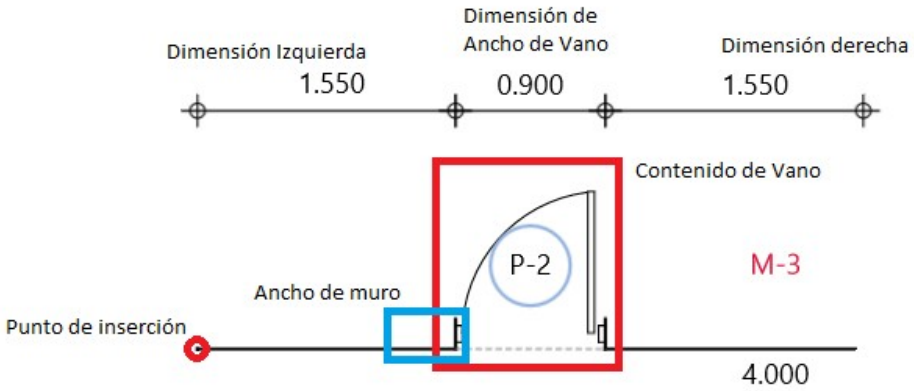


Ilustración 4.11. Definición de vanos

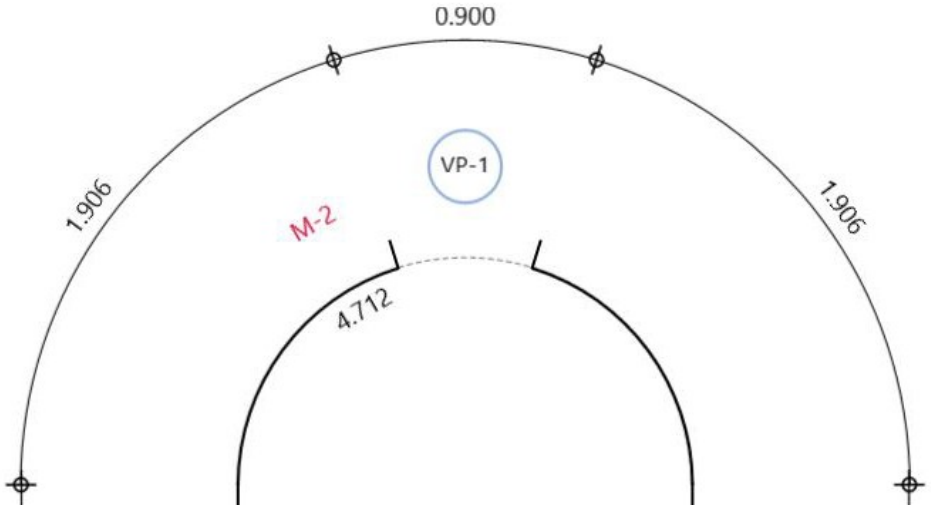


Ilustración 4.12. Definición de vanos en arco

Además, cumple las siguientes reglas:

- Los vanos se guardan en la colección de vanos del elemento segmento de muro
- Un muro tiene n-vanos
- La suma de todas las dimensiones de los vanos del muro es igual a la longitud del muro
- El contenido del vano es opcional
- El cerramiento del muro se define de la siguiente manera:
    - Si el vano es de piso a techo, no hay cerramiento, por lo que no se dibuja ninguna línea de cerramiento
    - Si el vano es de piso y hay una diferencia a techo, se utiliza la línea puntada para dibujar el cerramiento

- Si el vano tiene una diferencia del piso y después llega al techo, se dibuja una línea continua
- Los vanos con dos aguas se dibujan con cerramiento inclinado
- Cualquier otro caso siempre se dibuja una línea continua

- El vano se inserta siempre a ceros grados y se rota desde su punto de origen a la dirección del vector del segmento de muro formado con el punto inicial al punto final

Ejemplos de tipo de cerramientos (**VER ILUSTRACIONES 4.13-4.16**):

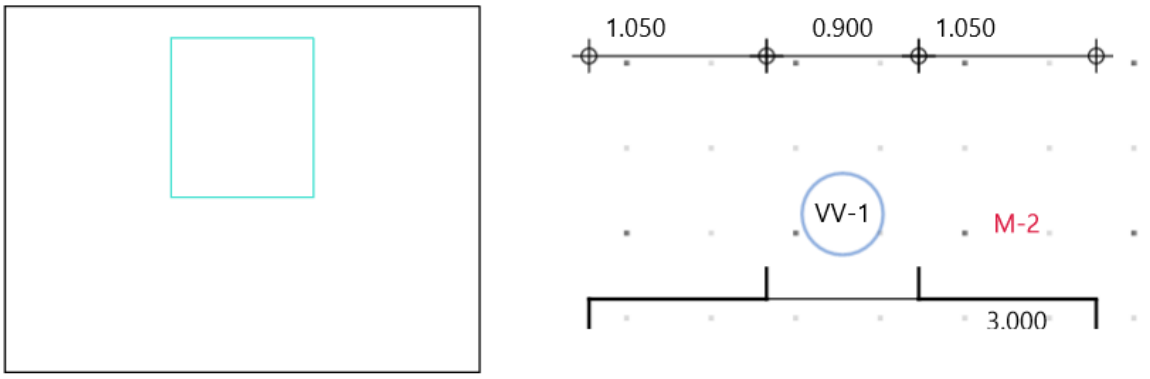


Ilustración 4.13. Línea continua para vano de tipo ventana

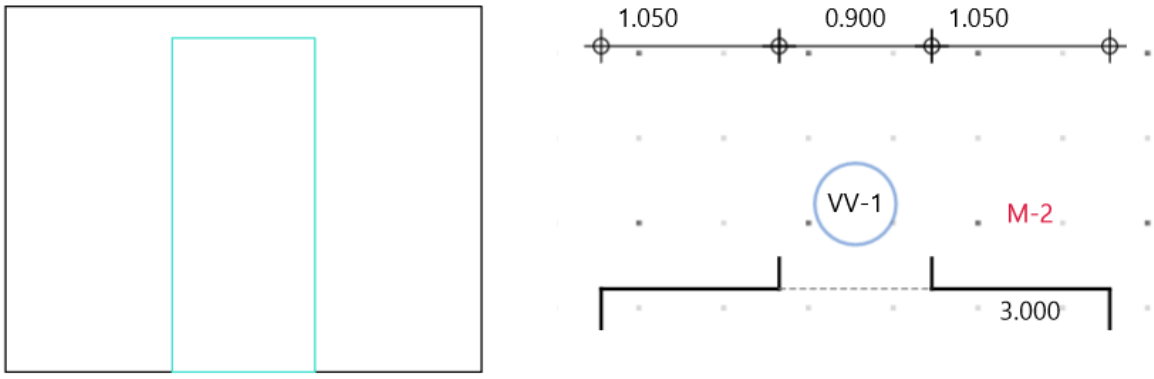


Ilustración 4.14. Línea punteada para vano tipo puerta

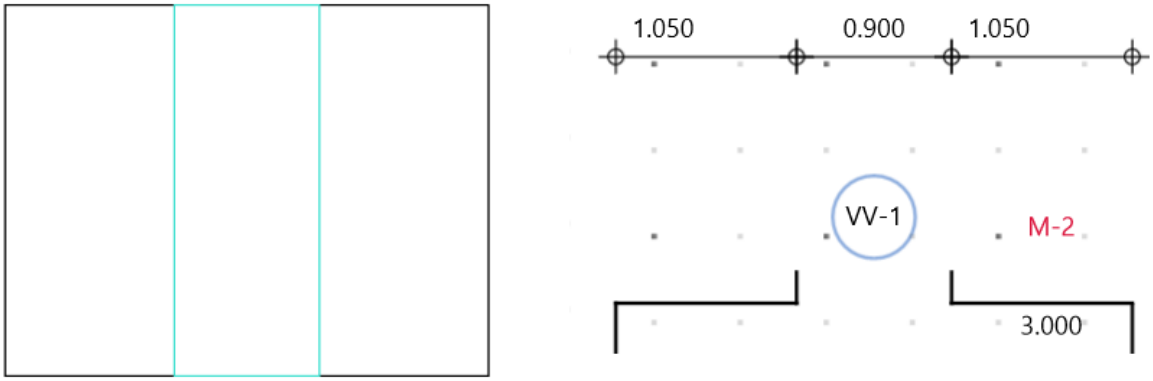


Ilustración 4.15. Sin línea para vano de piso a techo

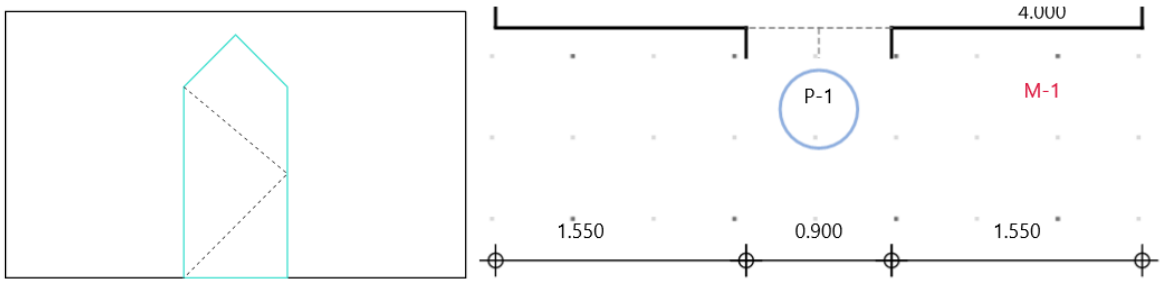


Ilustración 4.16. Vano con cerramiento inclinado

4.5.1. Vanos en arcos

Para los arcos se generó un caso especial ya que las distancias cambian conforme se alejan del centro de la circunferencia. Debido a esto, las dimensiones relacionadas con el vano se definieron en función de un ángulo. Este se calcula dividiendo la longitud del segmento entre el radio del arco.

Ejemplo:

En la ilustración **4.17** tenemos un medio círculo que tiene una longitud de arco de 4.712 m, la dimensión izquierda mide 1.906 m al igual que la derecha y la dimensión de ancho de vano es de 0.9 m y un radio de 1.5 m.

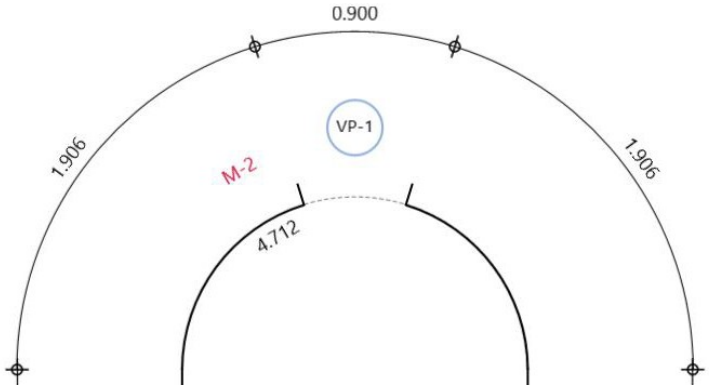


Ilustración 4.17. Vano de arco en medio circulo

Los ángulos de los segmentos se definen como:

$$\{\frac{1.906}{1.5}\text{°},\frac{0.9}{1.5}\text{°},\frac{1.906}{1.5}\text{°}\}$$

$$\{72.80°, 34.39°, 72.80°\}$$

Los puntos de los arcos de las dimensiones se calculan utilizando coordenadas polares y los ángulos previamente obtenidos.

4.5.2. Nomenclatura de vanos, puertas y ventanas

La etiqueta se inserta en el centro del vano y según el tipo insertado se tiene la siguiente nomenclatura:

Nomenclatura	Descripción
V	Ventana
P	Puerta
VR	Reja en ventana
PR	Reja en puerta
VV	Vano en ventana
PV	Vano en puerta

Tabla 4.4 Nomenclatura de puertas y ventanas

4.6. Cálculos de áreas

4.6.1. Área de trazo

El área del trazo se define como la suma del área de los triángulos generados en el proceso de triangulación. Para calcular el área del triángulo se usó el área de Herón (**APÉNDICE D4**).

4.6.2. Área de acabado

El área de acabado en un estado inicial es igual a la de trazo, pero se ve afectada por la inserción de arcos y filetes, por lo que el Área de Acabado se define en la **ECUACIÓN 4.3**.

$$\text{Área de acabado} = \text{Área de trazo} + \sum \text{Área de filetes} + \sum \text{Área de arcos} \qquad (4.3)$$

El área de un filete es positiva cuando la orientación del filete es en sentido contrario de las manecillas del reloj y positivo en el otro caso. El área se define en la **ECUACIÓN 4.4**.

$$\text{Área de filete} = \text{radio}^2 \times \left(\tan\left(\frac{\alpha}{2}\right)\right) \qquad (4.4)$$

Donde α es el ángulo del filete.

Finalmente, el área del arco siempre es positiva y es un semicírculo por lo que el área se define utilizando la **ECUACIÓN 4.5**.

$$\text{Área de arco} = \frac{\pi \times \text{radio}^2}{2} \qquad (4.5)$$

4.6.3. Área de piso y área de plafón

Estas áreas son iguales al área de acabado en un principio y sufren modificaciones al insertar vanos en el espacio o modificar el desnivel del piso o del plafón.

Al definir los niveles, se definió un *NPT* y un entrepiso por nivel, por lo que todos los espacios tienen el piso definido por el *NPT*, mientras que los plafones son definidos por la suma del *NPT* y el entrepiso. Se puede modificar las alturas del piso y del plafón para cada espacio, para lo cual se le pedirá al usuario que defina el desnivel respecto al original, afectando al área de estos.

Al definir un vano, se obtiene un ancho de muro y este genera una abertura en un muro del espacio. Dicha abertura genera una línea que es conocida como cerramiento la cual delimita el espacio del muro.

El área de vano se calcula utilizando la ECUACIÓN 4.6.

$$\text{Área}_{\text{vano}} = \text{Ancho}_{\text{vano}} * \text{Altura}_{\text{vano}} \quad (4.6)$$

Existen tres casos para cambiar el área del espacio con vanos como se muestra en las ilustraciones 4.18 y 4.19:

- Cuando existe un desnivel positivo respecto al espacio contiguo se suma el área completa del vano al espacio actual
- Cuando no existe desnivel, solo se suma la mitad del área a cada espacio
- Cuando existe un desnivel negativo respecto al espacio contiguo no se suma nada ya que el área completa del vano se ha agregado al espacio contiguo

Debido a la naturaleza de los vanos, no todos los vanos pueden modificar las áreas. En el caso del piso solo lo modificaran los vanos que lleguen a piso, mientras que para el plafón solo se verá afectado por los vanos que lleguen a techo.

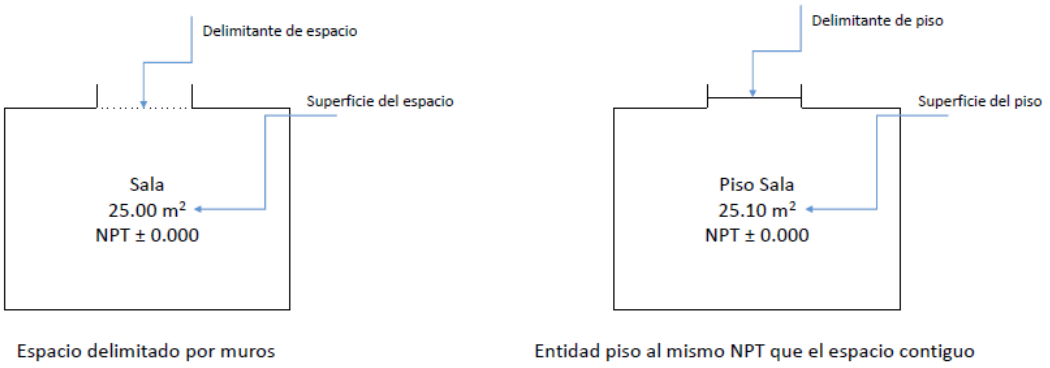


Ilustración 4.18. Área de piso cuando no hay cambio en el NPT

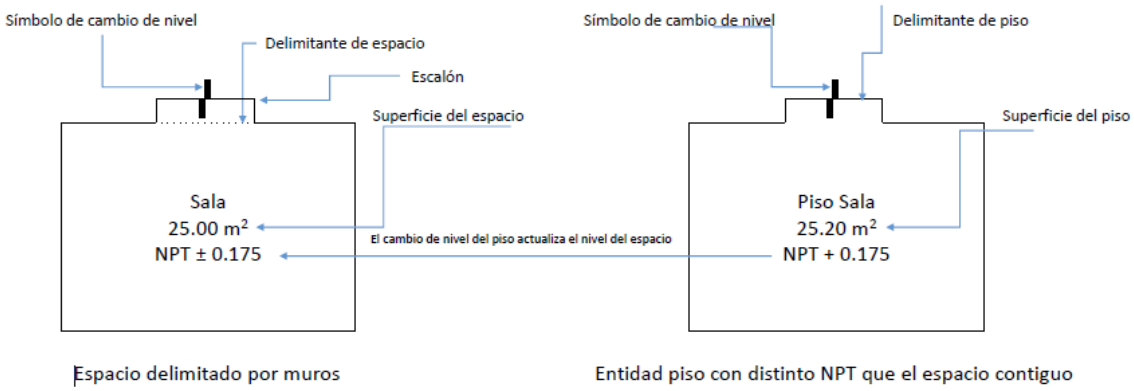


Ilustración 4.19. Área de piso cuando hay cambio positivo en el NPT

5. Modelo de la aplicación

La aplicación maneja toda la información almacenada de forma jerárquica, siendo el proyecto la estructura con mayor jerarquía. Esto permite que todos los dibujos realizados puedan agruparse y en conjunto forman un levantamiento arquitectónico completo.
Adicionalmente la aplicación cuenta con plantillas las cuales son estructuras independientes al proyecto, pero de menor importancia. Estas podrían ser consideradas como catálogos que permiten la definición de varias características de los proyectos.

5.1. Estructura del proyecto

Como previamente se mencionó, el proyecto es la estructura con mayor jerarquía y representa el levantamiento arquitectónico en su totalidad el cual está constituido por una serie de levantamientos parciales, así como la configuración, los datos geográficos y los datos del cliente.

A cada proyecto se le asignará una clave única que estará compuesta por el año en que se empezó el proyecto y un contador que se reiniciara cada año. El proyecto contiene los siguientes datos:

- **Datos generales**
  Incluye información general del inmueble, tales como el estado actual y su ubicación
- **Datos del cliente**
  Contiene datos de contacto y si los hubiera, datos de facturación
- **Materiales**
  Se refiere a los materiales que se usarán en la mayoría del levantamiento, facilitando la asignación de estos en procesos posteriores
- **Niveles**
  Es la siguiente estructura en jerarquía y equivaldrá a un piso o sótano
- **Imágenes**
  El proyecto almacena las imágenes asociadas al proyecto, entre las cuales se encuentran el mapa de la localización y la foto
- **Colaboradores**
  Esta sección define los contactos de la aplicación con estilo de agenda

Para algunos datos de proyecto se requería agregar una lista con los valores disponible. Estos valores se modificaban constantemente por lo que se optó por incluirlos dentro del archivo XML para facilitar la edición de estos (VER ILUSTRACIÓN 5.1).

El proyecto se define con los siguientes parámetros:

- **Nombre**
  Definido por el usuario y describe de que es el proyecto
- **ID**
  La clave del proyecto es única y se define con el año seguido por un guion y un índice autoincrementado con formato de tres dígitos, por ejemplo: 2015-001
- **Alta**
  La fecha de creación del proyecto
- **Inicio**
  La fecha cuando se inicia el levantamiento

- **Status**

  Define el estado del proyecto y los posibles valores son:
  - Nuevo
  - Activo
  - En proceso
  - Suspendido
  - Pospuesto
  - Cancelado
  - Terminado
  - En revisión

5.1.1. Datos generales de proyecto

Estos datos son únicamente informativos y facilitarán la localización del levantamiento arquitectónico. Entre estos datos se incluyen:

- Nombre del cliente
- Tipo de proyecto y obra
- Define el tipo de levantamiento y se podrá usar como filtro en caso de existir demasiados proyectos
- Dirección del levantamiento
- Una foto o imagen asociada al levantamiento
- Datos de contacto del encargado de proyecto
- Ubicación exacta por medio de coordenadas
- Mapa con la ubicación del levantamiento

5.1.2. Datos generales del inmueble

Estos datos definen el comportamiento principal del proyecto ya que incluyen la definición de los niveles con sus respectivos espacios, así como la definición de los materiales que posteriormente se utilizará para la cuantificación. Los datos del inmueble son:

- **Tipo**
- **Edad**
- **Estado del inmueble**
- **Estado de ocupación**
- **Plantas tipo**

  Define si el proyecto incluirá plantas tipo o no
- **Número de niveles**

  Define el número total de niveles, en caso de existir plantas tipo se consideran los totales de cada tipo
- **Niveles**

Define los niveles y los espacios de cada nivel

- **Materiales**

  Define los materiales por defecto que se utilizarán en todo el levantamiento. Se agrupan en 5 categorías:

  - **Muros exteriores**
  - **Muros interiores**
  - **Pisos**
  - **Azotea**
  - **Plafones**

  Estas categorías a su vez contienen cada una:

  - **Base**
  - **Sub-base**
  - **Acabado**
  - **Color**
  - **Estado**

Al momento de editar los espacios de cada nivel se utiliza un catálogo de espacios que provienen de una plantilla por defecto, pero la cual se puede modificar para un proyecto en particular.

5.1.3. Datos del cliente

Los datos del cliente son los siguientes:

- **Nombre del cliente**
- **Necesidad de factura**
- **Nombre o Razón social**
- **RFC**
- **Dirección de facturación**
- **Imágenes de RFC**

```
<Project Id="2015-001" Name="buu" AltaProyecto="635784316494533072" UnitType="m"
    GridSize="100" GridVisibility="true" UnitSize="100" ScaleSize="2"
    calle="" entrecalle1="" entrecalle2="" colonia="" pais="" numExt=""
    numInt="" cp="" otros="" delegacion="" estado="" ladaInt="" ladaNac=""
    telefono="" fax="" extensionT="" extensionF="" latitud="19° 21' 59&quot;"
    longitud="-99° 10' 22&quot;" mapZoom="15" propertyAge="">
        <Niveles Numero="2">
        ...
        </Niveles>
        <Image Clave="2015-001-Imagen del proyecto">C:\...\2015-001-Imagen del proyecto.jpeg</Image>
        <PushPin Latitude="19.366493154150191" Longitude="-99.1729441374597" />
</Project>
```

Ilustración 5.1. Nodo XML de proyectos

5.1.4. Estructura de niveles

Esta estructura corresponde a un piso del levantamiento, y contiene todos los espacios que se encuentren en él. La estructura permite el manejo general de las alturas, por lo cual se tendrán que definir ciertas características para que se pueda tener un nivel válido.

El nivel contiene la siguiente información:

- **Clave**
  Es un identificador único asociado al nivel actual.
- **Nombre**
  Es el nombre con el cual se mostrará el nivel actual
- **Tipo**
  Define el tipo de nivel que se requiere
- **Espacios totales**
  La lista de espacios definidos dentro de este nivel, los cuales tendrán un nombre único por nivel
- **Espacios especificados**
  Define el número de espacios que han sido asignados lo cual permite que sean editables
- **Nivel inferior**
  Es un apuntador al nivel que se encuentra debajo del nivel actual
- **Nivel superior**
  Es un apuntador al nivel que se encuentra arriba del nivel actual
- **Entrepiso**
  Este dato se refiere a la altura que hay entre el piso del nivel actual y la losa del nivel superior (VER ILUSTRACIÓN 5.2)
- **NPT (Nivel de piso terminado)**
  Se refiere a la altura en la que se encuentra el piso del nivel actual

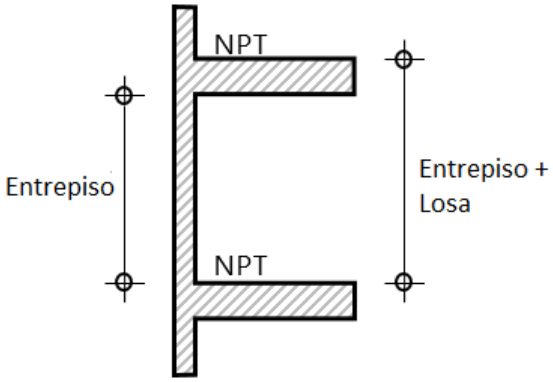


Ilustración 5.2. Esquema de nivel (vista frontal)

El tipo de nivel puede ser cualquiera de los siguientes:

- **Sótano**
- **Planta baja**
- **Nivel**
- **Mezzanine**
- **Pent-house**
- **Azotea**
- **Planta tipo**
- **Sótano tipo**

Estos tipos se separan en dos categorías:

- **Niveles normales**
- **Niveles tipo**

Los niveles tipo se usarán en construcciones donde varios niveles posean la misma estructura y por lo cual sería innecesario definir cada nivel por separado. Debido a esto se le asignará algún nivel tipo, ya sea planta tipo o sótano tipo, y permitirá definir el número de niveles que compartirán las mismas características.

Los niveles normales tendrán los datos descritos previamente, mientras que los niveles tipo tendrán como dato adicional el número de niveles (número de niveles que habrá con estas características).

Antes de poder definir los espacios que un nivel contendrá, se validará que la información introducida sea correcta, ya que de lo contrario podría generar errores en el proceso final de la aplicación. Se tienen que cumplir las siguientes reglas:

- El entrepiso, número de niveles y número de espacios tiene que ser un valor positivo
- La diferencia entre el NPT del nivel actual y el NPT del nivel superior debe ser mayor al valor del entrepiso actual (VER ECUACIÓN 5.1)

$$|\text{NPT}_{\text{nivel actual}} - \text{NPT}_{\text{nivel superior}}| > \text{Entrepiso}_{\text{nivel superior}} \qquad (5.1)$$

Si se cumplen estas dos reglas la aplicación acomodará los niveles por alturas y permitirá la asignación de espacios.

Los niveles se almacenan en la estructura mostrada en la ILUSTRACIÓN 5.3.

```
<Niveles Numero="8">
    <Nivel Id="1" Name="Planta baja" Type="Planta baja" Clave="PBA" NPT="0.000"
        Entrepiso="2.300" Espacios="2" LvlDown="" LvlUp="N01" Especificados="2">
        ...
    </Nivel>
    <Nivel Id="2" Name="Nivel 1" Type="Nivel" Clave="N01" NPT="2.500"
        Entrepiso="2.300" Espacios="2" LvlDown="PBA" LvlUp="" Especificados="2">
        ...
    </Nivel>
    ...
</Niveles>
```

Ilustración 5.3. Nodo de XML de niveles

5.1.5. Estructura de espacios

Esta estructura corresponde a un cuarto del levantamiento y su principal contenido es el trazo de la geometría correspondiente. Gráficamente un espacio será una polilínea, por lo cual, esta estructura contendrá todos los segmentos y etiquetas.

El espacio contiene la siguiente información:

- **Nombre**

  Es el identificador del espacio y será único para cada nivel, por lo tanto, puede que haya dos o más espacios con el mismo nombre, siempre y cuando pertenezcan a diferentes niveles.

- **Geometría de trazo**

  Esta geometría es la que se crea usando el trazo del usuario y se utiliza hasta terminar con el análisis geométrico, por lo tanto, guarda los segmentos obtenidos después del dimensionamiento, las diagonales que se crearon y todos los triángulos utilizados. Además, contiene el área calculada hasta este proceso, así como la posición de la etiqueta de área en caso de que se haya movido de su posición original (VER ILUSTRACIÓN 5.4).

```
<Geometry_Trazo Area="12" TagInsPt="610.430583863935,196.722319209376">
  <Triangles>
    <Triangle Area="6" PTA="410.430583863934,346.722319209376" PTB="410.430583863934,46.7223192093759"
    PTC="810.430583863934,46.7223192093759" />
    <Triangle Area="6" PTA="410.430583863934,346.722319209376" PTB="810.430583863934,46.7223192093759"
    PTC="810.430583863934,346.722319209376" />
  </Triangles>
  <Diagonals>
    <Diagonal Name="D-1" Longitud="5" TagInsPt="398.316497802734,195.024002075195@323.130096435547"
    StartPoint="410.430583863934,346.722319209376" EndPoint="810.430583863934,46.7223192093759" Edited=
    "2" />
  </Diagonals>
  <Segments/>
    <Segment Name="S-1" Longitud="3" TagInsPt="148.716491699219,295.023986816406@-90" StartPoint=
    "410.430583863934,346.722319209376" EndPoint="410.430583863934,46.7223192093759" Edited="2" />
    ....
  </Segments>
</Geometry_Trazo>
```

Ilustración 5.4. Nodo XML de geometría de trazo

- **Geometría de acabado**

  Esta geometría se genera al término del proceso de dimensionamiento y será la que se exporte al final de la aplicación. En un principio solo contendrá una copia de los segmentos que se encuentran en la geometría de trazo, pero al utilizar las diferentes herramientas de la aplicación permitirá almacenar segmentos de arco y filetes. Además, tendrá el área que se calcule después de modificar los muros, con arcos o filetes (VER ILUSTRACIÓN 5.5).

```
<Geometry_Acabado Area="0" TagInsPt="603.403015241438,206.879752961436">
  <Walls>
    <LineWall Name="M-1" Longitud="3" TagInsPt="410.430572509766,196.722320556641@-90"
    StartPoint="410.430583863934,346.722319209376" EndPoint=
    "410.430583863934,46.7223192093759" Edited="0">
      <Vano Name="P-2" Type="Doors" />
    </LineWall>
      ...
  </Walls>
</Geometry_Acabado>
```

Ilustración 5.5. Nodo XML de geometría de acabado

- **Geometría de alzado**

  Esta geometría contiene una geometría independiente por cada muro y se crea en el momento en que se define el tipo de muro que corresponde a cada segmento, además en esta se incluyen vanos, puertas y ventanas, las cuales se insertarán una por una y se reflejarán en la geometría de acabado (VER ILUSTRACIÓN 5.6).

```
<Geometry_Alzado>
    <WallViews>
        <WallView Name="M-1" Type="0" Area="6.9" Flip="1" WallType="Carga">
            <Moldura Type="0" Altura="0.1" Espesor="0.025" Apiso="0.9" Enabled="false" />
            <Moldura Type="1" Altura="0.1" Espesor="0.025" Apiso="0.9" Enabled="false" />
            <Moldura Type="2" Altura="0.1" Espesor="0.025" Apiso="0.9" Enabled="false" />
            <Dimensions>
              <Dimension Name="DimWidth" Value="3" />
              <Dimension Name="DimHeight" Value="2.3" />
            </Dimensions>
            <Notes />
            <Materiales/>
        </WallView>
        ...
    </WallViews>
    <Doors>
        <Door Name="P-1" Type="1" Area="1.89" VanoType="Puerta" Abatimiento="Interior" Apertura="Izquierdo"
         AcabadoType="0" OwnerWall="M-2" Flip="1" AdjacentRoom="" RelatedVano="">
            <Dimensions/>
            <Materiales>
              <Material Clave="Madera tambor" Concept="" Value="1.8900000000000001" UnitType="m2" Type="6"
              />
              <Material Clave="Estándar" Concept="" Value="1.8900000000000001" UnitType="m2" Type="5" />
            </Materiales>
    </Door>
        ...
    </Doors>
    <Windows />
</Geometry Alzado>
```

Ilustración 5.6. Nodo de XML de geometría de alzado

- **Pisos y plafones**

  Esta estructura define los materiales y el área calculada para los pisos y los plafones.

  La variable *Type* nos distingue entre piso y plafón, siendo “1” piso y “0” plafón

  El desnivel es la diferencia entre el nivel de piso terminado, ya sea un incremento o decremento en el piso del nivel (VER ILUSTRACIÓN 5.7).

```
<Surface Type="1" Area="12.162" Desnivel="0">
  <Materiales/>
</Surface>
```

Ilustración 5.7. Nodo de XML de pisos y plafones

- **Visibilidad**

  Almacena las opciones de visualización para el módulo de levantamientos (VER ILUSTRACIÓN 5.8).

```
<Cuartos >
    <Cuarto Name="Vestíbulo de acceso" Handle="12" Abs_Handle="13" Norte="0">
        <Visibilidad ViewMode="1">
            ...
        </Visibilidad>
        <Geometria_Trazo />
        <Geometria_Acabado />
        <Geometria_Alzado />
    </Cuarto>
    <Cuarto Name="Comedor" Handle="12" Abs_Handle="13" Norte="0">
        ...
    </Cuarto>
    ...
</Cuartos>
```

Ilustración 5.8. Nodo XML de espacios

5.2. Visibilidad del modelo

Esta estructura almacena la geometría que se muestra actualmente, así como las capas que cada geometría tiene visibles. Cada geometría tendrá opciones de visibilidad que son independientes de la otra (VER ILUSTRACIÓN 5.9).

Las capas que se podrán activar o desactivar son las siguientes:

- **Rejilla**

  Contiene la retícula que se utiliza como guía
- **Textos**

  Contiene todas las etiquetas asociadas a los segmentos de la geometría actual
- **Diagonales**

  Contiene todas las diagonales que se crearon. Se oculta por default al terminar el dimensionamiento
- **Trazo**

  Contiene los segmentos de la geometría de trazo. En caso de encontrarse sobre la geometría de trazo aplicara a todos los segmentos. Si se encuentra sobre la geometría de acabado dependerá de si se creó algún arco o filete ya que de lo contrario no se verá ningún cambio
- **Acabado**

  Contiene los segmentos de la geometría de acabado
- **Textos de vano**

  Contiene los textos de vanos, puertas y ventanas
- **Cotas**

  Las dimensiones de vano

- **Puertas**
  Contiene todas las puertas que se encuentren definidas, así como sus etiquetas
- **Ventanas**
  Contiene todas las ventanas que se encuentren definidas, así como sus etiquetas
- **Symbol**
  Contiene las cotas extras que el usuario necesite generar

```
<Visibilidad ViewMode="0">
  <Trazo Trazo="true" Grid="true" Area="false" Text="true" Diagonal="true" />
  <Acabado Trazo="false" Grid="true" Area="true" Text="true" Diagonal="false" Acabado=
  "false" Door="true" Window="true" Symbol="true" VanoTag="true" VanoDim="true" />
</Visibilidad>
```

Ilustración 5.9. Nodo XML de visibilidad

5.3. Unidades

Para el manejo adecuado de la información almacenada dentro de la aplicación se manejan diferentes tipos de unidades, esto con el objetivo de modular los procesos que se realizan y permitir que sean independientes de otros.

Se manejan tres tipos de unidades:

- Unidades internas
- Unidades gráficas
- Unidades de visualización

5.3.1. Unidades internas

Estas unidades son transparentes al usuario y cada una de estas equivaldrá a un metro. El objetivo de estas unidades es el de realizar todos los cálculos durante el análisis geométrico. Esto permite que sin importar el acercamiento o el formato que el usuario quiera visualizar, los cálculos se realicen de forma homogénea.

5.3.2. Unidades gráficas

Estas unidades también serán transparentes al usuario y serán manejadas en *pixeles*. Estas unidades se verán afectadas por el acercamiento y son las encargadas de controlar la separación de las guías de la retícula de dibujo. Al mismo tiempo son las que se reflejan en el dibujo ya que están serán aplicadas al *Path*, el cual es el encargado de dibujar lo que se muestra en pantalla.

5.3.3. Unidades de visualización

Estas unidades son elegidas por el usuario y serán las que se mostrarán en las etiquetas, así como las que el usuario introducirá para dimensionar. Debido al amplio rango de levantamientos que pueden existir, la aplicación permite el manejo de diferentes unidades de visualización. Esta funcionalidad permitirá una mejor sincronización con diferentes distanciómetros, ya que algunos de estos dispositivos envían las mediciones en diferentes

formatos. Al mismo tiempo se podrán cambiar las unidades usadas en cualquier momento durante la realización de un levantamiento, llevando a cabo la conversión correspondiente de forma automática.

Se manejarán las siguientes unidades:

- **Sistema Métrico**
    - Metros
    - Centímetros
    - Milímetros
- **Sistema Imperial**
    - Pies
    - Pulgadas

6. Periféricos y sensores

Este capítulo describe el uso e implementación de los periféricos y sensores que utilizamos para el funcionamiento de la aplicación.

Los dispositivos usados son:

- Distanciómetro láser
- GPS
- Brújula
- Cámara

6.1. Distanciómetro láser

Elegimos un distanciómetro láser cuyo modelo es Leica DISTO D510 (**VER ILUSTRACIÓN 6.1**) por que cumple con la norma ISO 16331-1 (**APÉNDICE H1**). La norma garantiza que las mediciones sean precisas, el alcance es suficiente para los espacios en los que se realizarán los levantamientos.

Este dispositivo cuenta con las siguientes características:

- ±1mm
- 200 m de alcance máximo
- *Bluetooth®* Smart 4.0
- Sensor de inclinación de 360º
- Protección *IP65* (**APÉNDICE H2**) contra salpicaduras de agua y estanco al polvo
- Cámara para captura de punto de mira en pantalla
- Compatibilidad con Sistema Imperial y Sistema Métrico
- Se necesitó usar el programa de lecturas de Leica

Ilustración 6.1. Distanciometro Leica DISTO D510

6.1.1. Análisis de unidades

El distanciómetro tiene distintos formatos para enviar las unidades a la computadora o tableta. Para enviar la información se utilizó un programa de Leica llamado *DISTO transfer* para PC.

Para el caso de las unidades métricas la transferencia fue directa ya que todos los valores eran numéricos y sin unidades.

Para el caso de las unidades imperiales, se crearon expresiones regulares. Entre los casos que tuvimos que analizar fueron:

- **Pies con treintavos de pulgadas**

  Los pies tenían un valor entero seguido de un punto el cual definía las pulgadas y un espacio definía la fracción.

  19.02 19/32

  19.11 15/16

  15.09 5/32

- **Pies con pulgadas enteras**

  Los pies tienen un valor entero seguido de una tabulación para separar las pulgadas como se muestra en los siguientes ejemplos:

  16 04

  2 05

  12 02

6.2. GPS

La aplicación se basa en la técnica de geolocalización por *IP* y en nuestro caso utilizamos el proveedor de localización de *Windows®*. Primero se encuentra la dirección *IP* de la antena del proveedor. Con esta se realizan *pings* a los conmutadores con ubicación geográfica conocida y que estén conectados al proveedor. Dependiendo de la latencia de los *pings* se realiza un proceso de triangulación que ubica el dispositivo con un error de 100 m.

En caso de que el dispositivo cuente con GPS, *Windows®* usará la lectura del dispositivo y el error será mucho menor en el orden de 5 m. La ubicación se genera con coordenadas geográficas.

Uno de los requerimientos de aplicación es ubicar la dirección del levantamiento usando el GPS para obtener un croquis del lugar.

Aprovechando que se utilizan tecnologías de *Microsoft®* optamos por *Bing Maps*, ya que esta *API* tiene una mayor compatibilidad con el sistema y no existía un *API* oficial de otros servicios de *geolocalización*.

6.2.1. Bing API [5]

Bing Maps es una plataforma de mapas de *Microsoft®* que ofrece una colección de servicios web los cuales pueden ser manejados desde el *API* en *.NET®*.

Para poder acceder al *API* es necesario registrarse y generar una llave la cual brinda permisos a la aplicación para realizar consultas.

Existen dos modalidades de llave:

Llave básica

- Acceso desde sitios web
- Acceso desde una aplicación de *Windows®*
- Acceso desde un dispositivo móvil
- Para fines educativos y sin lucro

- Para aplicaciones web y móviles que no sean de *Windows®* 125,000 consultas acumulativas en un periodo de 12 meses
- Para aplicaciones de *Windows®* 50,000 transacciones diarias
- Aplicaciones privadas de *Windows®*
- Desarrollo y pruebas

Llave empresarial

Ofrece lo mismo de la llave básica, pero sin restricción de consultas y permite comercializar la aplicación.

Para usar *Bing Maps* dentro de la aplicación, la *API* incluye el control del mapa, el cual permite la visualización de un mapa e incluye los controles básicos para la manipulación del mismo. Este control también permite configurar el tipo de mapa que se quiere mostrar, ya sea un mapa de carreteras o incluso un mapa con vista satelital. Para desplazarse dentro del mapa se puede llegar a la ubicación manualmente, o si se prefiere centrar el mapa en las coordenadas en las que se encuentra el dispositivo.

Para seleccionar la ubicación en el croquis el usuario da un *Tap* en la pantalla del mapa y se inserta un marcador. El control que despliega el mapa genera la información necesaria para generar una *URL* que se usa en el *web service*, que genera una imagen con los tamaños necesarios.

6.2.2. Formato de la URL [6] La URL está formada como sigue:

http://dev.virtualearth.net/REST/v1/Imagery/Map/imagerySet/centerPoint/zoomLevel?mapSize=mapSize&pushpin=pushpin&key=BingMapsKey

De la URL se definen los siguientes parámetros:

Parámetro	Descripción	Ejemplo
centerPoint	Un punto global definido con longitud y latitud	centerPoint=47.610,-122.107
imagerySet	El tipo de mapa, aéreo o de camino.	Aerial, Road
zoomLevel	El nivel de acercamiento requerido para desplegar la imagen.	Un valor entero que va de 0 a 21
pushpin	Un marcador de posición, se especifica la posición del marcador y el tipo de marcador a usar.	pushpin=47.610,-122.107;5; P10
mapSize	El ancho y alto de la imagen en pixeles	mapSize=100,600
key	La llave de la aplicación que permite acceder al servicio de Bing Maps	?key=AvQBwGL1u…IyHoa6S67I2f7

Tabla 6.1 Parámetros de la URL

El resultado de la consulta es visible en la ilustración 6.2

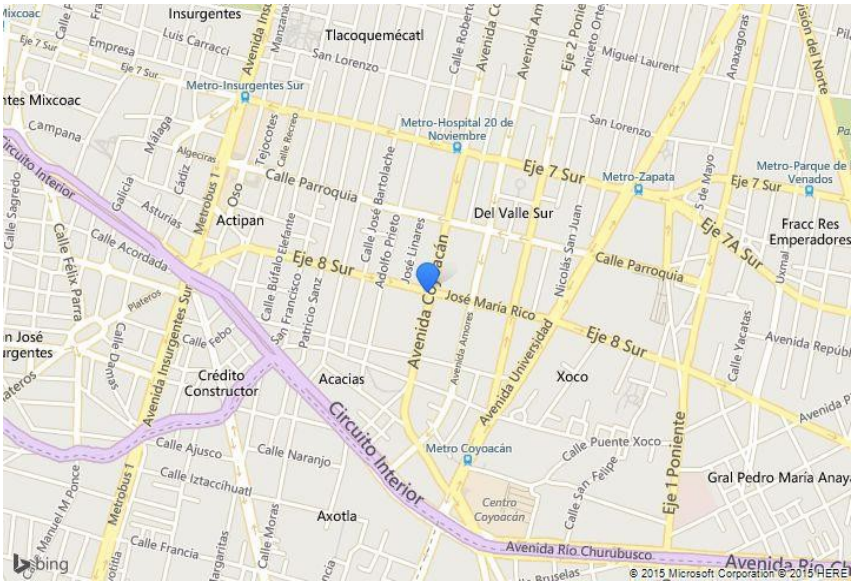


Ilustración 6.2. Vista de consulta de ubicación usando Bing Maps

Para utilizar esta característica se habilitó el permiso de Localización.

Para acceder a la ubicación se realizó el siguiente procedimiento.

Con ayuda de la clase *Geolocator* accedemos al dispositivo de geolocalización que implementa el API de *Windows® RT*. Se definió el rango de exactitud alto para obtener mejores resultados.

```
Windows.Devices.Geolocation Geolocator gps = new Geolocator();
gps.DesiredAccuracy = Windows.Devices.Geolocation PositionAccuracy.High;
Windows.Devices.Geolocation Geoposition pos = await gps.GetGeopositionAsync();
```

De la clase *Geoposition* podemos encontrar la latitud y longitud y crear una vista que se desplegará en el mapa de *Bings Maps*.

```
Bing.Maps.Location loc = Location(Latitude, Longitude);
Bing.Maps.Map.SetView(loc, MapZoom);
```

6.3. Brújula

En los levantamientos se requiere obtener la ubicación del norte por lo cual implementamos una brújula. Buscando una solución para la implementación encontramos que el API incluía acceso a un objeto de tipo brújula (**APÉNDICE H3**).

Solo fue necesario programar una clase que controlara la lectura de la brújula.

La clase inicializa administra los tiempos en el que la aplicación obtiene una lectura de la brújula la cual entra a un proceso de dibujo de la ubicación del Norte.

Para obtener la lectura de la brújula se utiliza el siguiente método:

```
Windows.Devices.Sensors.Compass.GetCurrentReading().HeadingMagneticNorth;
```

El valor del ángulo es en grados. Como la pantalla dibuja con el eje de la Y invertido, aplicamos una transformación geométrica con el ángulo leído invirtiendo el eje de las Y.

Utilizamos una imagen y un texto para mostrar la orientación (**VER ILUSTRACIÓN 6.3**).

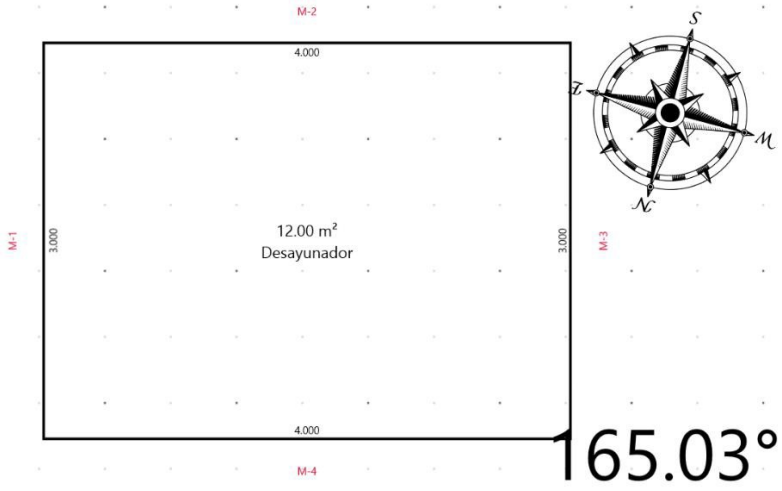


Ilustración 6.3. La brújula en la vista de espacios

6.4. Cámara

Como requerimiento adicional en la documentación de la aplicación se requería exponer el estado de los muros, la vista del lugar o cualquier detalle sobre algún elemento del proyecto.

La forma para documentar estos elementos es con imágenes, ya fueran desde un archivo o por medio de una fotografía. Es por esto que era indispensable que la tableta tuviera una cámara que ayudará a complementar la creación de reportes, haciendo que estos puedan incluir información que de otra manera seria muy difícil tener.

Con el *API* de *Windows* la forma de captura fue transparente. Solo fue necesario definir un permiso para que la aplicación accediera a la cámara.

Los permisos se definen como *Capabilities* en la aplicación y nosotros habilitamos los siguientes:

- **Webcam Capabilities**
  Permite usar la cámara del dispositivo
- **Pictures Library**
  Permite escribir y leer información de la carpeta de imágenes de Windows

Una vez definidos los permisos, se accede al dialogo de cámara de Windows con la siguiente clase:

```
Windows.Media.Capture.CameraCaptureUI
```

Se pueden definir los parámetros de la imagen, tales como el formato de archivo y la capacidad del usuario para recortar o editar la imagen tomada.

La foto se toma llamando al método:

```
CameraCaptureUI.CaptureFileAsync(CameraCaptureUIMode.Photo);
```

7. Cuantificación

El proceso de cuantificación se encarga de cuantificar los materiales asignados por el usuario usando las superficies generadas por el dimensionamiento del levantamiento.

La cuantificación es el proceso final de la aplicación, donde la salida es presentada en un reporte.

7.1. Asignación de materiales

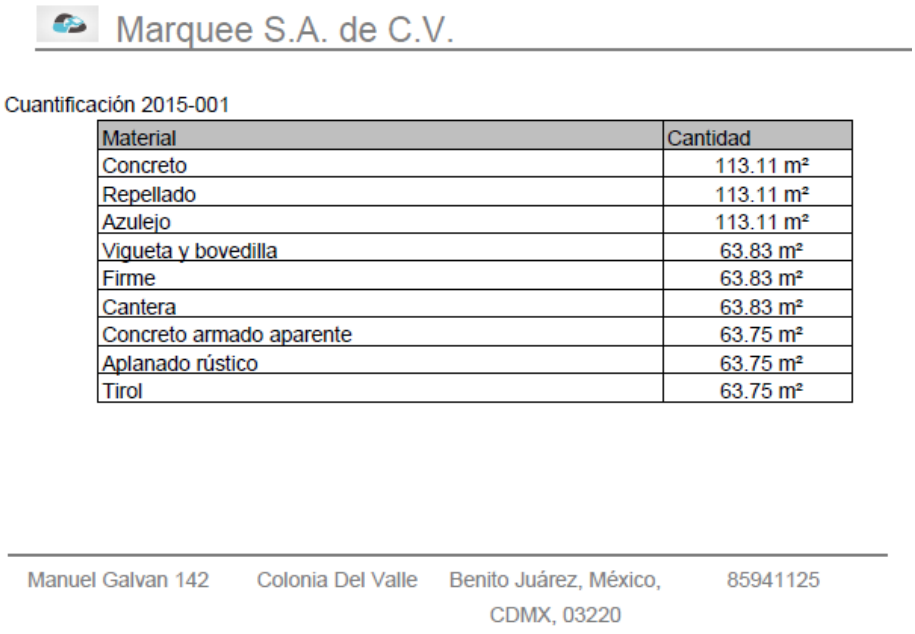
En esta sección el usuario se encarga de asignar materiales específicos a los muros, pisos y plafones del proyecto. Inicialmente todos los elementos tienen aplicados un material.

Actualmente la aplicación realiza cuantificaciones de área. Los muros la definen en la sección de alzados y los pisos o plafones en la sección de cambio de *NPT*.

Al realizar una cuantificación se puede especificar si se quiere el total del proyecto, el total de un nivel, o un cuarto en específico.

Como lo que importa es el total de materiales, se suman las áreas de los materiales asignados agrupándolos por nombres. En cualquiera de los tres casos se llega a un resumen del cálculo de los materiales.

El reporte que genera la aplicación es como se muestra en la ilustración **7.1**.



Marquee S.A. de C.V.

Cuantificación 2015-001

Material	Cantidad
Concreto	113.11 m²
Repellado	113.11 m²
Azulejo	113.11 m²
Vigueta y bovedilla	63.83 m²
Firme	63.83 m²
Cantera	63.83 m²
Concreto armado aparente	63.75 m²
Aplanado rústico	63.75 m²
Tirol	63.75 m²

Manuel Galvan 142 Colonia Del Valle Benito Juárez, México, CDMX, 03220 85941125

Ilustración 7.1. Ejemplo de reporte de cuantificación

La edición de materiales se revisará en el siguiente módulo de Interfaz gráfica.

8. Interfaz gráfica [2]

La interfaz gráfica cumple un punto clave para nuestra aplicación donde buscamos que el diseño sea limpio y sencillo para que pueda ser operado de manera táctil. Las aplicaciones de *Windows® Store* o *Universal App* cumplen con este propósito proporcionando una metodología y reglas que deben seguir las aplicaciones.

El diseño es llamado metro, el cual se basa en la simplicidad. Esta se logra transformando los textos de botones a un icono plano y removiendo degradados y bordes en los controles. La idea surge de los señalamientos basados en el lenguaje gráfico.

La aplicación entra en la categoría de *Windows® Store Apps* (**APÉNDICE S4**), la cual es el antecesor a *Universal Application de Windows®*.

La aplicación corre en un entorno distinto de *Windows*, llamado *Windows® Runtime o WinRT* el cual define un nuevo *Framework* basado en *.NET®* que tiene como objetivo el sistema operativo de *Windows® 8* y la siguiente versión de *Windows® 8.1*. Esta última incluye algunas mejoras y es donde se desarrolló la aplicación.

Una aplicación de *WinRT®* puede correr tanto en una arquitectura *ARM*, *x64* y *x86* sin cambiar el código y el desarrollo de su interfaz gráfica se basa en el lenguaje descriptivo *XAML* (**APÉNDICE A1**).

8.1. Tablero de proyectos

La ventana inicial (**VER ILUSTRACIÓN 8.1**) es el tablero de proyectos que muestra un resumen de aquellos que se encuentran activos de la aplicación.

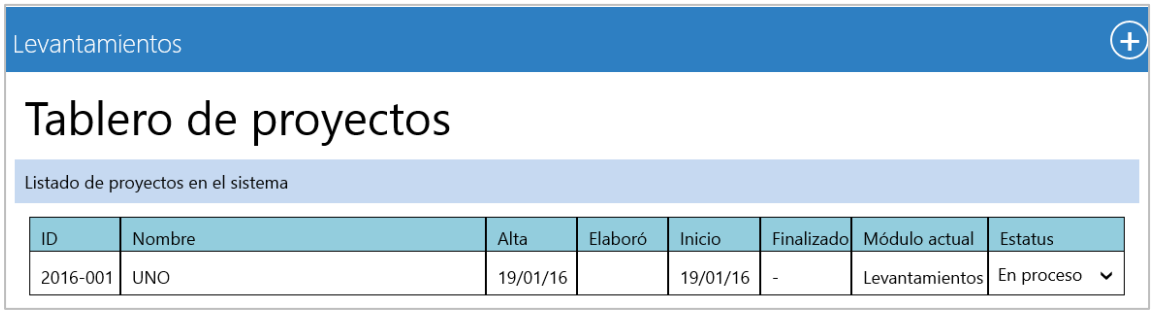


Ilustración 8.1. Ventana de tablero de proyectos

Cada sección tiene una barra superior (**VER ILUSTRACIÓN 8.2**) que permite navegar entre los distintos módulos.

Ilustración 8.2. Navegador de módulos de la aplicación

8.2. Proyectos

Para comenzar a usar la aplicación lo primero que se debe hacer es crear un proyecto. Para esto se definió la ventana de alta de proyecto (**VER ILUSTRACIÓN 8.3**) que solicita al usuario el nombre del proyecto.

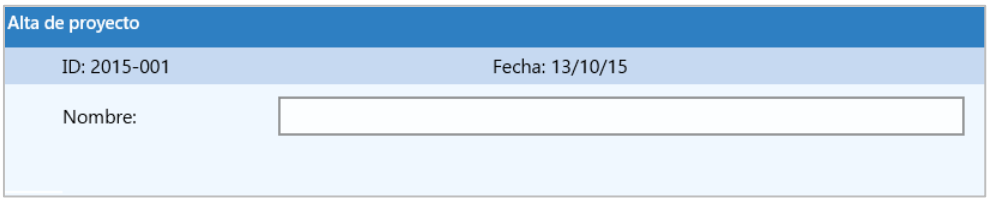


Ilustración 8.3. Diálogo para crear nuevo proyecto

Una vez ingresado el nombre, se carga por default la pestaña de datos generales del proyecto. El área de proyectos (**VER ILUSTRACIÓN 8.4**) está conformada por una barra de aplicación, un menú de selección de módulos y el área de trabajo.

Ilustración 8.4. Distribución del área de proyectos

En la barra principal nos aparece el botón de opciones que nos permite desplegar el menú de proyectos. Del lado derecho tenemos la lista de proyectos (**VER ILUSTRACIÓN 8.5**) y del lado izquierdo los módulos.

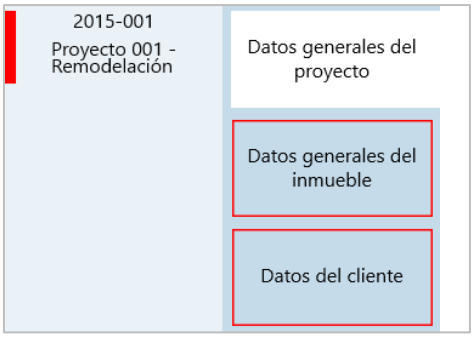


Ilustración 8.5. Lista de proyectos

Para agregar un nuevo proyecto solo es necesario dar clic en el botón superior derecho de la barra de aplicación.

8.2.1. Datos generales del proyecto

En esta sección (**VER ILUSTRACIÓN 8.6**) se tiene el formulario principal. El usuario proporciona la siguiente información:

- Tipo
- Nombre del cliente
- Imagen de la fachada
- Croquis
- Contacto del usuario

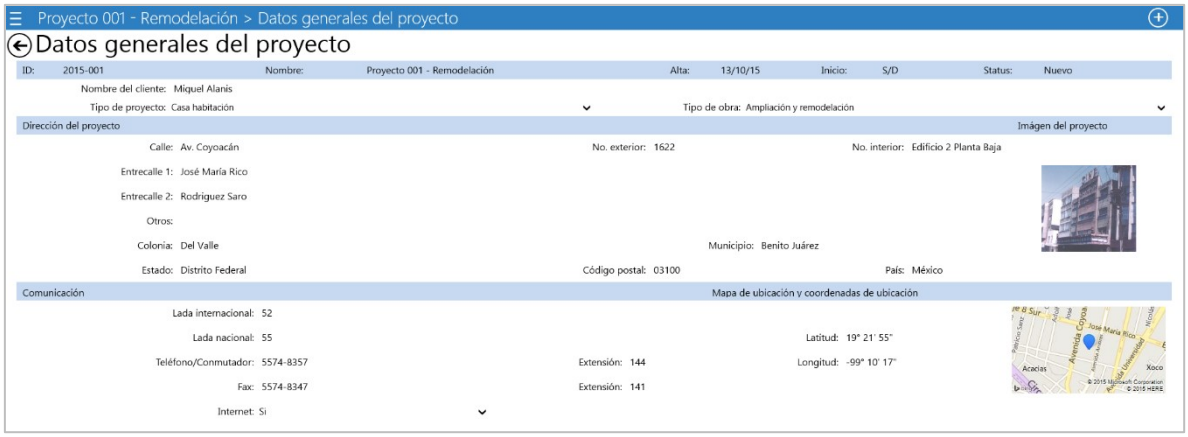


Ilustración 8.6. Ventana de datos generales del proyecto

Para obtener el croquis del levantamiento se creó una ventana (**VER ILUSTRACIÓN 8.7**) que implementa el control de *Bing Maps*. Este control permite insertar un marcador y crear una imagen del área seleccionada.

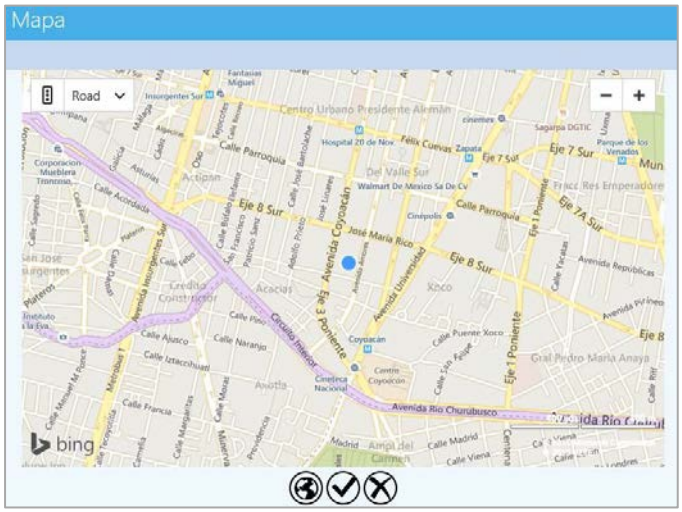


Ilustración 8.7. Selector de ubicación

La tarea principal de estos formularios consiste en registrar el estado del proyecto, la dirección del levantamiento, la información de contacto y la imagen del croquis del lugar. La imagen de la fachada se puede cargar desde un archivo o tomando una fotografía usando los botones de foto y archivo (**VER ILUSTRACIÓN 8.8**).

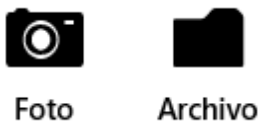


Ilustración 8.8. Botones para seleccionar la imagen del proyecto

8.2.2. Datos generales del inmueble

En esta sección (**VER ILUSTRACIÓN 8.9**) se define la estructura del levantamiento, se especifican las características del inmueble, el número de niveles y la selección de materiales.

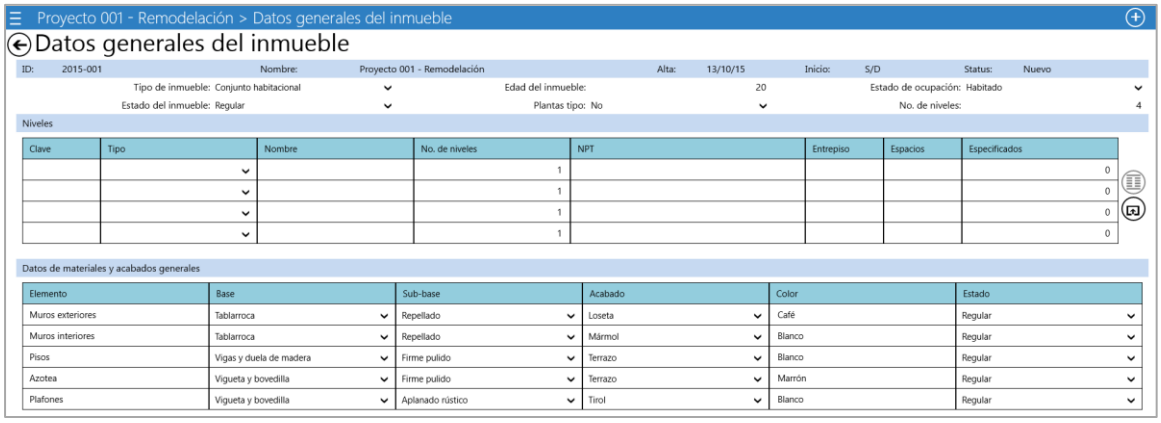


Ilustración 8.9. Ventana de datos generales del inmueble

La tabla de niveles se genera de forma automática al cambiar el número de estos. Es necesario especificar el tipo de nivel, la altura del piso terminado, la altura entre pisos y el número de espacios.

Una vez definida la información podemos abrir el selector de espacios (**VER ILUSTRACIÓN 8.10**). Para abrirlo solo es necesario dar clic en el botón .

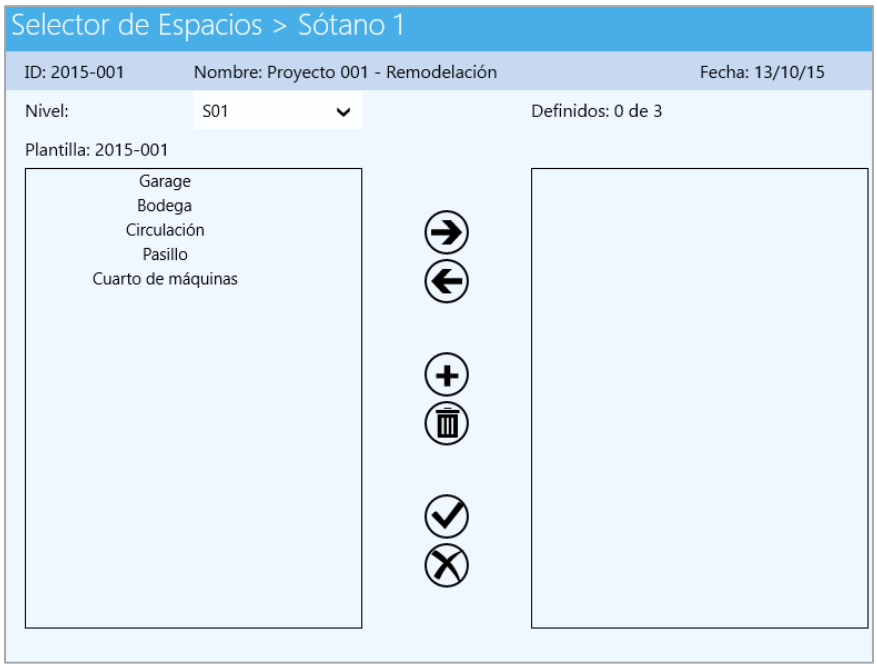


Ilustración 8.10. Editor de espacios de nivel

El selector de espacios cuenta del lado derecho con *una caja de selección* que permitir navegar entre los niveles definidos. Al seleccionar un nivel, se despliegan sus espacios disponibles.

Con los botones centrales podemos realizar las siguientes operaciones que se muestran en la tabla **8.1**:

	Botón	*Acción*
	Anexar un espacio	Anexa un espacio a un nivel. El número de espacios fue definido anteriormente.
	Remover un espacio	Remueve un espacio en el nivel seleccionado.
	Definir un nuevo tipo de espacio	En caso que un espacio no exista la aplicación permite definir un nuevo tipo de espacio.
	Elimina un espacio definido	Cuando la plantilla tiene demasiados espacios y es difícil encontrar alguno, se pueden eliminar sin dañar los niveles que tienen al espacio eliminado.
	Aceptar	Ejecuta los cambios recibidos
	Cancelar	Sale del editor sin realizar ningún cambio

Tabla 8.1 Botones del editor de espacios de nivel

Al definir un nuevo espacio aparece la siguiente ventana (**VER ILUSTRACIÓN 8.11**) la cual solicita su nombre. Los espacios de plantilla son a nivel de proyectos, por lo que no afectarán otros proyectos.

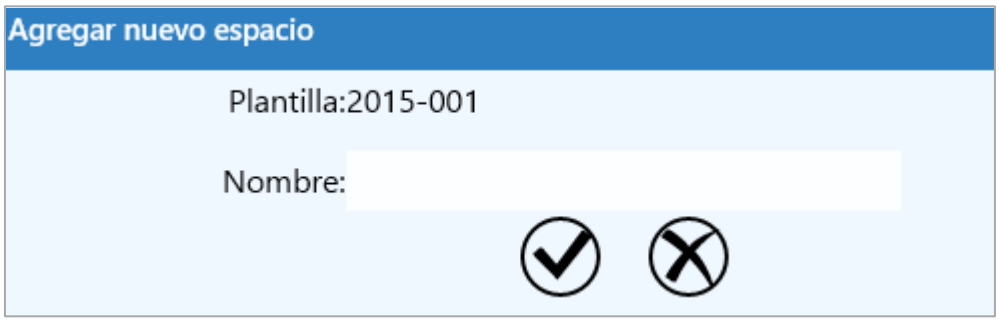


Ilustración 8.11. Nueva plantilla de proyecto

8.2.3. Datos del cliente

Esta sección (**VER ILUSTRACIÓN 8.12**) contiene la información de contacto y los datos que se requieren para facturarle.

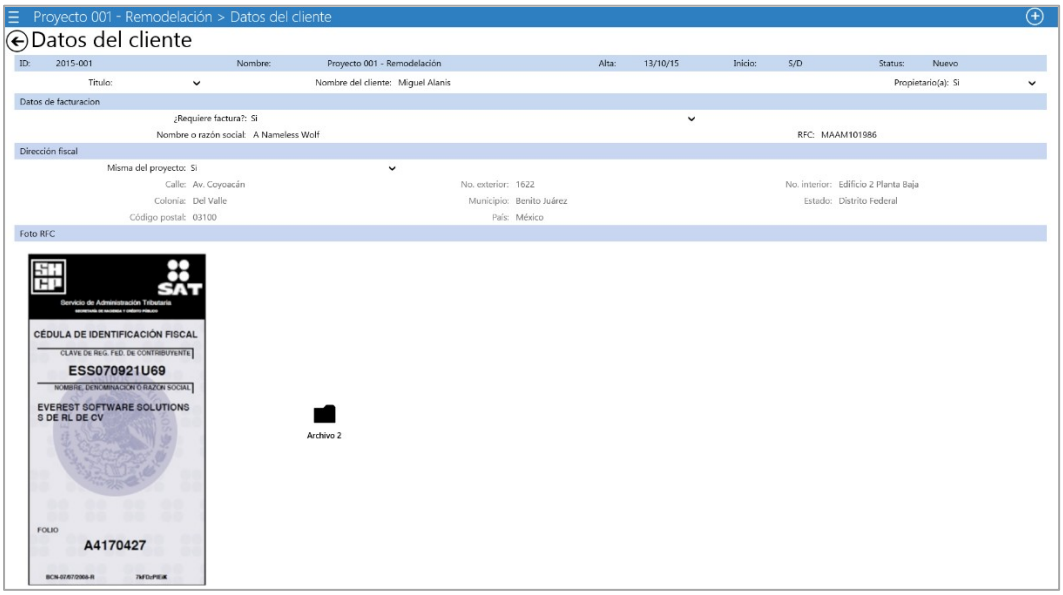


Ilustración 8.12. Ventana de datos del cliente

Se puede cargar la imagen del RFC dando clic en el botón de archivo .

8.2.4. Opciones de proyecto

Las opciones de proyecto se encuentran en la barra inferior (**VER ILUSTRACIÓN 8.13**) de la aplicación.

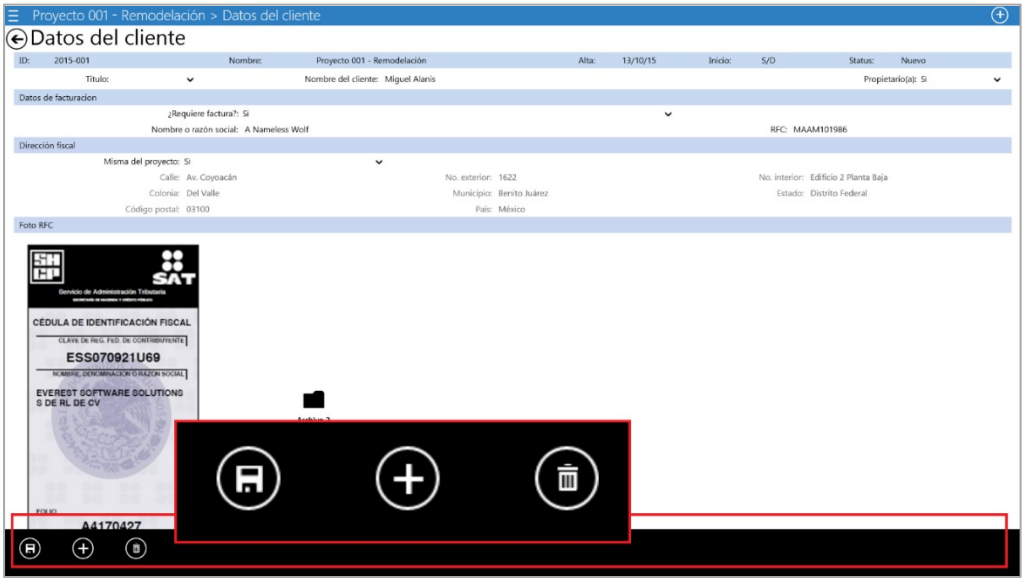


Ilustración 8.13. Barra de opciones de proyectos

Las operaciones para proyectos se describen en la tabla 8.2.

Botón	Acción
Exportar proyecto	Realiza la exportación del proyecto actual a un archivo de extensión “lpt”.
Importar proyecto	Realiza la importación de un proyecto desde un archivo “lpt”.
Borrar proyecto	Elimina el proyecto actual

Tabla 8.2 Acciones de proyectos

8.2.5. Colaboradores

Se creó una agenda (VER ILUSTRACIÓN 8.14) que permite crear contactos para cada tipo de proyecto.

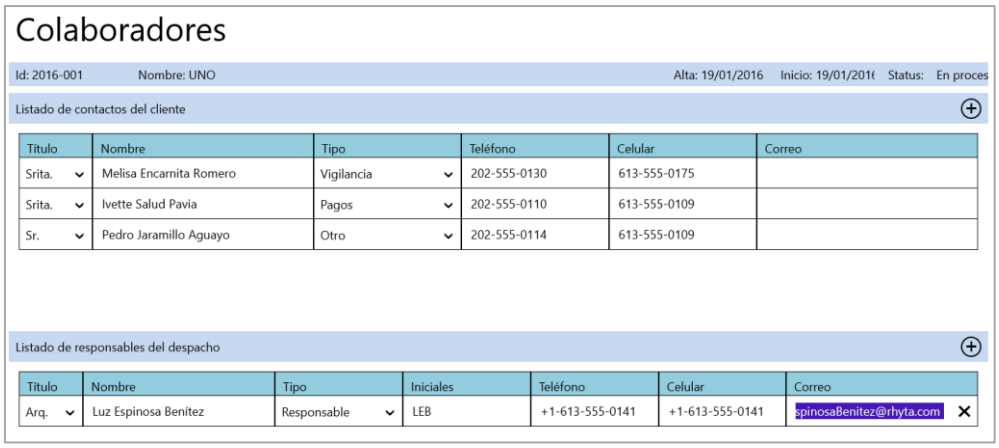


Ilustración 8.14. Agenda de colaboradores de proyectos

8.3. Levantamientos

En esta área (**VER ILUSTRACIÓN 8.15**) es donde se lleva acabo todo el proceso gráfico de la aplicación. La distribución del área de trabajo es la siguiente:

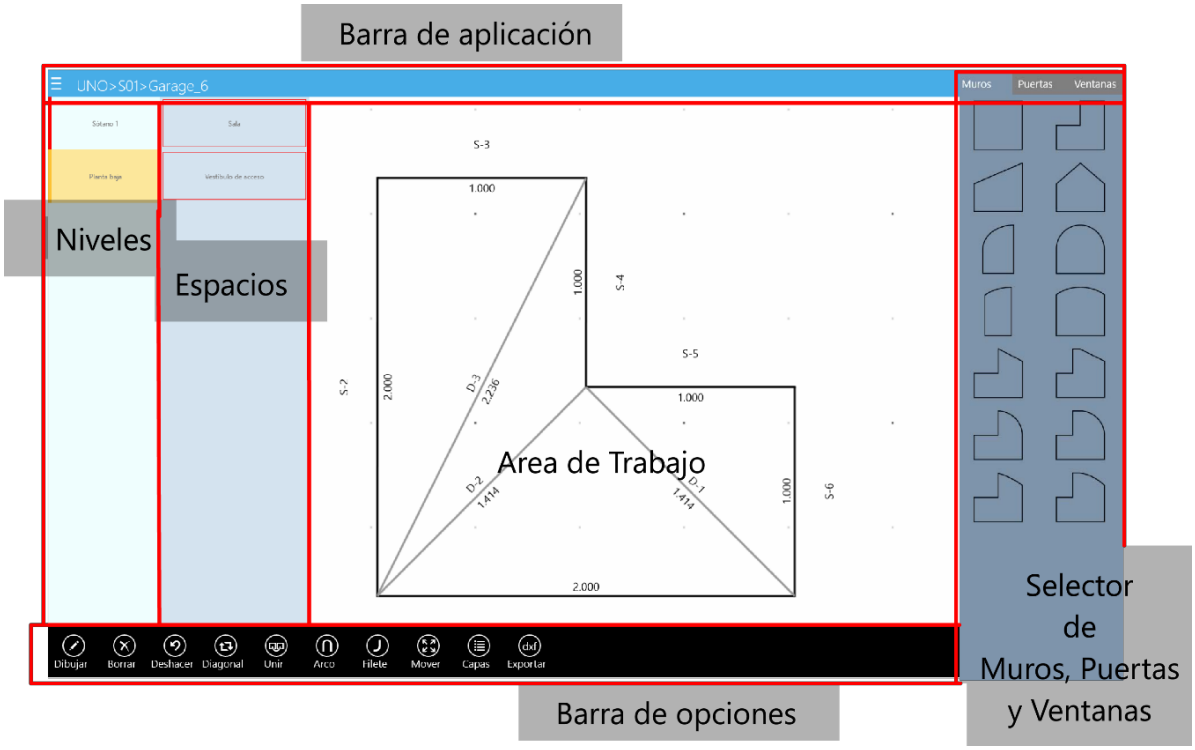


Ilustración 8.15. Distribución del área de levantamientos

En la barra superior de la aplicación tenemos un botón que despliega la lista de niveles y espacios disponibles. Para comenzar a trabajar en el área de levantamientos es necesario seleccionar un nivel y un espacio de sus respectivas colecciones. En la barra superior (**VER ILUSTRACIÓN 8.16)** aparecerá como título la jerarquía del espacio seleccionado.

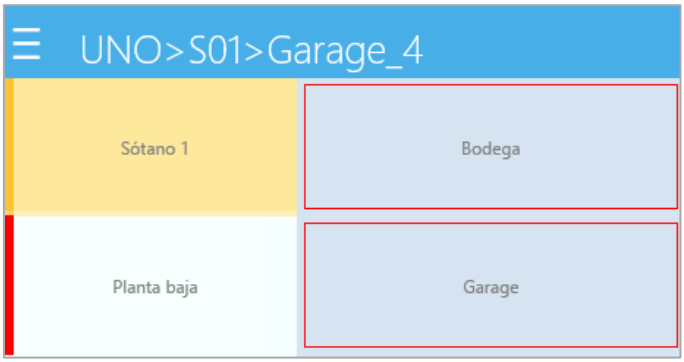


Ilustración 8.16. Lista de niveles y espacios

Al seleccionar un espacio, la aplicación abre una ventana blanca y permite al usuario cambiar el acercamiento y comenzar a dibujar.

En el área de trabajo se tiene una rejilla con dos tipos de puntos (VER ILUSTRACIÓN 8.17). Los puntos con una tonalidad gris fuerte se encuentran separados a una distancia de 1 [u] definida por el usuario y los puntos intermedios a ½ [u] definida por el usuario.

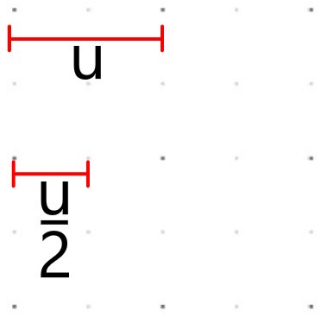


Ilustración 8.17. Espacios de rejilla en la aplicación

Las opciones de dibujo se definen la tabla 8.3:

Botón	Acción
	Permite el dibujado a mano alzada en el espacio seleccionado. Si ya existe un trazo se restringe el dibujado.
	Se elimina todo el contenido del espacio actual.
	Deshace la última acción.
	Habilita el intercambio de diagonales.
	Convierte un segmento seleccionado en un arco de media circunferencia.
	Crea un filete entre dos muros seleccionados.
	Permite el movimiento de etiquetas.
	Abre el visor de capas y permite cambiar la visibilidad del plano de planta.
	Realiza la exportación de la información a un formato legible en AutoCAD®.

Tabla 8.3 Opciones de dibujo

8.3.1. Filete

Para crear el arco de filete solo es necesario ingresar el radio en las unidades definidas del usuario y definir la opción de selección. La selección creará más de un arco de filete si la opción está habilitada como se muestra en la ilustración **8.18**.

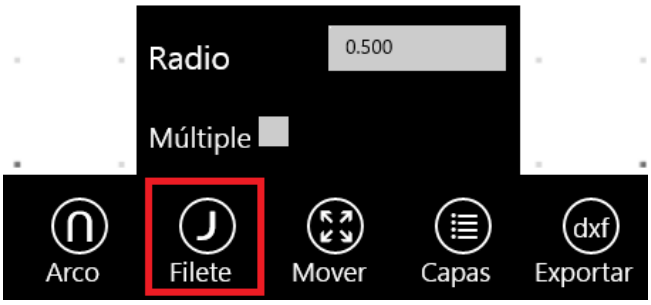


Ilustración 8.18. Interfaz de filete

8.3.2. Manejo de capas

Existen dos modos de capas (**VER ILUSTRACIÓN 8.19**), uno que se enfoca en el trazo y otro que se enfoca en el acabado.

En la capa de trazo, el usuario puede controlar la visibilidad de los siguientes elementos:

- Retícula
- Etiquetas de dimensionamiento
- Diagonales
- Etiqueta de área

En la capa de acabado, el usuario puede controlar la visibilidad de los siguientes elementos:

- Retícula
- Etiquetas de dimensionamiento
- Diagonales
- Etiqueta de área
- Puertas
- Ventanas
- Simbología
- Trazo
- Diagonales del trazo

Para navegar entre capas el usuario debe seleccionar una opción trazo o acabado.

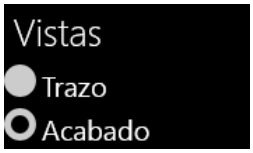


Ilustración 8.19. Visibilidad por tipo de dibujo

Para habilitar o deshabilitar la visibilidad de algún componente se da clic en las opciones disponibles (VER ILUSTRACIÓN 8.20):

Ilustración 8.20. Capas del dibujo

8.3.3. Muros, puertas y ventanas

Una vez terminado el acabado, el usuario puede definir muros, puertas y ventanas al dar clic sobre cualquier etiqueta de muro (VER ILUSTRACIÓN 8.21).

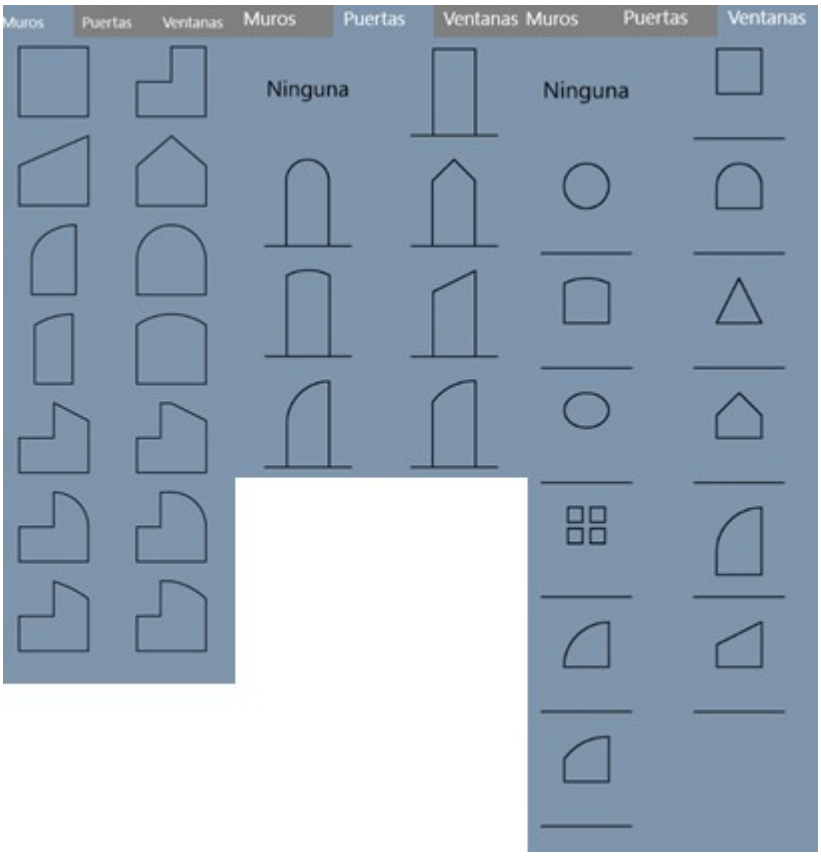


Ilustración 8.21. Interfaz de muros, puertas y ventanas

Al dar clic en cualquiera de los elementos se abre el editor de alzado en cualquiera de los tres modos seleccionados.

8.3.4. Cambio de nivel de piso y plafón

La siguiente interfaz permite modificar el desnivel del *NPT* (nivel de piso terminado) del espacio seleccionado. La ventana de edición se muestra en la ilustración **8.22**:

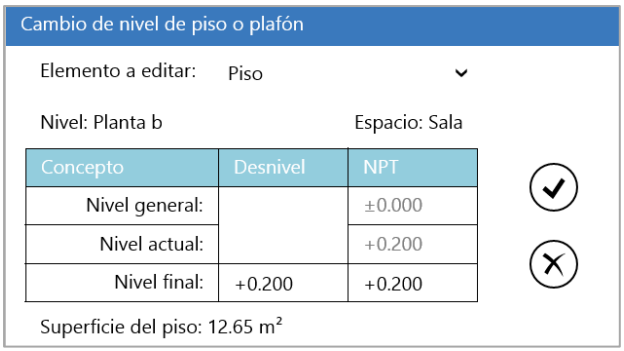


Ilustración 8.22. Ventana de cambio de nivel o de plafón

En el selector de elemento a editar el usuario define si cambiará el *NPT* del piso o del plafón.

En la tabla se muestra una comparación de niveles y el usuario escribe el nuevo valor.

En la parte inferior se calcula la nueva área del nivel.

Los cambios se aplican al dar clic en aceptar ✓.

8.4. Alzado

En esta sección (**VER ILUSTRACIÓN 8.23**) se lleva a cabo todos los procesos relacionados con el alzado de un muro. Cuenta con la siguiente distribución:

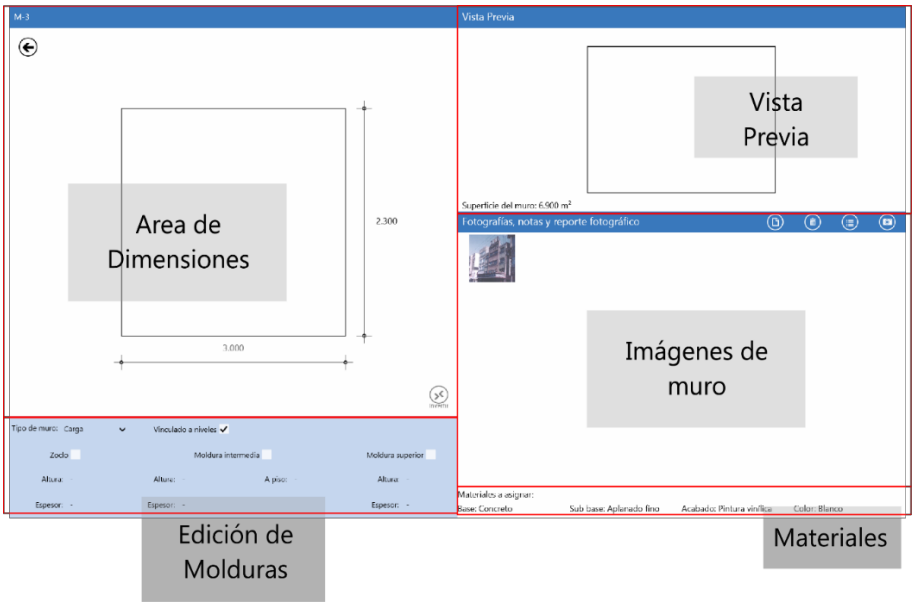


Ilustración 8.23. Distribución del área de alzado

En el área de dimensiones se define una plantilla para cada uno de los diferentes tipos de muro, la cual incluye todas las dimensiones que pueden afectar al muro, de tal forma que al editarlas se refleje el cambio en la sección de vista previa. Cada tipo de muro cuenta con su propia plantilla que contiene sus reglas y limitaciones específicas.

Las dimensiones (VER ILUSTRACIÓN 8.24) que se encuentren en color gris son dimensiones que por especificaciones del tipo de muro no se pueden editar y se obtienen de algún proceso anterior o se calculan utilizando alguna otra dimensión o dimensiones. La dimensión correspondiente a la base del muro siempre se encontrará en gris ya que esta dimensión es heredada de la geometría de acabado por lo cual se definió previamente.

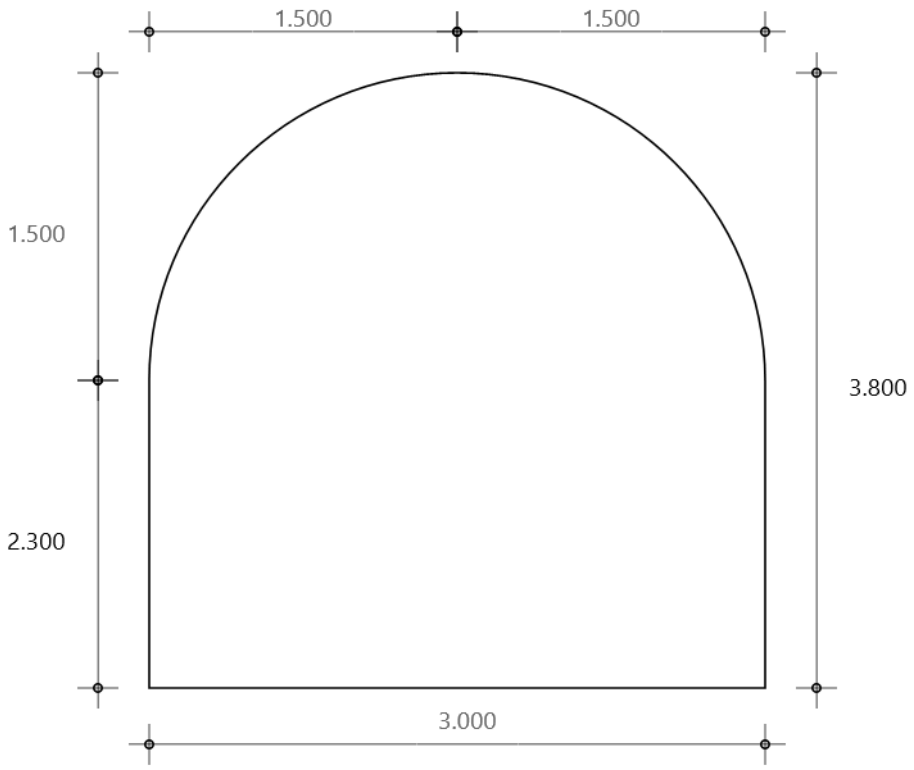
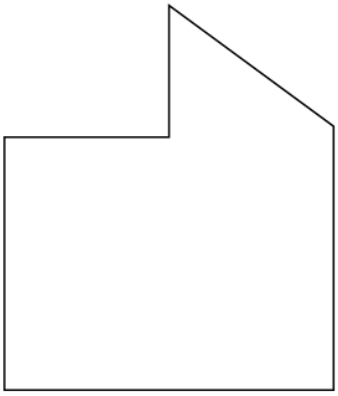


Ilustración 8.24. Dimensiones de muro con arco

Adicionalmente para los tipos de muros que no sean simétricos (VER ILUSTRACIÓN 8.25), esta área cuenta con un botón para invertir la figura, lo cual permite que se adapte de la mejor forma al muro en cuestión.



Superficie del muro: 7.875 m²

Ilustración 8.25. Vista previa de muro

La sección dedicada a la vista previa de encarga de dibujar en tiempo real la geometría del muro utilizando las dimensiones del usuario. Cuenta con un procedimiento que escala la imagen de forma que toda la geometría este dentro del área y se pueda visualizar la forma del muro con las proporciones correctas. En la parte inferior muestra el área actual del muro, la cual se calcula cada vez que haya algún cambio en cualquier dimensión.

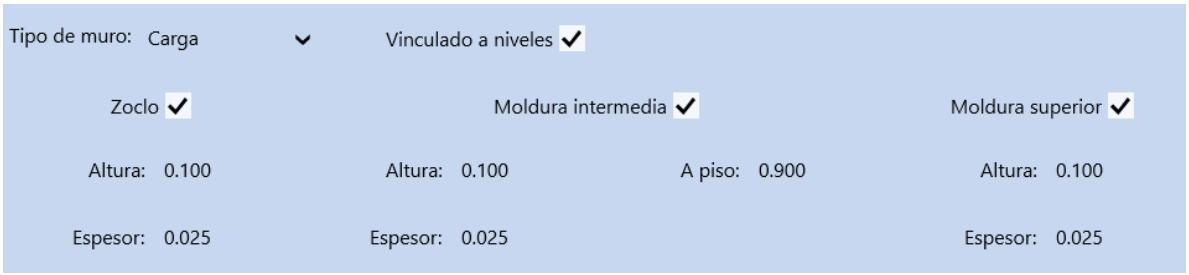


Ilustración 8.26. Opciones de alzado

El área de molduras (**VER ILUSTRACIÓN 8.26**) especifica la información necesaria para dibujar cada moldura y detalla también los valores que permitirán cuantificar el material asociado a estas. En la parte superior se presenta una lista con los diferentes tipos de muro que pueden existir. Estos tipos son:

- Carga
- Divisorio
- Contención

Algunos ejemplos de muros con molduras se muestran en las ilustraciones **8.27** y **8.28**.

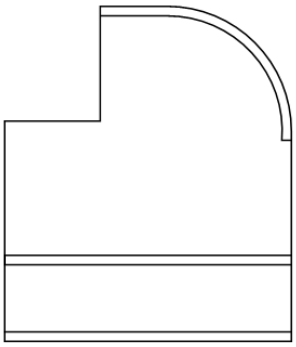


Ilustración 8.27. Muro con arco con molduras

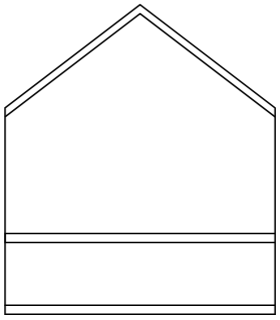


Ilustración 8.28. Muro de dos aguas con molduras

En la parte inferior de la ventana se dedica una sección a cada moldura, así como un *CheckBox* que permite habilitar o deshabilitar cada una de las molduras las cuales se dibuja al momento en la vista previa. El zoclo y la moldura intermedia se conforman de líneas rectas mientras que la moldura superior dependerá del tipo de muro, ya que cada tipo de muro tiene una forma superior especifica.

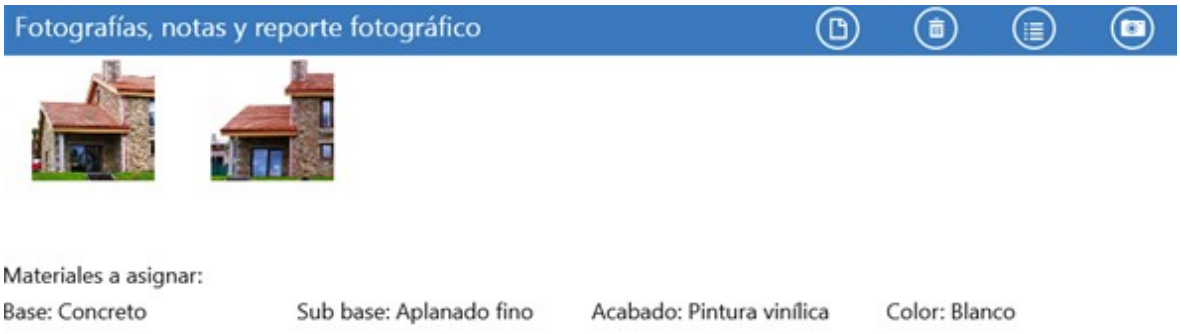


Ilustración 8.29. Fotografías de la fachada de muro

El área de imágenes (**VER ILUSTRACIÓN 8.29**) se divide en tres secciones.

En la primera sección el funcionamiento, detallado en la tabla **8.4.**

En la segunda sección se anexarán todas las imágenes que se hayan capturado en una lista. Para agregar una nota se requiere al menos una imagen.

En la última sección se muestran los materiales que se asignaron durante la creación del proyecto y que por default se asignan a los muros.

Botón	Acción
Agregar Nota	Agrega una nota a la imagen seleccionada
Eliminar Imagen	Elimina la imagen seleccionada
Crear Reporte	Crea un reporte con todas las imágenes y notas
Tomar fotografía	Toma una fotografía

Tabla 8.4 Funcionamiento de la sección de fotografías y notas.

Al presionar el botón de agregar nota se mostrará la ventana de notas (**VER ILUSTRACIÓN 8.30**):

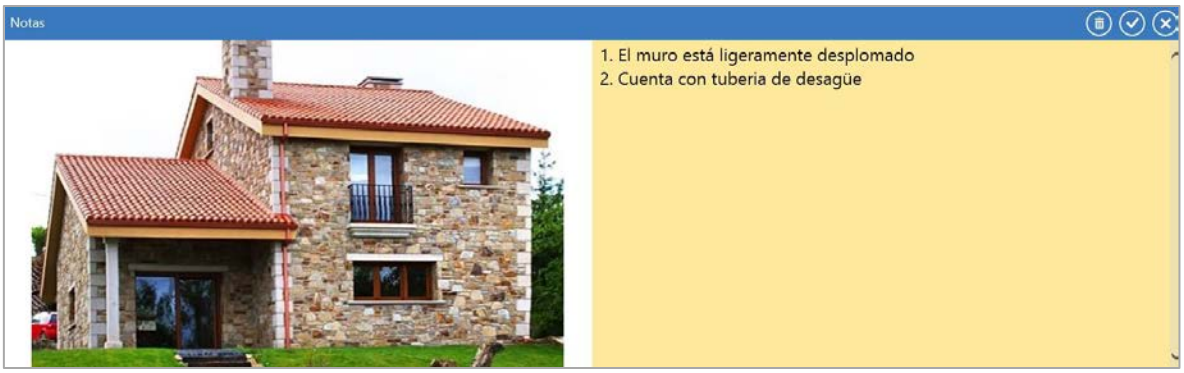


Ilustración 8.30.Interfaz gráfica de notas de muro

En esta ventana se mostrará la imagen seleccionada y un campo donde se podrá editar su contenido. Cuenta con la funcionalidad de guardar, salir sin guardar y eliminar la nota actual.

8.5. Materiales

En esta sección (**VER ILUSTRACIÓN 8.31**) se pueden administrar los materiales por muros, plafones o pisos seleccionando un nivel y un espacio.

De forma similar al módulo de levantamientos se tiene un navegador de niveles y espacios en el lado derecho.

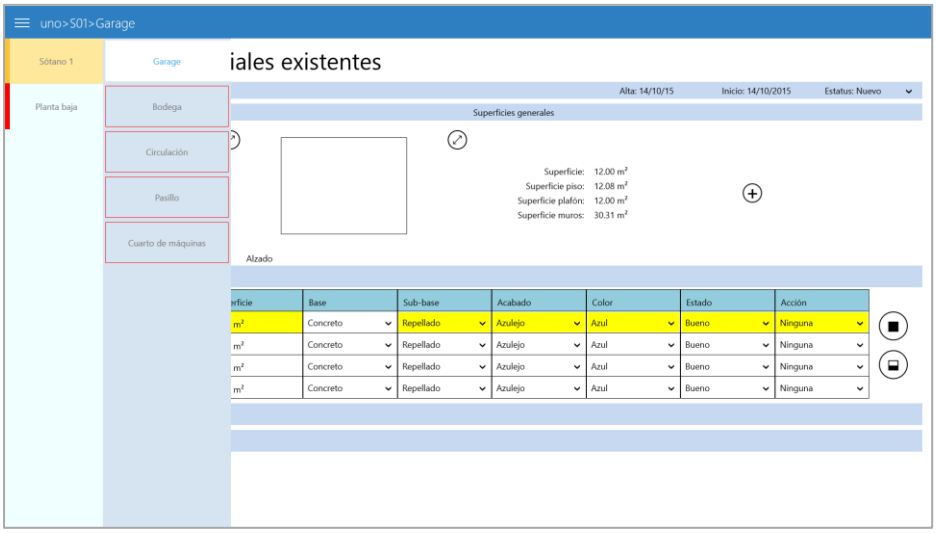


Ilustración 8.31. Distribución del área de materiales

Al seleccionar un proyecto se carga un visor de levantamientos el cual es una miniatura del módulo de levantamientos (**VER ILUSTRACIÓN 8.32**).

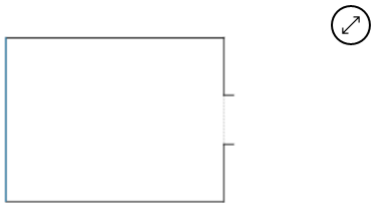


Ilustración 8.32. Visor de vista de acabado

Al dar clic en el botón de la miniatura se abre un visor de levantamientos (**VER ILUSTRACIÓN 8.33**) con el cual se puede observar con más detalle la geometría.

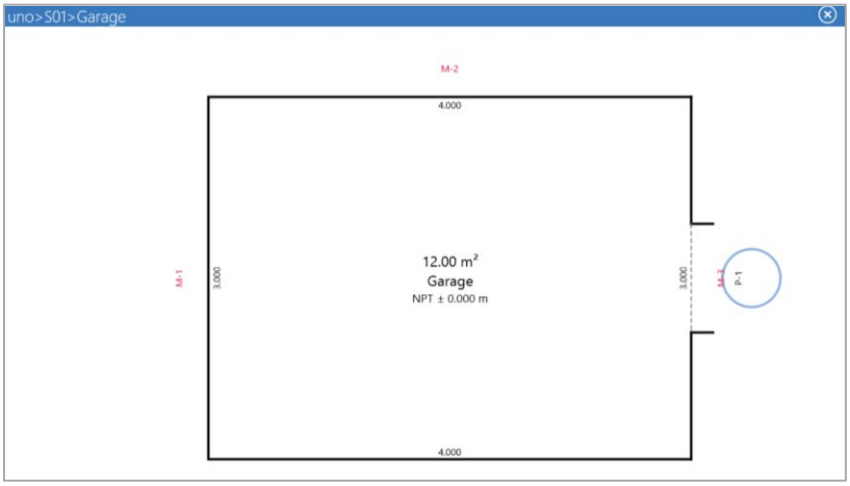


Ilustración 8.33. Visor de acabados

Una colección de tablas (**VER ILUSTRACIÓN 8.34**) con la que podemos cambiar los materiales de muros, pisos y plafones.

Ilustración 8.34. Tabla de materiales

Cuando damos clic en uno de los muros la aplicación lo resalta en amarillo y se muestra el visor de vista muro (**VER ILUSTRACIÓN 8.35**).

Ilustración 8.35. Vista previa de muros

Finalmente, a lado del visor se muestra un resumen de áreas netas en el espacio seleccionado.

Para realizar una cuantificación se da clic en el botón de cuantificar ⊕.

8.5.1. Cuantificación

Este módulo (**VER ILUSTRACIÓN 8.36**) es bastante sencillo. Cuenta con un selector de proyectos, de nivel y de espacios. Una vez generada la selección, el botón de resumen ☰ genera una tabla que permite revisar los totales.

Para finalizar, el botón de cuantificar 🗎 crea un reporte PDF en la carpeta de reportes de la aplicación.

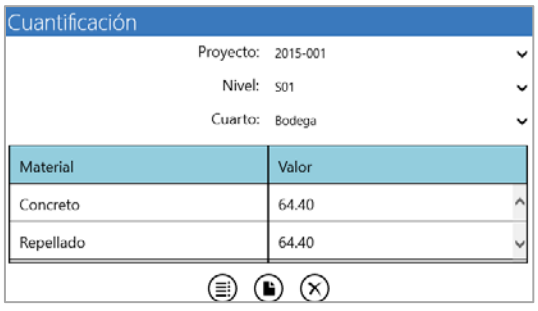


Ilustración 8.36. Ventana de cuantificación

9. Pruebas y conclusiones de la aplicación [7]

En este capítulo hablaremos de los ambientes controlados que diseñamos para probar la aplicación. Estos ambientes se generaron a través de situaciones normales y anormales buscando como objetivo crear un problema en la aplicación.

Las pruebas las categorizamos de la siguiente forma.

- **Caja negra**
  Se basan en los objetivos de la aplicación y no en la lógica de la programación
- **Caja blanca**
  Se basan en los conocimientos lógicos de la programación.
- **Pruebas de integración**
  Buscar la integración y combinación de distintos módulos de la aplicación para determinar si funcionan correctamente o no.
- **Pruebas de contenido**
  Se usaron para verificar que el contenido del sistema fuese coherente y consistente
- **Pruebas de funcionalidad**
  Este tipo de pruebas válida que la aplicación cumpla con los objetivos establecidos.

9.1. Caja negra

9.1.1. Pruebas de dibujado de trazo

En un inicio el trazo de la aplicación guardaba todos los puntos trazados por el usuario y dibujaba una línea de punto a punto.

Al terminar el dibujado se generaba la geometría y si el usuario dibujaba muy lento, los puntos que entraban a la fase de captura de puntos (**VER 2.3**) superaban el orden de los miles.

Solución:

Se crearon constantes (**VER CONSTANTES 2.6 Y 2.7**) que restringían la lectura de los puntos en un tiempo específico y se definió un método para encontrar similitud entre los puntos (**APÉNDICE A4**) y evitar leer puntos similares.

9.1.2. Pruebas de discretización de segmentos

Al terminar el proceso de vectorización, el usuario mencionaba que la figura tenía demasiados segmentos, por ejemplo, al tratar de dibujar un cuadrado se podía dibujar un polígono con cinco o más lados. Los segmentos que sobraban tenían una inclinación muy similar a los segmentos adyacentes o un tamaño muy pequeño para ser vistos como muro.

Solución:

Se desarrollaron algoritmos de segmentación que restringían ciertos segmentos mediante las constantes **2.8, 2.9, 2.10**.

9.1.3. Calibración del evento pinch

Para controlar el efecto del zoom se le pidió al usuario que realizará el evento de *zoom in* y *zoom out* mientras que nosotros variábamos la constante de *pinch* (**VER 2.5.1**), cuando el usuario noto fluidez en el zoom decidimos dar por terminada la prueba.

9.1.4. Prueba de cerrado de intersecciones

Basado en el requisito de solo permitir geometrías cerradas se mostraba un mensaje de error cuando se detectaba que la figura quedaba abierta, pero debido al proceso de discretización y vectorización de segmentos o a fallas en el trazo del usuario se presentaban demasiados casos donde la figura quedaba abierta, lo cual se volvía un proceso complicado ya que varios trazos eran rechazados. Por otro lado, no se podía procesar trazos donde la geometría presentara un error demasiado grande ya que deformaba mucho la geometría obtenida respecto a la original.

Solución:

Se creó una constante (**VER CONSTANTE 2.11**) que limitaba la distancia permitida entre el primer punto y el último punto de la geometría. Esto permitió que la aplicación no fuera tan estricta respecto al trazo del usuario y al mismo tiempo no creara geometrías muy diferentes al trazo original.

9.2. Caja blanca

9.2.1. Captura de los eventos de los punteros

Los eventos de la aplicación funcionan de tipo burbuja, en donde un evento de un control que se encuentra dentro de un contenedor ejecutaría ambos eventos y no en todos los casos requeríamos esto.

Sin las banderas de control se generaban acciones inesperadas, como que al dar clic en una etiqueta se implementaba el paneo.

Solución:

- Creación de banderas para controlar los eventos como *IsInTagEvents* y *Multi_Contact* (**VER 2.3**) con estas banderas podíamos restringir los eventos del *canvas* o indicar que la aplicación tiene más de un contacto activo.

9.2.2. Ortogonalización del muro inicial

La aplicación genera la figurada vectorizada de manera correcta, pero con una rotación que le molestaba al usuario, debido a que el primer segmento del trazo podría ser inclinado.

Solución:

Ortogonalizar el primer segmento. Esto se logró definiendo una condición en donde si la inclinación estaba más cerca de los 90° grados que de los 0° se decía que el segmento era vertical, en otro caso horizontal. Los demás segmentos dibujados se orientaban a partir de este cambio.

9.2.3. Pruebas de intercambio de diagonales

El proceso de triangulación era automático, por lo que las diagonales que generaba podrían ser difícil de medir, a petición del usuario se creó una herramienta que realizaba el intercambio de diagonales (**VER 3.1.5**).

Solución:

La herramienta de intercambio de diagonales no tenía restricciones para intercambiar la diagonal, nos dimos cuenta que en regiones cóncavas la diagonal quedaba afuera del polígono. Por lo que la diagonal solo debe cambiarse en regiones convexas donde al intercambiar la diagonal no haya un cambio en el área de la región.

9.2.4. Pruebas de segmentos de arcos

El dibujado de los arcos de filete tenían restricciones que no nos dimos cuenta hasta las pruebas del usuario, los casos encontrados fueron:

- Un filete se crea a partir de la longitud ingresada por del usuario, sin ninguna restricción el filete se dibujaba y cambiaba la longitud de los muros residuales a un valor negativa.
- Un filete reducía la longitud de un muro a un valor muy pequeño, el usuario definió como longitud de mínima el ancho de muro más pequeño utilizado en la construcción, 9.8 cm.

- Un filete en un vértice de tipo boca (**APÉNDICE C1**) se dibujaba en sentido de las manecillas del reloj se notaba que el arco estaba forzado y tenía una apariencia de chipote, en estos casos debía ser dibujado en sentido anti horario.

  Solución:

Antes de dibujar el filete validar si la longitud de los segmentos después del proceso será mayor a la longitud mínima. En caso de ser menor pero mayor o igual a 0, se elimina el segmento y se reordenan los segmentos dibujados, en otros casos los filetes no deben dejar dibujarse.

También se debe validar el vértice que une los segmentos para saber si se dibujara en sentido horario o en sentido anti horario.

9.3. Pruebas de integración

9.3.1. Prueba de comunicación de alzado y acabado

Los elementos dibujados en alzado como puertas y ventanas debían ser visibles en la vista de acabados, para esto se definió un nodo en el XML en acabado que pudiera referenciar la información de alzado sin tener que copiarla dos veces.

El nodo de acabado se muestra en la ilustración **9.1**.

```
<LineWall Name="M-1" Longitud="4.85" TagInsPt="275.537567138672,382.938751220703@-90"
"275.537565183689,-53.608868214464" Edited="4" AlzadoStatus="116">
  <Vano Name="P-1" Type="Doors" Nivel="PBA" />
  <Vano Name="P-2" Type="Doors" Nivel="PBA" />
  <Vano Name="N-1" Type="Windows" Nivel="PBA" />
</LineWall>
```

Ilustración 9.1 Nodo XML de Vano en acabados

El nodo de alzado con la información de la puerta “P-1” se muestra en la figura **9.2**.

```
<Geometry_Alzado>
    <WallViews>
        <WallView Name="M-1" Type="0" Area="11.834" Flip="1" WallType="Carga" IsVinculado="true">
            <Moldura Type="0" Altura="0.1" Espesor="0.025" Apiso="0" Enabled="true" Longitud="4.85">
                <Material Clave="Madera" Espesor="0" Value="0" Type="9" />
            </Moldura>
            ...
            <Dimensions>
                <Dimension Name="DimWidth" Value="4.85" />
                ...
            </Dimensions>
            <Notes />
            <Materiales>
                <Material Clave="Regular" Espesor="0" Value="0" Type="4" />
                ...
            </Materiales>
        </WallView>
    <WallViews>
    <Doors>
        <Door Name="P-1" Type="1" Area="2.1" VanoType="Puerta" Abatimiento="Exterior"/>
            <Dimensions>
                <Dimension Name="DimHeight" Value="2.44" />
                ...
            </Dimensions>
            <Materiales>
                <Material Clave="Madera entablerada" Espesor="0" Value="2.1" Type="6" />
                ...
            </Materiales>
        </Door>
    <Doors>
<Geometry Alzado>
```

Ilustración 9.2 Nodo XML de la puerta "P-1" en alzado

Debimos crear una clase que encapsulara los nodos XML y los pudiera acceder de forma relativa. Las pruebas se realizaron con el depurador de Visual Studio y encontramos los siguientes casos.

- Los vanos se encimaban ¿Cual debería dibujarse?, para el caso de dos ventanas a distintas alturas.
- Si existía un error en alzado, al tratar de cargarlo se generaban una excepción que cerraba la aplicación.
- Si cambiaba la fachada los vanos ya no coincidían.

Solución:

Para el dibujado evitamos encimarlos y optamos por dibujar el que este más lejos de nivel de piso terminado.

Para los errores de carga definir una excepción y no dibujar ninguno de los vanos del muro. Si cambiaba la fachada se advertía que los vanos serían borrados y tendría que volverlos a insertar, esta decisión fue para mantener la integridad de la información. Sin este cambio se generaba basura en el XML.

9.3.2. Pruebas de comunicación entre dimensión de alzado y visor de alzado

Dentro de la ventana de alzado (**VER 8.3.3**) se tiene un visor que muestra una miniatura del muro proporcionado, para probarlo fue necesario realizar primero pruebas de caja negra ingresando valores aleatorios en las dimensiones.

Los errores encontrados fueron:

- Cuando la proporción de la figura era estilo *portrait* (**APÉNDICE J1**). El muro se dibujaba a fuera del control
- Si algún cálculo de la figura tenía como entrada un NAN, la aplicación se detenía

Las pruebas de caja blanca nos ayudaron a encontrar las líneas de código que merecían ser validadas o que ameritaban un tratamiento de excepción.

Solución:

Para el visor se evaluaba el caso de que la figura fuera *portrait* o *landscape* (**APÉNDICE J1 Y J2**) y dependiendo el caso se calculaban el factor de escala que realiza el zoom extendido del visor.

Para evitar los posibles errores, se detenía la visualización si una dimensión no era válida, en las pruebas de contenido se hablará de las validaciones que utilizan las dimensiones.

9.3.3. Pruebas de comunicación con módulos de materiales

Estas pruebas generaron grandes cambios en la aplicación. Debido que su implementación fue al final, nos dimos cuenta que no solo requeríamos leer la información del XML, sino que debía existir una coherencia en la información. Estos errores eran que existían muros sin materiales y espacios sin fachadas de muros.

Dentro de los ajustes tenemos:

- Crear un muro témplate para que, aunque el usuario no haya elegido un muro exista uno al momento de cuantificar. Por default la aplicación creará una fachada de muro ortogonal con los acabados definidos en el catálogo de materiales del proyecto.
- Al crear un nuevo proyecto se crea un catálogo default de materiales y se asignan los valores por default a muros, puertas, ventanas, acabados y otros.

Para más información revisar los temas (**7** y **8.5**)

9.4. Pruebas de contenido

9.4.1. Pruebas de validación de contenido del formulario

Estas pruebas la realizaron los usuarios que probaron la aplicación y el objetivo era ver qué tipo de información ingresaban en la aplicación.

Los resultados que obtuvimos fueron:

- En algunos campos que pensábamos que eran numéricos el usuario deseaba ingresar otros caracteres como en los campos teléfonos
- Los usuarios comentaron que no había opciones predeterminadas y existían muchas opciones a elegir en el caso de los controles de tipo combo box
- La información se perdía si cerraban la aplicación sin entrar algún punto de guardado de la aplicación.
- Errores al cargar imágenes, cuando se detenían en mitad del proceso de selección
- Accesibilidad para navegar entre páginas
- En caso de las dimensiones ingresaban valores irreales que ocasionaban resultados erróneos

Solución:

Para los proyectos se generó una interfaz llamada *IProjectSection* que se define en la ilustración **9.3**. Con esta interfaz y una clase auxiliar llamada *ProjectUtility* de tipo estática creamos la validación de todos nuestros controles con métodos extendidos que usaban como entrada a la interfaz.

```
25 references
public interface IProjectSection
{
    50 references
    Control[] ControlOrder { get; }

    71 references
    Project_Xml Xml { get; }
    13 references
    void AddEvents();
    13 references
    void RemoveEvents();
}
```

Ilustración 9.3 Interfaz de sección de proyectos

Las acciones para solucionar la entrada de la aplicación fueron.

- Revisar el tipo de entrada y validar la información ingresada, en caso de ser inválida insertar un campo default
- Seleccionar una opción default para los *ComboBox*
- Crear eventos de edición en donde al terminar una edición guardar el campo en el XML, aplicaba para *CheckBox*, *ComboBox*, *TextBox* e imágenes
- En el caso de las imágenes fue validar la salida cuando el usuario cancelaba la selección o toma de la imagen
- Crear una barra de navegación (**VER ILUSTRACIÓN 8.2**)

Para el caso de dimensionamientos fue más sencillo, usamos las reglas de jerarquía de dimensionamiento (**VER 4.4.2**) y como límite estos dos criterios:

- Altura máxima definida por la altura de piso a techo
- Longitud máxima definida por la longitud del muro

Para las dimensiones que podrían cambiar estos valores se decidió bloquearlas.

9.4.2. Pruebas de visualización de unidades

El manejo de unidades generó un problema cuando optamos por guardarlas en las unidades definidas por el usuario y en el formato de despliegue. Nos dimos cuenta que al cambiar de sistema ingles a metros se perdía información y después de un tiempo terminábamos con datos irreales.

Solución:

Guardar todo en metros sin truncar la dimensión y dejar la conversión al control que se encargaba de imprimir en el formato acordado.

9.4.3. Pruebas de distribución del contenido en distintas resoluciones

Primero realizamos estas pruebas usando la herramienta de Visual Studio para cambiar el aspecto de la pantalla, una vez que obtuvimos los resultados agradables decidimos realizar el *deployment* en distintas maquinas descritas en las conclusiones.

Los únicos ajustes en estas pruebas eran de estética reacomodando los controles en pantalla.

9.5. Pruebas de funcionalidad

9.5.1. Pruebas de triangulación

Una sección importante dentro de la aplicación es la triangulación de geometrías, ya que permite el funcionamiento de procesos posteriores. Se generó una colección de geometrías con el objetivo de abarcar todos los casos posibles. Observamos que las figuras comunes no presentaban ningún problema, pero conforme aumentábamos la complejidad de la geometría se presentaban triangulaciones incompletas o inservibles para nuestros objetivos.

Solución:

Se analizaron las geometrías antes de intentar ser trianguladas y dependiendo de sus ángulos en los vértices se utilizaron diferentes algoritmos para simplificar la geometría y finalmente usar algoritmos más sencillos (**VER 3.1.1-3.1.3**).

9.5.2. Pruebas de dimensionamiento

En un principio se contempló dimensionar la figura conforme las medidas eran introducidas por el usuario, pero el proceso se volvía lento en figuras complejas. Al mismo tiempo era confuso para el usuario ya que no se pudo definir un orden que facilitara el dimensionamiento, por lo que las variantes en cómo se introducía la información generaba figuras deformadas que nada tenían que ver con el espacio real.

Solución:

Se puso como regla que el primer segmento fuera el único que permitiera escalar globalmente la figura. Adicionalmente el proceso de dimensionamiento solo se ejecutará al terminar de ingresar todas las dimensiones, por lo cual las etiquetas cambian de color cuando han sido modificadas por el usuario dando por hecho que han sido aceptadas por el usuario (**VER 3.2**).

9.5.3. Pruebas de cálculos de área

Al tener la triangulación adecuada, el cálculo del área fue sencillo. Debido al método que se utilizó para calcular el área (**VER 4.6**), se presentó el problema de que el usuario introducía valores con tres dígitos de precisión y en algunos casos no eran suficientes para obtener un área precisa.

El mayor problema de este resultado era que ciertas cantidades podían causar confusión al usuario debido a que el mismo esperaba un área entera sin dígitos de precisión.

Solución:

Se analizó el resultado y en caso de poder redondear se mostró al usuario un valor que fuera más sencillos de tomar en cuenta. Por ejemplo, mostrar 5 m^2 en lugar de mostrar 4.999 m^2.

9.5.4. Pruebas de lectura del distanciometro laser

En un principio se presentaron problemas de compatibilidad del dispositivo con un sistema de arquitectura *ARM*, ya que no existe driver del dispositivo para esta arquitectura. Posteriormente se presentaron diversos casos de formatos de salida del dispositivo, los cuales podían alterar los resultados obtenidos y en algunos casos mandar error dentro de la aplicación por usar valores no permitidos.

Solución:

Se utilizaron dispositivos con arquitecturas *x86* y *x64* para obtener la compatibilidad con el dispositivo. Para los diversos casos de salida del dispositivo se diseñó un *parser* que procesara todos los formatos, para evitar que el usuario tuviera que usar alguno en específico.

9.5.5. Pruebas de lectura de GPS

Para el requerimiento de anexar la ubicación del levantamiento en la aplicación se requería utilizar el GPS del dispositivo. Se detectaron casos donde los dispositivos no contaban con

GPS o debido a que se usa en interiores la señal de GPS puede fallar mucho. Para evitar depender completamente del GPS se utiliza la red a la que se encuentre conectada el dispositivo, pero se han dado casos donde por alguna falla en la red, la ubicación varía mucho.

9.5.6. Pruebas de lectura de la brújula

Para facilitar la integración de la planta se pensó utilizar la ubicación del norte en cada espacio, pero después de probar varias veces los valores obtenidos variaban mucho. Se anexo un mensaje al usuario donde se indicaba que la brújula tenía que ser calibrada, pero dicha calibración podía ser muy tardada y en ciertos casos debido a interferencias por alguna estructura o con algún otro dispositivo.

Solución:

Debido a que estos errores no se pudieron evitar se optó por utilizar los vanos para integrar la planta. Y además se permite que si el dispositivo no tiene brújula o la precisión no es adecuada, el usuario pueda modificar este valor manualmente utilizando algún valor previamente conocido.

9.6. Conclusiones

En un principio se analizaron los tiempos para realizar un levantamiento arquitectónico utilizando las herramientas que existían en el mercado, con el objetivo de tener una referencia durante el desarrollo de la aplicación.

Con el uso de estas herramientas nos dimos cuenta que las aplicaciones que dibujaban a mano alzada y generaban un proceso desconocido de ortogonalización, tenían una mejor aceptación con respecto a las que no lo implementaban. Realizando una encuesta con los arquitectos de nuestra oficina llegamos a la conclusión de que esas aplicaciones tenían una interfaz amistosa pero los resultados no eran los que ellos deseaban, ya que carecían de precisión e incluso comentaban que preferían usar *AutoCAD®*.

Con base a esto decidimos que la aplicación debía dibujar a mano alzada con una vectorización más rigurosa que ofreciera una salida útil para el usuario. Es por eso que al finalizar nuestra primera versión del motor gráfico se logró mejorar el tiempo para hacer un levantamiento de espacios básicos hasta 15 veces más y aunque teníamos una mejor precisión que las aplicaciones probadas (VER TABLA 1.1), nos dimos cuenta que se presentaban casos donde la aplicación no podía solucionarlos, como por ejemplo espacios con huecos.

Para solucionar estos problemas se crearon herramientas y metodologías de dibujo, así como las mejoras en los algoritmos utilizados con el fin de evitar que la aplicación fallara en esos casos, entre las que tenemos:

- Intercambio de diagonales (VER 3.1.5)
- Movimiento de etiqueta (VER 2.6.7)
- Creación de arcos (VER 4.2 Y 4.3)

- Notas (VER 4.4.3)
- Algoritmo de triangulación mixta (VER 3.1.3)
- Algoritmo de dimensionamiento (VER 3.2)

Estas herramientas se crearon principalmente usando los conocimientos obtenidos en las áreas de computación gráfica, geometría analítica, estructura de datos y diseño asistido por computadora. Debido a la naturaleza de la aplicación nuestro trabajo requirió de un desarrollo interdisciplinario, el cual consistió en las encuestas realizadas y en el aprendizaje de la terminología básica utilizada en arquitectura. Adicionalmente contamos la ayuda de un especialista en el área el cual nos proporcionó bibliografía y resolvió dudas sobre la marcha.

Con estos cambios obtuvimos la información y vista deseada en la aplicación para que el levantamiento cumpliera con los requerimientos que utiliza un arquitecto al realizar este proceso. Al finalizar estos cambios los tiempos que se utilizaban para realizar el levantamiento dentro de la aplicación aumentaron, así como la complejidad de la misma. Aún después de estos aumentos de tiempo y complejidad, la aplicación seguía siendo superior a las metodologías que se utilizan actualmente, cumpliendo con los objetivos planteados en la aplicación.

Sabiendo que *AutoCAD®* es el software de Diseño Asistido por Computadora más utilizado actualmente, se propuso obtener como salida de la aplicación un formato compatible con este para permitir que los usuarios a los que va dirigida la aplicación no tuvieran que realizar algún proceso adicional durante el proceso. Al elegir este formato de salida aprovechamos el previo conocimiento que teníamos con el API de desarrollo de *AutoCAD®* y creamos un *addin* que lee y carga de manera automática la información exportada de la aplicación en un dibujo de *AutoCAD®*.

Utilizando un distanciómetro *Bluetooth®* se mejoró considerablemente la precisión y la rapidez de una medición. El problema inicial que tuvimos fue la conexión del dispositivo ya que el driver del distanciómetro solo funcionaba en máquinas con arquitectura *x86* y *x64*, eliminando las arquitecturas *ARM* de nuestra lista. Otra dificultad consistió en procesar los diferentes formatos que enviaba el distanciometro al utilizar el Sistema Inglés, ya que el dispositivo presentaba diferentes variaciones que en un principio desconocíamos y tuvimos que realizar un *parser* para cada caso.

Originalmente se planteó usar el *Framework* utilizado por las *Windows Store Apps®* debido a la portabilidad que presenta y a la similitud que tiene con el *Framework* completo de *.NET®*, ya que teníamos experiencia en este *Framework* y eso nos facilitaría el desarrollo de la aplicación. Pero durante el desarrollo se observaron las diferencias entre *Frameworks* y algunas de estas retrasaron el desarrollo, ya que tuvimos que implementar herramientas con comportamientos similares.

Entre las herramientas que implementamos tenemos:

- Zoom y paneo

- Log en excepciones, ya que no se pueden usar métodos asíncronos dentro de un bloque catch y la forma de guardar y modificar archivos es estrictamente asíncrona.
- El control de una tabla estilo *GridView*.
- Manejo de pop ups, ya que no se acostumbran en las aplicaciones metro.
- Acceso a los ancestros, ya que no se contaba con los métodos del *Framework* completo.
- Implementación de la función de deshacer para el dibujo de levantamientos
- Manejo de propiedades dinámicas uniendo los nombres de los controles a los campos de los XML

Durante el desarrollo de las interfaces se presentó otro problema ya que la adaptabilidad a diferentes tamaños de pantallas generaba distintos *layouts*, por lo que para garantizar una presentación agradable en los distintos dispositivos probados, utilizamos más tiempo acomodando los controles.

Los dispositivos en los que probamos la aplicación fueron

- *Surface RT®*
- *Surface Pro 2®*
- *Surface Pro 4®*
- PC *Windows 8.1®* monitor 25”
- PC *Windows 10®* monitor 17”
- Laptop *Windows 8.1®* Asus con pantalla touch de 19”
- Laptop *Windows 10®* Dell XPS 15”

Al implementar estas diferencias logramos que no se afectaran los objetivos de la aplicación y que mantuviera el diseño inicial de una aplicación de la tienda de *Windows®*. La principal ventaja de haber elegido esta plataforma es la existencia de Universal Apps como sucesor, la cual no presentará demasiados cambios durante la migración y en un futuro permitir que nuestra aplicación se ejecute en cualquier dispositivo con *Windows 10®*. Además gracias a las distintas pruebas que realizamos en distintos dispositivos logramos un diseño, que puede ser escalado sin realizar ningún cambio en la versión para *Windows 10®.*

Apéndice

A Funcionalidad de puntos [3]

1 Distancia entre dos puntos

Se define a la distancia entre dos puntos como la distancia "ordinaria" (que se mediría con una regla) entre dos puntos de un espacio euclidiano. Se define en la ecuación **A.1.**

$$\text{} = \sqrt{(\text{}_1 - \text{}_0)^2 + (\text{}_1 - \text{}_0)^2} \qquad (\text{}.1)$$

Donde $\text{}$ es la distancia, $\text{}_0$ y $\text{}_0$ son las coordenadas del punto inicial y $\text{}_1$ y $\text{}_1$ las coordenadas del punto final.

2 Punto medio

Se define como el punto que se encuentra a la misma distancia de cualquier de los puntos definidos en el espacio euclidiano. Se define en la ecuación **A.2.**

$$\text{}_{\text{}}(\text{},\text{}) = (\frac{\text{}_0 + \text{}_1}{2}, \frac{\text{}_0 + \text{}_1}{2}) \qquad (\text{}.2)$$

Donde $\text{}_{\text{}}$ es el punto medio, $\text{}_0$ y $\text{}_0$ son las coordenadas del punto inicial y $\text{}_1$ y $\text{}_1$ las coordenadas del punto final.

3 Igualdad

Si los puntos tienen las mismas coordenadas. Se define en la ecuación **A.3**

$$\text{}_0(\text{}_1, \text{}_1) = \text{}_1(\text{}_1, \text{}_1) \ \therefore \text{}_0 = \text{}_1 \qquad (\text{}.3)$$

Donde $\text{}_0$ es el punto de comparación y $\text{}_1$ el punto a comparar.

4 Similitud

Se define una constante de error absoluto (**VER ECUACIÓN A.4**) y se calcula la diferencia absoluta por componente del punto. Si alguna diferencia supera el error absoluto se puede decir que los puntos son diferentes, en otro caso serán iguales (**VER CONDICIÓN A.5**).

$$\text{}_{\text{}}(\text{},\text{}) = \text{}(\text{}_{\text{}}(\text{},\text{}) - \text{}_{\text{}}(\text{},\text{}))$$

$$(\text{}.4)$$

$$\text{}_0 \approx \text{}_1 \mid \text{}_{\text{}} \leq \text{} \qquad (\text{}.5)$$

Donde $\text{}_0$ es el punto de comparación, $\text{}_1$ el punto a comparar y $\text{}_{\text{}}$ el error absoluto.

B Vectores [8]

Un *vector* es una representación formada por una magnitud, una dirección, un sentido y que sigue un conjunto de reglas de combinación. La magnitud se define por la distancia entre dos puntos y la dirección y el sentido son definidos por el ángulo que forma el vector

respecto con el eje de las abscisas.

1 Propiedades

- **X**

Coordenadas del vector en X

- **Y**

  Coordenadas del vector en Y

- **Magnitud**

  La magnitud del vector, queda definida en la ecuación **B.1**

$$|\blacksquare| = \sqrt{\blacksquare^2 + \blacksquare^2} \qquad (\blacksquare.1)$$

  Donde $|\blacksquare|$ es el modulo del vector.

- **Vector unitario**

  El vector en forma unitaria, queda definido en la ecuación **B.2**

$$\hat{\blacksquare} = \frac{\blacksquare}{|\blacksquare|} \qquad (\blacksquare.2)$$

  Donde $\blacksquare$ $\blacksquare\blacksquare$ $\blacksquare\blacksquare$ $\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare$, $|\blacksquare|$ es el modulo del vector y $\hat{\blacksquare}$es el vector unitario

- **Ángulo**

  El ángulo del vector respecto al eje de las abscisas, para este cálculo usamos la ecuación **B.3**, el ángulo se guardó en radianes

$$\theta = \blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\frac{\blacksquare}{\blacksquare} \qquad (\blacksquare.3)$$

  Donde θ es el ángulo del vector

2 Producto punto o producto escalar

El producto escalar se define en la ecuación **B.4** y el resultado siempre es una constante y $\bar{\blacksquare}$ son los términos aplicar el producto escalar, $\blacksquare\blacksquare$ $\bar{\blacksquare}\blacksquare$ las componentes del vector $\blacksquare$ y $\blacksquare\blacksquare$ las componentes del vector $\bar{\blacksquare}$ El resultado de la operación es la constante $\blacksquare$.

$$\blacksquare\ \bar{\blacksquare} = \blacksquare\blacksquare + \bar{\blacksquare}\blacksquare \qquad (\blacksquare.4)$$ Donde

3 Producto cruz

El producto cruz en dos dimensiones (**Ver Ecuación B.5**) nos ayuda a encontrar la normal del vector, la cual se utilizó para validar el sentido de rotación al dibujar.

$$\blacksquare\ \blacksquare\ \bar{\blacksquare} = \begin{vmatrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & 0 \\ \blacksquare & \blacksquare & 0 \end{vmatrix} \qquad (\blacksquare.5)$$

Donde $\blacksquare$ y $\bar{\blacksquare}$son los términos aplicar el producto cruz, $\blacksquare\blacksquare$ $\bar{\blacksquare}$las componentes del vector $\blacksquare$ y $\blacksquare\blacksquare$ $\bar{\blacksquare}$las componentes del vector $\bar{\blacksquare}$ El resultado de la operación es vector con componente en k.

C Vértices [9]

Un *vértice* es un punto especial en donde una o más líneas se intersectan. Un polígono simple solo intersecta dos líneas por vértice. A diferencia de un punto, un vértice forma un ángulo el cual se mide en la intersección de sus segmentos (**VER ILUSTRACIÓN C.1**).

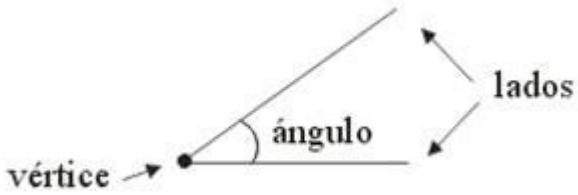


Ilustración C.1 Ángulo del vértice

Los vértices se pueden clasificar en los siguientes:

1 Cóncavo

También llamado de bocas o *reflex* pertenece a un vértice $\hat{v}_i$ donde el ángulo interno es obtuso y la diagonal formada del polígono simple P definido por $[\hat{v}_{i-1}, \hat{v}_{i+1}]$ se encuentra fuera de los límites de P.

Donde P es el polígono, un $\hat{v}_{i-1}$ es el vértice anterior y $\hat{v}_{i+1}$el vértice siguiente.

2 Convexo

También llamado de oreja pertenece a un vértice $\hat{v}_i$ donde el ángulo interno es agudo y la diagonal formada del polígono simple P definida por $[\hat{v}_{i-1}, \hat{v}_{i+1}]$ se encuentra totalmente en P.

Donde P es el polígono, un $\hat{v}_{i-1}$ es el vértice anterior y $\hat{v}_{i+1}$el vértice siguiente.

Otra característica del vértice es la normal que se genera dependiendo del sentido en el que los vértices son recorridos. El sentido de la normal se define en la tabla **C.1**.

	Vértice Cóncavo	Vértice Convexo
Sentido horario (CW)	$(-)$	$(+)$
Sentido anti-horario (CCW)	$(+)$	$(-)$

Tabla C.1 Sentidos de orientación

La normal se calcula con el producto cruz (**VER ECUACIÓN C.1**) del vector definido $\bar{v}_1 = (\hat{v}_i - \hat{v}_{i-1})$ contra el vector definido $\bar{v}_2 = (\hat{v}_i - \hat{v}_{i+1})$

$$\bar{n} = \bar{v}_1 \times \bar{v}_2 \qquad (\hat{n}.1)$$

D Triángulo

Un *triángulo* se define como un polígono simple de tres lados y la suma de sus ángulos internos es 180°.

Todos los triángulos tienen vértices convexos y se definen en la ecuación **D.1**:

$$\hat{t} = (\hat{v}, \hat{v}, \hat{v}) \qquad (\hat{t}.1)$$

Donde T es el triángulo y A, B, C son tres coordenadas en el espacio.

1 Perímetro

Se define como la suma de las longitudes de los segmentos que forman al triángulo (**VER ECUACIÓN D.1**).

$$\text{P} = \overline{\text{AB}} + \overline{\text{BC}} + \overline{\text{CA}} \qquad (\text{D}.3)$$

Donde P es el perímetro, $\overline{\text{AB}}$ la distancia del punto A al punto B, $\overline{\text{BC}}$ la distancia del punto B al punto C y $\overline{\text{CA}}$ la distancia del punto C al punto A

2 Semi-perímetro

Es la mitad del perímetro definida por la ecuación **D.4**

$$\text{s} = \frac{\text{P}}{2} \qquad (\text{D}.4)$$

Donde s es el semi-perímetro y P el perímetro.

3 Centroide

Es el punto donde se intersectan las tres medianas del triángulo, el centroide se calcula usando la ecuación **D.5**

$$\text{C} = \frac{\text{A} + \text{B} + \text{C}}{3} \qquad (\text{D}.5)$$

Donde C es el centroide y A, B, C los puntos que definen al triángulo.

4 Área [9]

Es la región limitada por el perímetro del triángulo. Se calculó utilizando la fórmula de Herón. (**VER ECUACIÓN D.6**)

$$\text{Área} = \sqrt{(\text{s} - \overline{\text{AB}})(\text{s} - \overline{\text{BC}})(\text{s} - \overline{\text{CA}})} \qquad (\text{D}.6)$$

Donde s es el semi-perímetro, $\overline{\text{AB}}$ la distancia del punto A al punto B, $\overline{\text{BC}}$ la distancia del punto B al punto C y $\overline{\text{CA}}$ la distancia del punto C al punto A.

5 Puntos resueltos

Define la cantidad de puntos que se encuentran en *posición correcta* (**VER 2.2 DIMENSIONAMIENTO**) utilizando las dimensiones que el usuario proporcionó.

6 Ángulos internos [3]

Es el ángulo que forma uno de los vértices del triángulo y queda definido como el ángulo entre los vectores de los segmentos que definen al vértice.

El ángulo entre dos vectores se define en la ecuación **D.6**.

$$\propto = \cos^{-1}\left(\frac{\text{u} \cdot \overline{\text{v}}}{|\text{u}| \times |\overline{\text{v}}|}\right) \qquad (\text{D}.6)$$

Donde u y $\overline{\text{v}}$ son los vectores que entran al vértice y $\propto$ es el ángulo de los vectores.

E Geometría [9]

1 Polígono simple

Un polígono P es llamado *simple* (**VER ILUSTRACIÓN E.1**). si ninguno de sus segmentos es intersectado por otro segmento del polígono. Un polígono simple cerrado divide al plano geométrico en dos regiones, región interior y región exterior.

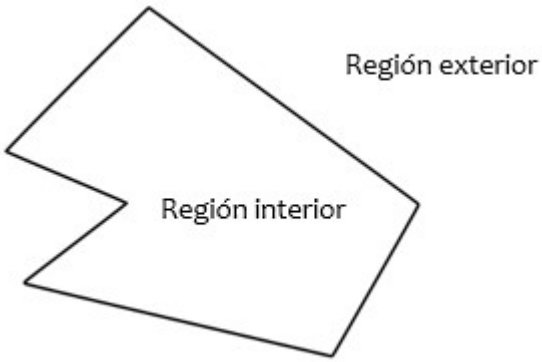


Ilustración E.1 Regiones de un polígono simple

Clasificación de los polígonos simples:

- **Polígono convexo**

  Los ángulos interiores miden menos de 180° y todas las diagonales se dibujan en la región interior del polígono (**VER ILUSTRACIÓN E.2**)

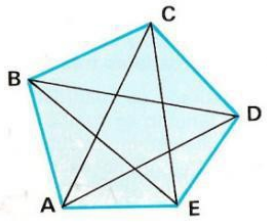


Ilustración E.2 Polígono convexo de cinco lados

- **Polígono convexo regular**

  Los ángulos interiores son iguales, en la ilustración **E.3** se muestra un ejemplo de un polígono convexo regular.

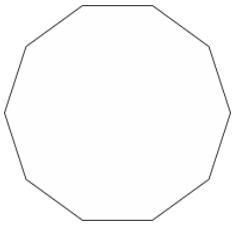


Ilustración E.3 Decágono

- **Polígono convexo irregular**

Los ángulos interiores son diferentes, en la ilustración **E.4** se muestra un ejemplo de un polígono convexo irregular

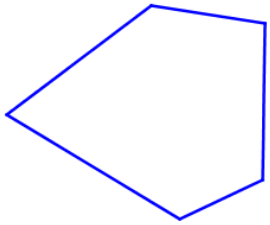


Ilustración E.4 Pentágono Irregular

- **Polígono cóncavo o no convexo**

  Existen algunos ángulos interiores que miden más de 180° y existen diagonales en la región exterior del polígono (**VER ILUSTRACIÓN E.5**)

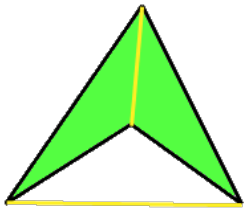


Ilustración E.5 Polígono cóncavo

2 Polígono complejo

Un polígono P es llamado *complejo* o *no simple* si uno o más de sus segmentos son intersectados por otro segmento del polígono (**VER ILUSTRACIÓN E.6**)

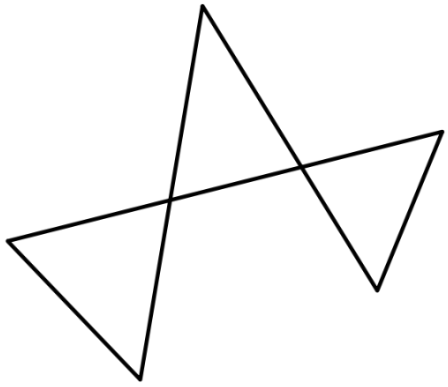


Ilustración E.6 Polígono Complejo

3 Ecuación punto-pendiente de la recta

La pendiente de una recta es la *tangente* (**VER ILUSTRACIÓN E.7**) del ángulo que forma la recta con la dirección positiva de las abscisas.

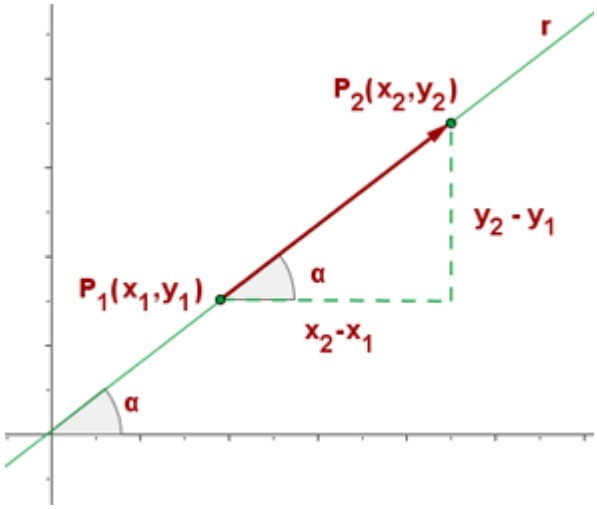


Ilustración E.7 Diagrama de recta entre dos puntos

La pendiente se define en la ecuación **E.1**:

$$\text{pendiente} = \text{pendiente}\ \frac{(y_2 - y_1)}{(x_2 - x_1)} \qquad (\text{E}.1)$$

Donde pendiente es la pendiente, x_1 y y_1 son las coordenadas de un punto de la línea y x_2 y y_2 son las coordenadas de otro punto de línea.

La ecuación de la recta usando la pendiente y un punto en la ecuación **E.2**:

$$y = \text{pendiente}(x - x_1) + y_1 \qquad (\text{E}.2)$$

Donde pendiente es la pendiente, x_1 y y_1 son las coordenadas de un punto de la línea, x la variable de las abscisas y y la variable de las ordenadas.

4 Intersección de dos rectasError! Reference source not found.

La intersección de dos líneas busca el punto que tienen en común dos rectas. La condición para encontrar la intersección es que las rectas no sean paralelas.

Igualando la ecuación de punto-pendiente de las dos rectas en función de X tenemos la ecuación **E.1**.

$$x = \frac{m_1 \times x_1 - m_2 \times x_2 + y_1 - y_2}{m_1 - m_2} \qquad (\text{E}.3)$$

Donde m_1 es la pendiente de la primera recta, m_2 es la pendiente de la segunda recta x_1 y y_1 son las coordenadas de un punto de la primera recta y x_2 y y_2 son las coordenadas de un punto de la segunda recta.

Finalmente sustituimos x en alguna de las dos ecuaciones de la recta (**APÉNDICE E.3**), se puede encontrar el punto de intersección.

5 Puerto de vista [10]

Refiriéndonos a computación gráfica, el puerto de vista (**VER ILUSTRACIÓN E.8**) es la región rectangular de la ventana donde la imagen es dibujada. El puerto de vista es medido en coordenada de ventana, las cuales reflejan la posición de los pixeles en la pantalla relativa a la esquina inferior izquierda de la ventana. Hay que tener en cuenta que todos los vértices ya se les han aplicado las matrices de modelo, vista y proyección.

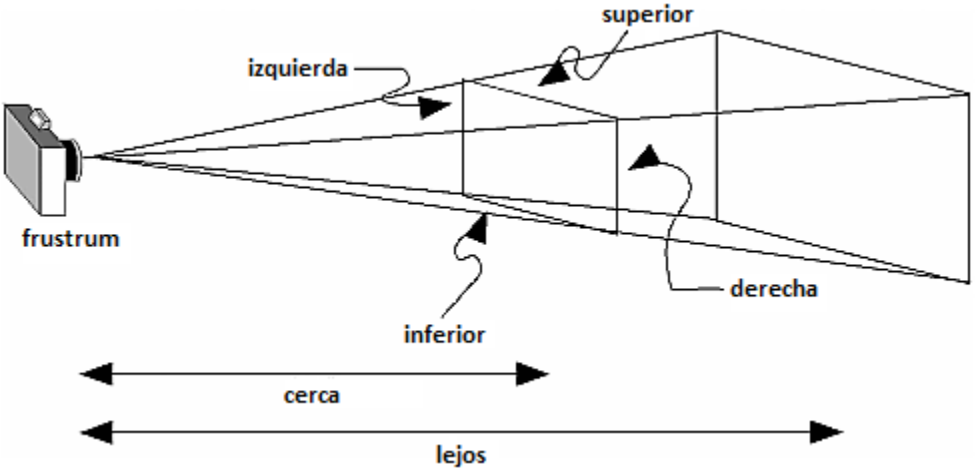


Ilustración E.8 Puerto de vista

6 Transformada homogénea de traslación

La transformada de traslación se define en la matriz **E.4**

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix} \qquad (E.4)$$

Donde T_x es la distancia a desplazar en X y T_y la distancia a desplazar en Y

7 Transformada homogénea de rotación

La transformada de rotación se define en la matriz **E.5**

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \cos\theta & 0 \\ -\sin\theta & \sin\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (E.5)$$

Donde θ es el ángulo de rotación

8 Transformada homogénea de escalamiento

La transformada de escalamiento se define en la matriz **E.6**

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (E.6)$$

Donde S_x es el factor de escala para X y S_y es el factor de escala para Y

9 Arco de semi-circunferenciaError! Reference source not found. [9]

En geometría un arco es cualquier curva continua que une dos puntos, siendo sus propiedades las siguientes:

- **Ángulo**
  180° o π rad
- **Área**

Se define en la ecuación **E.7**

$$\text{Área} = \frac{\pi}{2}r^2 \qquad (E.7)$$

Donde r es el radio del arco

- **Longitud de arco**

Se define en la ecuación **E.8**

$$L = \pi r \qquad (E.8)$$

Donde r es el radio del arco y L la longitud del arco

- **Centro**

  El punto medio entre el punto inicial y final del arco, se define en la ecuación **E.9**

$$C = \frac{P_i + P_{i+1}}{2} \qquad (E.9)$$

Donde C es el centro del arco y P_i es el punto inicial del arco y P_{i+1} el punto final del arco.

- **Cuadrante**

  Un punto del arco se define en la ecuación **E.10**

$$P(x,y) = (P_x + r \times \cos\frac{\pi}{2}, P_y + r \times \sin\frac{\pi}{2}) \qquad (E.10)$$

Donde P_x es la componente en X del centro de la circunferencia, P_y es la componente en Y del centro de la circunferencia y r es el radio de la circunferencia.

- **Arco menor y arco mayor**

  El arco menor (**VER ILUSTRACIÓN E.9**) es el arco que se forma de la circunferencia que tiene una menor longitud al arco complementario

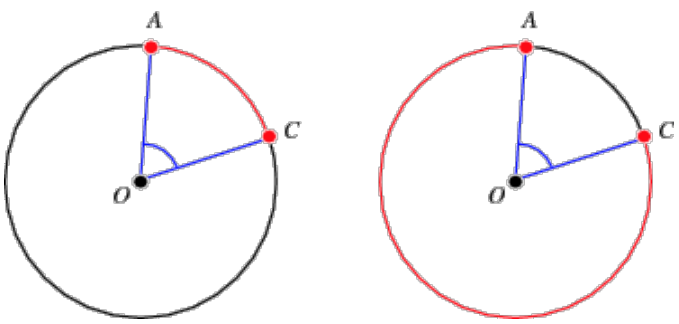


Ilustración E.9 Arco menor y arco mayor

10 Arco de circunferencia a partir de 3 puntos

Cuando no se cuenta con el radio, para crear un arco solo es necesario contar con tres puntos que pasen por la circunferencia.

Los puntos son A, B y C:

$$\text{A}(\text{x}_1, \text{y}_1), \qquad \text{B}(\text{x}_2, \text{y}_2), \qquad \text{C}(\text{x}_3, \text{y}_3)$$

Mediante un análisis geométrico, podemos encontrar el centro del arco, el radio y el ángulo que forma el arco. El análisis es el siguiente.

Primero se calculan las rectas (**APÉNDICE E3**) definidas en la ecuación **E.11** y **E.12**:

$$\text{L}_{\text{AB}}: \text{y} = \text{m}_{\text{AB}}(\text{x} - \text{x}_1) + \text{y}_1 \qquad (\text{E}.11)$$

$$\text{L}_{\text{BC}}: \text{y} = \text{m}_{\text{BC}}(\text{x} - \text{x}_2) + \text{y}_2 \qquad (\text{E}.12)$$

Se sabe que el centro del arco está a cierta distancia del punto medio del segmento AB y BC y que es perpendicular al punto medio de AC.

Las rectas que son perpendiculares tienen pendientes negativas inversas. Las pendientes de estas rectas se definen en las ecuaciones **E.13** y **E.14**:

$$\text{m}'_{\text{AB}} = -\frac{1}{\text{m}_{\text{AB}}} \qquad (\text{E}.13)$$

$$\text{m}'_{\text{BC}} = -\frac{1}{\text{m}_{\text{BC}}} \qquad (\text{E}.14)$$

Las rectas tangentes pasaran por el punto medio, de los segmentos AB y BC. Las ecuaciones de las rectas se definen en las ecuaciones **E.15** y **E.16**:

$$\text{L}'_{\text{AB}}: \text{y} = \text{m}'_{\text{AB}}\left(\text{x} - \left(\frac{\text{x}_1 + \text{x}_2}{2}\right)\right) + \left(\frac{\text{y}_1 + \text{y}_2}{2}\right) \qquad (\text{E}.15)$$

$$\text{L}'_{\text{BC}}: \text{y} = \text{m}'_{\text{BC}}\left(\text{x} - \left(\frac{\text{x}_2 + \text{x}_3}{2}\right)\right) + \left(\frac{\text{y}_2 + \text{y}_3}{2}\right) \qquad (\text{E}.16)$$

Las rectas se intersectan en el punto $\text{O}(\text{x}_0, \text{y}_0)$

Donde O es el centro del arco, x_0 la componente X del centro del arco y y_0 la componente en Y del centro del arco.

Igualando en x las ecuaciones **E.15** y **E.17** se despeja x para obtener la ecuación **E.17**

$$\text{m}'_{\text{AB}}\left(\text{x}_0 - \left(\frac{\text{x}_1 + \text{x}_2}{2}\right)\right) + \left(\frac{\text{y}_1 + \text{y}_2}{2}\right) = \text{m}'_{\text{BC}}\left(\text{x}_0 - \left(\frac{\text{x}_2 + \text{x}_3}{2}\right)\right) + \left(\frac{\text{y}_2 + \text{y}_3}{2}\right) \qquad (\text{E}.17)$$

$$-\frac{1}{\text{m}_{\text{AB}}}\left(\text{x}_0 - \left(\frac{\text{x}_1 + \text{x}_2}{2}\right)\right) + \left(\frac{\text{y}_1 + \text{y}_2}{2}\right) = -\frac{1}{\text{m}_{\text{BC}}}\left(\text{x}_0 - \left(\frac{\text{x}_2 + \text{x}_3}{2}\right)\right) + \left(\frac{\text{y}_2 + \text{y}_3}{2}\right) \qquad (\text{E}.17.1)$$

$$-\frac{\text{x}_0}{\text{m}_{\text{AB}}} + \frac{\text{x}_1 + \text{x}_2}{2\text{m}_{\text{AB}}} + \frac{\text{y}_1 + \text{y}_2}{2} = -\frac{\text{x}_0}{\text{m}_{\text{BC}}} + \frac{\text{x}_2 + \text{x}_3}{2\text{m}_{\text{BC}}} + \frac{\text{y}_2 + \text{y}_3}{2} \qquad (\text{E}.17.2)$$

$$-\frac{\text{x}}{\text{m}_{\text{AB}}} + \frac{\text{y}}{\text{m}_{\text{AB}}} = \frac{\text{x}_2 + \text{x}_3}{2\text{m}_{\text{BC}}} + \frac{\text{y}_2 + \text{y}_3}{2} - \frac{\text{x}_1 + \text{x}_2}{2\text{m}_{\text{AB}}} - \frac{\text{y}_1 + \text{y}_2}{2} \qquad (\text{E}.17.3)$$

$$= \frac{\frac{-\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}\mathbf{\Phi}_0 + \mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}\mathbf{\Phi}_0}{\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}}}{2\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}} \quad (\mathbf{\Phi}.17.4)$$

$$2(\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}} - \mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}})\mathbf{\Phi}_0 = \mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}(\mathbf{\Phi}_3 - \mathbf{\Phi}_1) + \mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}(\mathbf{\Phi}_2 + \mathbf{\Phi}_3) - \mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}(\mathbf{\Phi}_1 + \mathbf{\Phi}_2) \quad (\mathbf{\Phi}.17.5)$$

$$\mathbf{\Phi}_0 = \frac{\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}(\mathbf{\Phi}_3 - \mathbf{\Phi}_1) + \mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}(\mathbf{\Phi}_2 + \mathbf{\Phi}_3) - \mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}}(\mathbf{\Phi}_1 + \mathbf{\Phi}_2)}{2(\mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}} - \mathbf{\Phi}_{\mathbf{\Phi}\mathbf{\Phi}})} \quad (\mathbf{\Phi}.18)$$

Y finalmente *y* quedaría definida en la ecuación **E.19**:

$$\mathbf{\Phi}_0 = \mathbf{\Phi}'_{\mathbf{\Phi}\mathbf{\Phi}}\left(\mathbf{\Phi}_0 - \frac{\mathbf{\Phi}_1 + \mathbf{\Phi}_2}{2}\right) + \frac{\mathbf{\Phi}_1 + \mathbf{\Phi}_2}{2} \quad (\mathbf{\Phi}.19)$$

Una vez calculado el centro se puede calcular la distancia a cualquiera de los puntos del arco (**VER ECUACIONES E.20 Y E.21**) usando el radio y el ángulo.

$$\mathbf{\Phi} = \sqrt{(\mathbf{\Phi}_1 - \mathbf{\Phi}_0)^2 + (\mathbf{\Phi}_1 - \mathbf{\Phi}_0)^2} \quad (\mathbf{\Phi}.20)$$

$$\propto = \mathbf{\Phi}\mathbf{\Phi}\mathbf{\Phi}\mathbf{\Phi}\mathbf{\Phi}\mathbf{\Phi}\left(\frac{\bar{\mathbf{\Phi}} \cdot \bar{\mathbf{\Phi}}}{|\bar{\mathbf{\Phi}}| \times |\bar{\mathbf{\Phi}}|}\right) \quad (\mathbf{\Phi}.22)$$

Donde $\mathbf{\Phi}$ es el radio del arco y $\propto$ el ángulo medido entre los vectores $\bar{\mathbf{\Phi}}$ y $\bar{\mathbf{\Phi}}$(**APÉNDICE D6**).

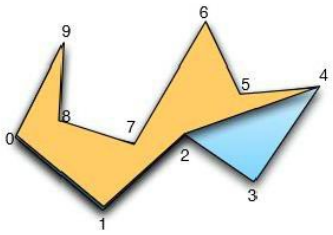
F Triangulación "Ear Clipping" [11]

Las características de este algoritmo son:

- Es el algoritmo más simple, de orden $\mathbf{\Phi}(\mathbf{\Phi}^3)$ y se aplica a polígonos simples
- Una oreja es un vértice convexo y está definido por tres vértices consecutivos
- El segmento de recta entre $\mathbf{\Phi}_{\mathbf{\Phi}-1}$ y $\mathbf{\Phi}_{\mathbf{\Phi}+1}$ se llama diagonal
- El vértice $\mathbf{\Phi}_{\mathbf{\Phi}}$ se llama punta de oreja
- Un triángulo consiste en una sola oreja, aunque se puede poner como punta de la oreja cualquiera de los tres vértices
- Un polígono de cuatro lados tiene al menos dos orejas que no traslapan
- Si tenemos una oreja en un polígono con n≥4 vértices y removerla tendremos un polígono de n-1 vértices y repetir el proceso

El algoritmo se describe a continuación (**VER ILUSTRACIÓN F.1**) [12]:

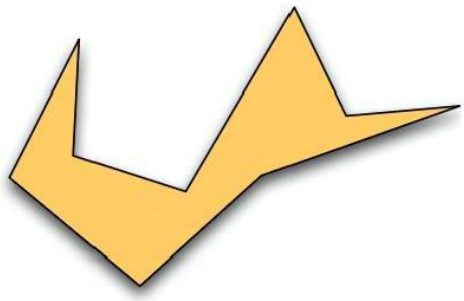
1. Almacenar el polígono como una lista doblemente ligada de manera a poder eliminar rápidamente las puntas de las orejas
2. Cada vértice $\mathbf{\Phi}_{\mathbf{\Phi}}$ y su triángulo correspondiente ($\mathbf{\Phi}_{\mathbf{\Phi}-1}$, $\mathbf{\Phi}_{\mathbf{\Phi}}$, $\mathbf{\Phi}_{\mathbf{\Phi}+1}$)
3. Generar una diagonal en la oreja encontrada y remover la punta de la oreja
4. Repetir el proceso hasta tener una figura de tres lados



Vértices convex C = { 0, 1, 3, 4, 6, 9 }

Vértices reflex R = { 2, 5, 7, 8 }

Vértices oreja E = { 3, 4, 6, 9 }



Eliminar oreja 3 :

Primer triángulo en la triangulación T0 = { 2, 3, 4 }

Ilustración F.1 Algoritmo Ear Clipping

G Gestos táctiles [13]

1 Tap

Se presiona la pantalla de forma rápida en algún elemento para ejecutar la acción de clic de un puntero.

2 Doble tap

Se presiona la pantalla de forma rápida dos veces sobre algún elemento para ejecutar la acción de doble clic.

3 Deslizar

Se presiona la pantalla y sin desprender el dedo se desliza hacia cualquier dirección para realizar el desplazamiento, tal y como se muestra en la ilustración **G.1**

Ilustración G.1 Deslizamiento con el índice derecho

4 Pellizcar (pinch in o pinch out)

Se presiona la pantalla con dos dedos para realizar un pellizco o separar los dedos. Al juntar los dedos el evento es llamado *pinch in* y al separarlos *pinch out*, tal y como se muestra en la ilustración **G.2**

Ilustración G.2 Ejemplo de pinch in

5 Swipe (deslizamiento rápido)

Se presiona la pantalla y se desliza el dedo sobre ella de manera rápida. Se utiliza para desplegar las opciones de *Windows®*, tal y como se muestra en la ilustración **G.3**

Ilustración G.3 Deslizamiento vertical y horizontal

H Periféricos

1 Norma ISO 16331-1

La norma tiene como objetivo regular la precisión y el rango de las medidas obtenidas mediante la tecnología láser con las siguientes condiciones:

- Condiciones favorables (espacios interiores):
  - Luz ambiental tenue (3 Klux)
  - Pared con pintura blanca
  - Temperatura ambiente
- Condiciones desfavorables (espacios exteriores)
  - Luz del sol (30 Klux)
  - Pared con pintura blanca
  - Temperatura total operativa ± 3°C
- Otras condiciones:
  - Condiciones no favorables a superficies con fuerte o poca reflexión como (vidrios, metales, concreto mojado)
  - Mediciones a elementos con ángulos de inclinación.

2 Norma IP65 [14]

Los equipos electrónicos son utilizados en diversas aplicaciones y tienen que trabajar de una manera segura durante un largo período de tiempo y bajo condiciones ambientales adversas. El polvo y la humedad no se pueden evitar siempre, así como la presencia de cuerpos extraños. Las distintas clases de protección fijan, en qué medida se puede

exponer un aparato eléctrico en condiciones ambientales adversas, sin ser dañado o sin representar un riesgo de seguridad o para la salud.
La norma se define de la siguiente manera, el primer digito marca la resistencia al polvo y el segundo al agua (VER ILUSTRACIÓN H.1)

Ilustración H.1 Lectura de la norma

Niveles de protección		
IP	**Protección contra el Ingreso de objetos solidos**	**Protección contra agua**
0	sin protección	sin protección
1	Protección contra cuerpos extraños con diámetro >50mm	Protegido contra gotas de agua que caen verticalmente
2	Protección contra cuerpos extraños con diámetro >12mm	Protegido contra gotas de agua que caen inclinado (15° respecto de la vertical)
3	Protección contra cuerpos extraños con diámetro >2,5mm	Protegido contra agua pulverizada (hasta 60° respecto de la vertical)
4	Protección contra cuerpos extraños con diámetro >1mm	Protegido contra agua pulverizada
5	Protección completa contra contacto, protección contra sedimentaciones de polvos en el interior	Protegido contra los chorros de agua (desde todas las direcciones)
6	Protección completa contra contacto, protección contra penetración de polvo	Protegido contra la penetración de agua en caso de invación pasajera
7		Protegido contra la penetración de agua sumergiendolo
8		Protegido contra la penetración de agua sumergiendolo por un período indefinido
9		Protegido contra la penetración de agua de todas direcciones también en caso de una presión alta contra el chasis. (limpiadora de alta presión o de chorro de vapor, 80-100 bar)

Tabla H.1 Niveles de protección

3 Brújula

La brújula es un elemento de orientación que funciona detectando los campos magnéticos naturales de la Tierra. Debido a que nuestro planeta tiene un núcleo de hierro líquido y parte cristal sólido por su presión gravitacional, se genera un movimiento en la parte líquida del núcleo produciendo el fenómeno del campo magnético de la Tierra.

Como todo cuerpo magnético, la Tierra tiene dos polos, el Polo Sur y el Polo Norte (VER ILUSTRACIÓN H.2). Con respecto a la ubicación geográfica el eje magnético se encuentra desfasado, a esto se le llama declinación. La declinación no es suficiente para desviar la orientación.

Actualmente la declinación es de 1600km y cada año aumenta 40km. Se cree que hace 780, 000 años la declinación era de 180° al polo Norte Geográfico.

Las brújulas no apuntan al polo Norte Geográfico sino al polo Norte magnético, definido como el lugar donde el campo magnético es perpendicular a la superficie.

Ilustración H.2 Representación del norte geográfico y el norte magnético

I Lenguajes de programación

1 XAML

Conocido por sus siglas en ingles "e**X**tensible **A**pplication **M**arkup **L**anguage" un lenguaje de marcado basado en *XML* y desarrollado por *Microsoft®*. Este lenguaje facilita la forma de crear aplicaciones que sean complejas y con contenido abundante. Anteriormente se utilizaba el *API* de *Windows® Forms*, en donde los controles tenían limitantes y era un problema crear un controlador que se adaptara a las necesidades del desarrollador.

2 WPF (Windows Presentation Foundation)

Es un sistema gráfico que se encarga desplegar la interfaz gráfica haciendo uso de las tecnologías de aceleración por hardware. Anteriormente una aplicación User32 contenía un conjunto de controles que facilitaban el desarrollo al usuario, pero sin adaptabilidad a distintas resoluciones y una personalización limitada.

Entre las características de *WPF* encontramos:

- Interfaz orientada al diseño web

  En lugar de fijar las coordenadas de los controles de forma relativa se especifica la orientación y distribución de los controles
- Dibujo de controles abundante en lugar de dibujar controles con pixeles directamente se tratan con primitivas y figuras sencillas, además permite el manejo de transparencias y apilar capas con distintas opacidades
- Un formato de texto enriquecido (*Rich Text Format*)

Permite mostrar texto con un formato específico

- Animaciones
- Soporte de Audio y Video
- Estilos y plantillas de diseño
- Comandos

  Una abstracción de evento que permite ejecutar una acción sin importar la forma en la que fue ejecutado
- Aplicación basada en páginas

3 MVC

MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo mismo, *Model, Views & Controllers*, si lo prefieres en inglés. En este artículo estudiaremos con detalle estos conceptos, así como las ventajas de ponerlos en marcha cuando desarrollamos.

4 Windows Store Apps y Universal Apps

Las aplicaciones de *Windows® Store Apps* fueron agregadas en la versión de *Windows® 8*. Se ejecutan en un entorno virtual llamado *Windows® RT* (**VER ILUSTRACIÓN I.1**), el cual es una versión simplificada de *Windows®*. Estas tienen un estilo sencillo llamado metro. El objetivo de este es mostrar la mayor información posible manteniendo el diseño simple. Con el paso del tiempo hubo problemas y disgustos de los usuarios con la compatibilidad de *Windows® RT* con el entorno de *Windows®*. La solución de este problema fue la creación de un nuevo formato de aplicación llamado *Windows® Universal App* (**VER ILUSTRACIÓN I.2**). Usan el mismo *API* y como característica principal pueden ejecutarse en cualquier dispositivo que corra *Windows®*, como una *Surface*, una PC, un *smartphone* o un *Xbox*.

Ambas aplicaciones utilizan *.NET®* en distintas versiones del *framework*.

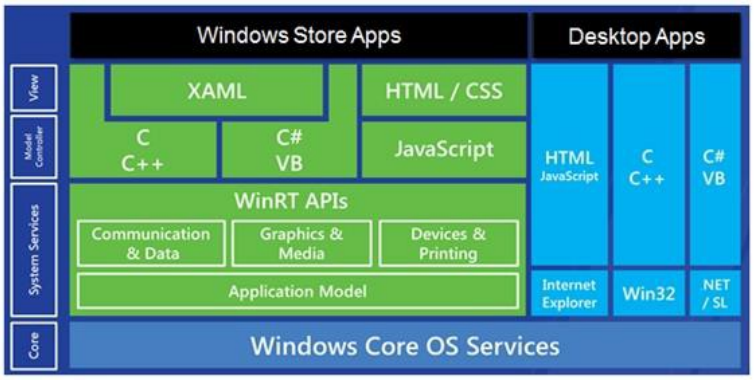


Ilustración I.1 Diagrama de capas de Windows Store Apps y Desktop Apps

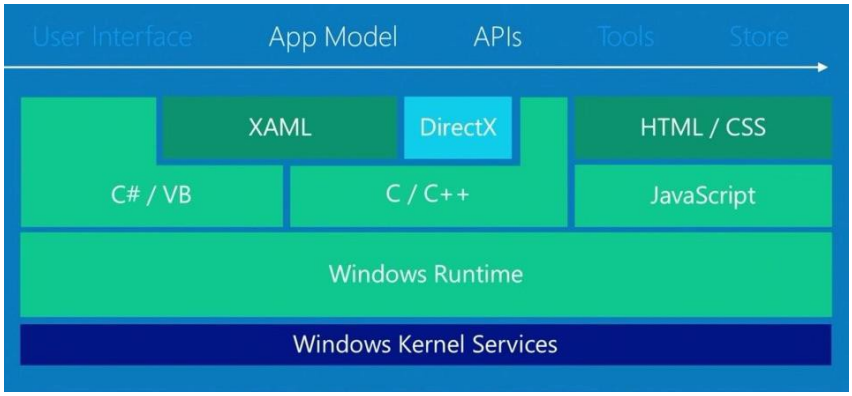


Ilustración I.2 Diagrama de capas de Windows Universal Apps

J Glosario

1 BIM

Building **I**nformation **M**odeling, también llamado modelado de información para la edificación, es el proceso de generación y gestión de datos del edificio durante su ciclo de vida. Utilizando distintos softwares generando un modelo dinámico en tres dimensiones y en tiempo real, para disminuir la pérdida de tiempo y recursos en el diseño y la construcción.

Este proceso produce el modelo de información del edificio, que abarca la geometría del edificio, las relaciones espaciales, la información geográfica, así como las cantidades y las propiedades de los componentes del edificio.

2 Portrait

Se tiene un aspecto donde predomina el alto y no el ancho, en la ilustración **J.1** se muestran estilos de aspecto de tipo *portrait*.

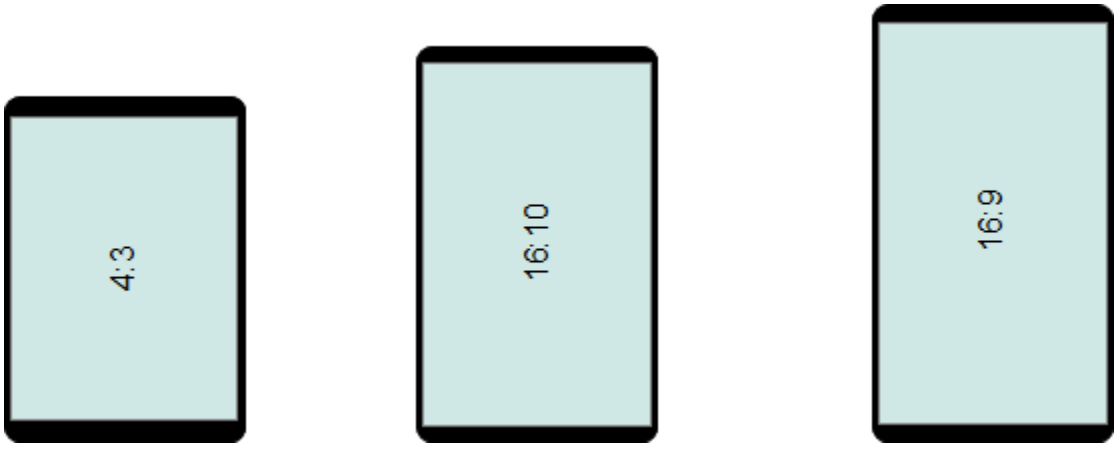


Ilustración J.1 Aspectos de tipo portrait

3 Landscape

Se tiene un aspecto donde predomina el ancho y no el alto, en la ilustración **J.2** se muestran estilos de aspecto de tipo *landscape*.

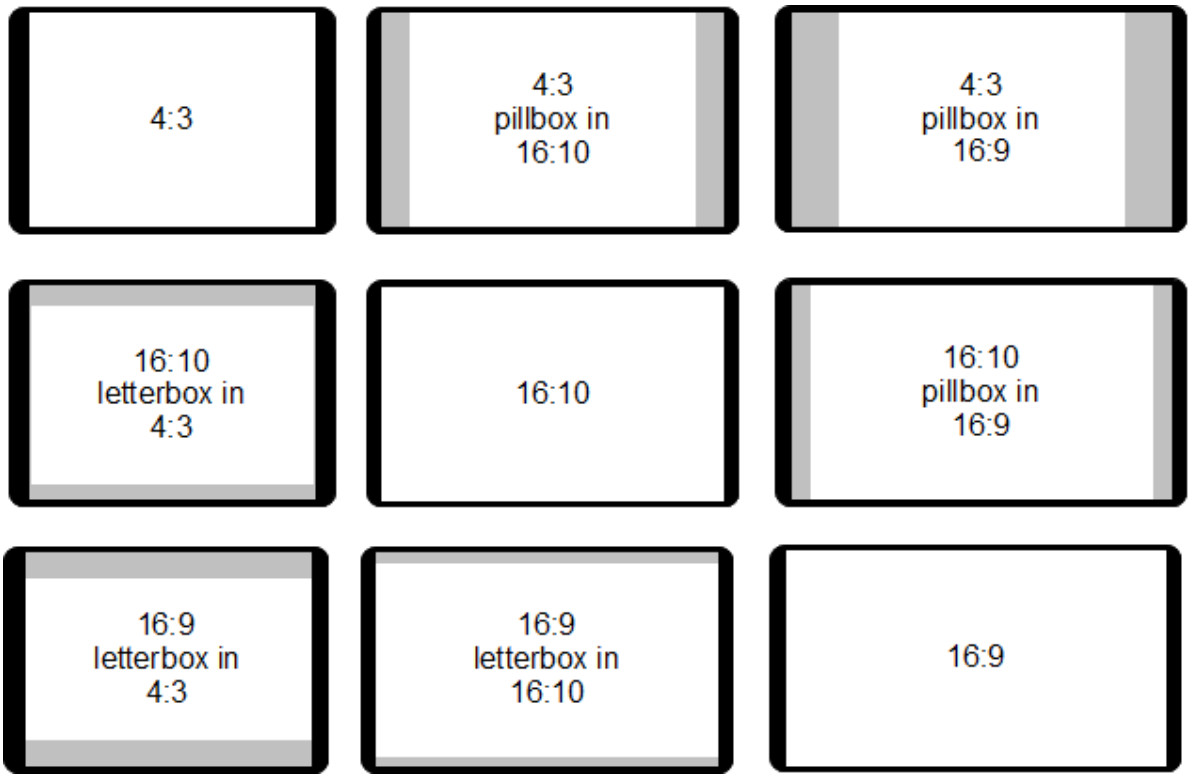


Ilustración J.2 Aspectos de tipo landscape

Referencias

[1] G. M. Antonio, Topografía básica para ingenieros, Murcia: Universidad de Murcia, 1996.

[2] M. MacDonald, «Pro WPF 4.5 in C#,» de *Windows Presentation Foundation in .NET 4.5*, New York, Apress, 2012, pp. 13,56.

[3] C. H. Lehmann, «Geometría Analítica,» New York, Limusa, 1989, pp. 99-103.

[4] R. García Diéguez y A. García Martínez, «Universidad de Sevilla, TEMA 13: El Edificio y El Muro (I),» 11 06 2006. [En línea]. Available: http://fama2.us.es/earq/mdd/construccion1/Objetos%20de%20Aprendizaje/apuntes%20tema_13_muros.pdf. [Último acceso: 13 04 2016].

[5] Microsoft, «Getting Started with Bing Maps,» 19 February 2016. [En línea]. Available: https://msdn.microsoft.com/en-us/library/ff428643.aspx.

[6] Microsoft, «Create a Bing Maps Key,» 2016. [En línea]. Available: https://www.microsoft.com/maps/create-a-bing-maps-key.aspx.

[7] J. L. M. Álvarez, «Aplicación de un Sistema Experto para el desarrollo de Sistema Evaluador del modelo Capability Maturity Model (CMM) niveles dos y tres,» de *Capitulo 5, Pruebas del Sistema*, Cholula, Puebla, Universidad de las Américas Puebla, 2004, pp. 1-15.

[8] R. Resnick, «Física Volumen I,» Continental, 2001, p. 42.

[9] E. W. Weisstein, «MathWorld,» 22 Julio 2015. [En línea]. Available: http://mathworld.wolfram.com/Vertex.html.

[10] D. Shreiner, «OpenGL Programming Guide: The Official Guide to Learning OpenGL,» de *OpenGL Programming Guide: The Official Guide to Learning OpenGL*, Addison-Wesley, 2009, p. 206.

[11] G. H. Meisters, «Polygons Have Ears,» *The American Mathematical Monthly, Vol. 82 No. 6,* pp. pp. 648-651, 1975.

[12] C. E. Jaramillo, «CIMAT, Centro de Investigación en Matemáticas A.C.,» 2 Febrero 2014 . [En línea]. Available: http://www.cimat.mx/~cesteves/cursos/comp420/pdf/22_TriangulacionPoligonos.pdf.

[13] Microsoft, «Táctil: tocar, deslizar rápidamente y mucho más,» 10 01 2016. [En línea]. Available: https://www.microsoft.com/surface/es-mx/support/touch-mouse-and-search/using-touch-gestures-tap-swipe-and-beyond?os=windows-8.1-rt-update-3&=undefined. [Último acceso: 14 04 2016].

[14] Reinmedical, «Clases de protección IP,» 27 Abril 2015. [En línea]. Available: http://www.reinmedical.com/es/conocimientos-tecnologia/clases-de-proteccion-ip.html.