DESARROLLO DE UN PROGRAMA DE DIBUJO CAD USANDO ALGORITMOS BÁSICOS.

M. A. Piedras Morales[1]
B. F. González Castillo[2]

RESUMEN.

Se desarrolló un programa de dibujo CAD, usando algoritmos elementales, mismos que se desarrollan dentro de la materia de Graficación, que corresponde al quinto semestre de la carrera de Ingeniería en Sistemas Computacionales, en el Tecnológico de Estudios Superiores de Cuautitlán Izcalli (TESCI). El programa se desarrolla como un paquete completo que permite hacer dibujos con base en figuras básicas, como líneas, círculos, elipses y curvas spline (Bezier), y a cada uno de ellos se les aplica transformaciones en dos dimensiones, como Traslación, escalación, rotación, corte y reflexión. Como se trata de dibujo vectorial, cada coordenada generada, se almacena en una matriz de superficie, en tiempo de ejecución. Posteriormente, estos datos, se pueden almacenar en archivos de acceso aleatorio, lo que garantiza que el programa, no depende de bases de datos o programas de terceros. De tal suerte que se desarrolló un algoritmo para guardar la información, y otro para recuperarla, reconstruyendo posteriormente, el dibujo completo.

ANTECEDENTES.

La materia de Graficación se encuentra dentro de la estructura curricular de la carrera de Ingeniería en Sistemas Computacionales del Tecnológico de Estudios Superiores de Cuautitlán Izcalli (TESCI), en el quinto semestre de 9. Lo que garantiza que el alumno en este punto, ya sabe programar, conoce las metodologías de desarrollo de software. Sin embargo, no ha desarrollado programas complejos. En este semestre es cuando se ha terminado el tronco común y se comienza la especialización, por lo tanto, la materia es una excelente oportunidad para introducir al estudiante en la construcción de software complejo.

En el desarrollo de este software, se utilizaron los algoritmos mínimos, sin acceder a librerías de dibujo del lenguaje de programación, procurando diseñar y desarrollar cada elemento, usando la menor cantidad de recursos del lenguaje y apegándose a la idea de crear todo lo que se pueda, partiendo de cero.

Objetivo

Desarrollar un programa de dibujo CAD, usando algoritmos de figuras elementales como línea, círculo, elipse, Bézier, y aplicando transformaciones en 2D, tales como, traslación, escalación, rotación, corte y reflexión, a cada una. El programa guarda los datos vectoriales de la imagen y lo puede reconstruir.

Objetivos específicos.

1. Implementar los algoritmos existentes de dibujo y de transformaciones en 2D, a un programa que permita dibujar las distintas figuras, cambiarles de color y aplicarles transformaciones en 2D, indistintamente.
2. Crear un algoritmo para el manejo de la matriz de superficie, que permita, seleccionar la figura que se desee, y aplicarle cualquier transformación.
3. Crear un algoritmo que permita guardar en un archivo de acceso aleatorio la matriz de superficie.

[1] Profesor Asociado Tipo A. Tecnológico de Estudios Superiores de Cuautitlán Izcalli. profmiguelpiedras@hotmail.com
[2] Alumno. Tecnológico de Estudios Superiores de Cuautitlán Izcalli. gonzalezcastillobf@gmail.com

4. Crear un algoritmo que permite abrir el archivo de acceso aleatorio, recuperar los datos y reconstruir la figura.

Justificación

En la carrera de Ingeniería en Sistemas Computacionales se imparten materias de introducción a la programación, programación intermedia y avanzada, sin embargo hasta quinto semestre de la carrera, en ninguna se desarrolla software complejo, no porque no se pueda sino porque la estructura curricular no lo solicita, al llegar a este nivel, se pretende que el alumno comience con su especialización, por lo que el desarrollo de programas sofisticados, que exijan la creación de algoritmos es necesaria para garantizar la formación de los futuros profesionistas.

La creación del programa CAD, obedece a una conclusión obvia de la materia, al enseñar los algoritmos de dibujo que permiten crear figuras básicas, es evidente pensar en realizar un programa de dibujo, tal como AUTOCAD o cualquier programa de dibujo vectorial.
Se presentan los algoritmos usados en el proceso de creación del programa, así como uno de los resultados obtenidos en el curso, creado por uno de los alumnos de la materia.

METODOLOGÍA.

Se usaron dos metodologías para la creación, se separan en desarrollo del software, donde se establece el método utilizado, y el proceso de enseñanza aprendizaje.

Desarrollo del software

Se utilizo el modelo en espiral para la realización del software, Figura 1. El cual tiene las siguientes características.

Es un modelo de proceso de software evolutivo que acompaña la naturaleza evolutiva de con los aspectos controlados y sistemáticos del ciclo de vida tradicional. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. En este modelo, el sistema se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones se producen versiones cada vez más completas de ingeniería del sistema.

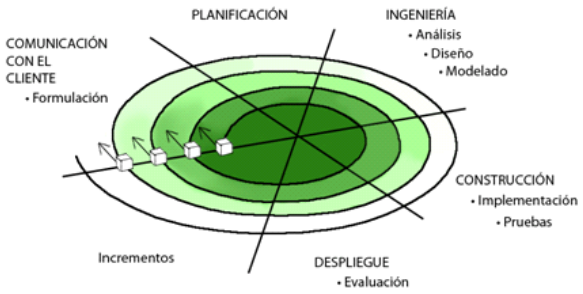

Figura 1. Proceso de desarrollo de software en espiral.

El Modelo en Espiral se divide en un número de actividades estructurales, también llamadas "regiones de tareas". Generalmente existen entre tres y seis regiones de tareas

Comunicación con el cliente. Las tareas requeridas para establecer comunicación entre el desarrollador y el cliente, sea revisar especificaciones, plantear necesidades, etc.

Planificación. Las tareas requeridas para definir recursos, tiempos e información relacionada con el proyecto.

Análisis de riesgos. Las tareas requeridas para evaluar riesgos técnicos y de gestión.

Ingeniería. Las tareas requeridas para construir una o más representaciones de la aplicación.

Construcción y adaptación. Las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario.

Evaluación del cliente. Las tareas requeridas para obtener la reacción del cliente, según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

Para este proceso se consideró al cliente como el profesor, quién es el que establece los requerimientos del sistema.

Proceso de enseñanza aprendizaje
Se usan dos teorías del aprendizaje, el primero es el socioconstructivista, y el segundo el aprendizaje basado en proyectos.

Paradigma educativo socioconstructivista
Uno de los fundamentos principales del paradigma educativo socioconstructivista da cuenta de que el proceso de enseñanza-aprendizaje se propicia en la interacción social en comunidades dialógicas y participativas, donde el contexto histórico-cultural del estudiante cumple un factor fundamental (Gómez y Rubio, 2017).

Aprendizaje basado en proyectos (ABP)
En el modelo de aprendizaje basado en proyectos se encuentra la esencia de la enseñanza problémica, mostrando al estudiante el camino para la obtención de los conceptos.

Las contradicciones que surgen y las vías para su solución, contribuyen a que este objeto de influencias pedagógicas se convierta en un sujeto activo.

Este modelo de aprendizaje exige que el profesor sea un creador, un guía, que estimule a los estudiantes a aprender, a descubrir y sentirse satisfecho por el saber acumulado, lo cual puede lograrse si aplica correctamente la enseñanza basada en proyectos.

El ABP aplicado en los cursos, proporciona una experiencia de aprendizaje que involucra al estudiante en un proyecto complejo y significativo, mediante el cual desarrolla integralmente sus capacidades, habilidades, actitudes y valores. Se acerca a una realidad concreta en un ambiente académico, por medio de la realización de un proyecto de trabajo. Estimula en los estudiantes el desarrollo de habilidades para resolver situaciones reales, con lo cual se motivan a aprender; los estudiantes se entusiasman con la investigación, la discusión y

proponen y comprueban sus hipótesis, poniendo en práctica sus habilidades en una situación real. En esta experiencia, el estudiante aplica el conocimiento adquirido en un producto dirigido a satisfacer una necesidad social, lo cual refuerza sus valores y su compromiso con el entorno, utilizando además recursos modernos e innovadores (Maldonado, 2008).

Bases teóricas

Para el desarrollo del programa se usó Delphi 10 y los algoritmos ya existentes de figuras y transformaciones, y se desarrollaron algoritmos para guardar los datos de la imagen, recuperarlos y reconstruirla.

Delphi es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual. En Delphi se utiliza como lenguaje de programación una versión moderna de Pascal, llamada Object Pascal.

Imágenes vectoriales

Una imagen vectorial es una imagen digital formada por objetos geométricos dependientes (segmentos, polígonos, arcos, muros, etc.), cada uno de ellos definido por atributos matemáticos de forma, de posición, etc. Por ejemplo, un círculo de color rojo quedaría definido por la posición de su centro, su radio, el grosor de línea y su color.

Este formato de imagen es completamente distinto al formato de las imágenes de mapa de bits, también llamados imágenes matriciales que están formados por píxeles. El interés principal de los gráficos vectoriales es poder ampliar el tamaño de una imagen a voluntad sin sufrir la pérdida de calidad que sufren los mapas de bits. De la misma forma, permiten mover, estirar y retorcer imágenes de manera relativamente sencilla. Su uso también está muy extendido en la generación de imágenes en tres dimensiones tanto dinámicas como estáticas.

Algoritmos

Se presenta una breve descripción de los algoritmos usados (Hearn & Baker, 2006).

Algoritmos para trazar líneas

DDA

El algoritmo de análisis diferencia/ digital (DDA) es un algoritmo de digitalización de líneas basado en calcular Sy o Sx. Las líneas se muestrean a intervalos unitarios según una de las coordenadas y los correspondientes valores enteros más próximos al trayecto lineal se calculan para la otra coordenada.

Línea de Bresenham

El Algoritmo de Bresenham es un método rápido para el trazado de líneas en dispositivos gráficos, cuya cualidad más apreciada es que solo realiza cálculos con enteros.
Se puede adaptar para rasterizar también circunferencias y curvas. Los ejes verticales muestran las posiciones de rastreo y los ejes horizontales identifican columnas de pixel.

Bézier

En los gráficos vectoriales, las curvas de Bézier se utilizan para modelar curvas suaves que se pueden escalar indefinidamente. Las "rutas", como se las conoce comúnmente en los programas de manipulación de imágenes, son combinaciones de curvas de Bézier

vinculadas. Las rutas no están limitadas por los límites de las imágenes rasterizadas y su modificación es intuitiva.

Las curvas de Bézier también se utilizan en el dominio del tiempo, particularmente en animación, diseño de la interfaz de usuario y suavización de la trayectoria del cursor en las interfaces controladas por la mirada. Por ejemplo, una curva de Bézier se puede usar para especificar la velocidad a lo largo del tiempo de un objeto, como un icono que se mueve de A a B, en lugar de simplemente moverse a un número fijo de píxeles por paso. Cuando los animadores o diseñadores de interfaces hablan sobre la "física" o la "sensación" de una operación, pueden estar refiriéndose a la curva Bézier particular utilizada para controlar la velocidad en el tiempo del movimiento en cuestión.

Un sistema de coordenadas cilíndricas es aquél en que cada punto P= (x, y, z) del espacio queda determinado por tres números P= (r, α, z)

Relación entre coordenadas cilíndricas y cartesianas.

1. Para cambiar de coordenadas cilíndricas a cartesianas, se usan las fórmulas:
   $x = r \cos \alpha \qquad y = r \operatorname{sen} \alpha \qquad z = z$
2. Para cambiar de coordenadas cartesianas a cilíndricas, se usan las fórmulas:
   $$r = \sqrt{x^2 + y^2} \qquad \operatorname{tg} \alpha = \frac{y}{x} \quad \Rightarrow \quad \alpha = \operatorname{arc\,tg} \frac{y}{x} \qquad z = z$$

Se utiliza este concepto para graficar, elipses, circunferencias y arcos.

Transformaciones

Traslación

Se aplica una traslación, en un objeto para cambiar su posición a lo largo de la trayectoria de una línea recta, de una dirección de coordenadas a otra. Convertimos un punto bidimensional al agregar las distancias de traslación, tx y ty la posición de coordenadas original (x, y).

El par de distancia de traslación se llama vector de traslación o vector de cambio. Se pueden expresar las ecuaciones anteriores en una sola ecuación matricial al utilizar vectores de columna para representar las posiciones de coordenadas y el vector de traslación.

Los polígonos se trasladan al sumar el vector de traslación a la posición de coordenadas de cada vértice y se vuelve a generar el polígono utilizando un nuevo conjunto de coordenadas y vértices y las especificaciones actuales de los atributos.

Rotación

Se aplica una rotación bidimensional en un objeto al cambiar su posición a lo largo de la trayectoria de una circunferencia en el plano de xy. Para generar una rotación, especificamos un ángulo de rotación θ y la posición (x r, y r) del punto de rotación (o punto pivote) en torno al cual se gira el objeto.

Escalación

Una transformación de escalación altera el tamaño de un objeto. Se puede realizar esta operación para polígonos al multiplicar los valores de coordenadas (x, y) de cada vértice por los factores de escalación s x y s y para producir las coordenadas transformadas (x', y').

RESULTADOS

En el desarrollo del software, se siguió el siguiente proceso:
Comunicación con el cliente. El profesor estableció las características del software.

Planificación. Una vez que el profesor establece las características, se dedica a enseñar los algoritmos básicos, y ejemplifica cada uno, directamente en el lenguaje de programación, con el objeto de que el alumno tenga, programas funcionales que después ira integrando en el programa final.

Análisis de riesgos. - Las tareas requeridas para evaluar riesgos técnicos y de gestión. Se establecen como un proceso de prueba y error, donde el estudiante debe identificar dónde se pueden disparar errores de código o funcionamiento, debiendo controlarlos.

Ingeniería. Conforme avanza el proceso de enseñanza por parte del profesor, se le asignan al estudiante actividades que debe ir resolviendo de forma continua.

Construcción y adaptación. Cada programa que se hace se prueba de forma individual y posteriormente se implementa en el sistema CAD para garantizar que no se generen conflictos.

Evaluación del cliente. Usando un proceso de evaluación continua, se garantiza que el profesor irá calificando cada programa en lo individual y su aplicación integral en el software solicitado.

El aprendizaje socioconstructivista se aplica de tres formas:

1. El estudiante captura los programas que el profesor muestra en clase, de forma individual y bajo la orientación del docente se asegura que funcione.
2. De forma individual se motiva al estudiante a integrarlo en el sistema complejo.
3. Se realizan sesiones de trabajo colaborativas, en donde cada estudiante muestra el proceso de avance y como lo logró. Motivando también a que se reúnan en equipos para solventar dudas y errores.

Lo anterior se desarrolla con la condición de que, si bien todos están desarrollando un software similar con las mismas bases y algoritmos, el resultado final debe tener características únicas de interfaz y procesos internos en tiempo de ejecución.

Producto obtenido

El resultado de reunir los algoritmos o primitivas básicas de los gráficos, fue lograr un programa de dibujo CAD que implementa tanto los algoritmos de dibujo como los de transformaciones.

La interfaz se muestra en la Figura 2, se logró una aplicación sencilla y amigable para el usuario.

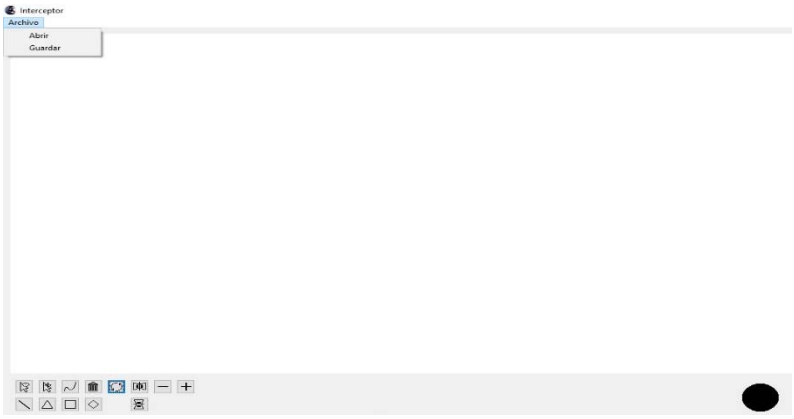

Figura 2. Interfaz del graficador CAD

En la Figura 3, se muestra un ejemplo de cómo funciona el programa, se pueden dibujar cualquier número de figuras, cambiando el color del relleno.

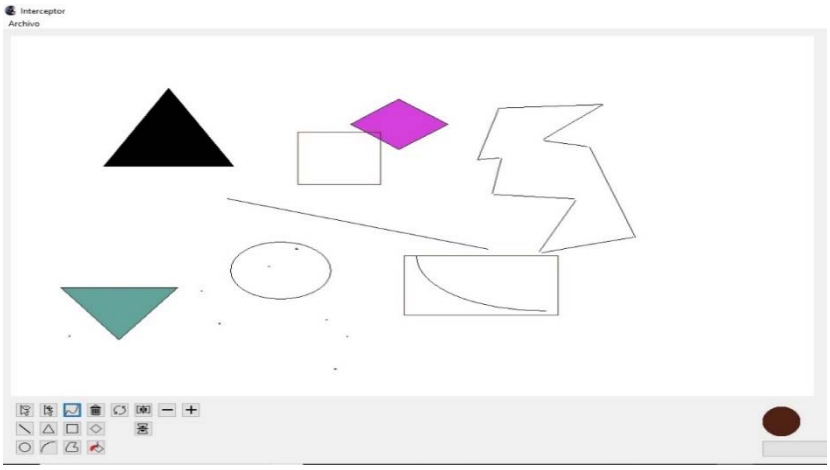

Figura 3. Ejemplo de la aplicación funcionando

Se dieron tres resultados importantes en el proceso, el primero, implementar todos los algoritmos en el mismo programa, logrando que estos funcionen de forma conjunta, sin que ninguna acción se traslape o deshaga la anterior. El segundo es el poder guardar el archivo de datos, donde solo se guardan números, el tercero que, al abrir el archivo, se recuperan los datos, y se reconstruye la figura.

Proceso de guardar

Para poder guardar los datos se usó un archivo de acceso aleatorio, construyendo un registro desde código, como se muestra en la Figura 3, con esto se crea un tipo de datos definido por el usuario.

```
type
  TAtributos=record
    N:Integer;
    R:Integer;
    G:Integer;
    B:Integer;
    MX:Integer;
    MY:Integer;
    NX:Integer;
    NY:Integer;
    PI:Integer;
    C:Integer;
    PF:Integer;
end;
```

Figura 4. Registro para almacenar datos.

Una vez que se tiene el registro creado, se procede a guardar el documento, el algoritmo se muestra en la Figura 4, se comienza por crear el archivo, y posteriormente, se vacía en el archivo los datos usando un ciclo for, para poder guardar todos los atributos de la figura se usan dos archivos más, con terminación a y c. En ellos se almacena información tal como, ancho de contorno, color de relleno y atributos de las figuras como ángulos, radios, etcétera.

Para poder identificar a cada figura se almacena el número de objeto, es decir, conforme se dibuja cada elemento obtiene un identificador único durante el proceso de dibujo, de tal suerte que se puede encontrar en la lista. Es de hacer notar que este proceso, sustituye el uso de una base de datos para almacenar, con lo que se evita la dependencia con un programa de terceros, además que se les da a los archivos creados, una extensión única, con lo que se evita que otros programas de dibujo, puedan abrirlo.

```
procedure TForm8.GuardarlClick(Sender: TObject);
var
GFiguras: TFigura;
GF1: File of TFigura;

GAtributos: TAtributos;
GF2: File of TAtributos;

GColores: TColores;
GF3: File of TColores;
  I: Integer;
begin
Save.Execute;

AssignFile( GF1 , Save.FileName + '.igc');
Rewrite(GF1);
Reset(GF1);
for I := 0 to NPuntos-1 do
  begin
  Seek(GF1,I);
  GFiguras.N:=Figura[I,0];
  GFiguras.Px:=Figura[I,1];
  GFiguras.Py:=Figura[I,2];
  write(GF1,GFiguras);
  end;
CloseFile(GF1);

AssignFile( GF2 ,  Save.FileName + 'a.igc');
Rewrite(GF2);
Reset(GF2);
for I := 0 to NFigura-1 do
  begin
  Seek(GF2,I);
  GAtributos.N:=Atributos[I,0];
  GAtributos.R:=Atributos[I,1];
  GAtributos.G:=Atributos[I,2];
  GAtributos.B:=Atributos[I,3];
  GAtributos.MX:=Atributos[I,4];
  GAtributos.MY:=Atributos[I,5];
  GAtributos.NX:=Atributos[I,6];
  GAtributos.NY:=Atributos[I,7];
  GAtributos.PI:=Atributos[I,8];
  GAtributos.C:=Atributos[I,9];
  GAtributos.PF:=Atributos[I,10];
  write(GF2,GAtributos);
  end;
CloseFile(GF2);

AssignFile( GF3 , Save.FileName + 'c.igc');
Rewrite(GF3);
Reset(GF3);
for I := 0 to NColor-1 do
  begin
  Seek(GF3,I);
  GColores.Px:=Colores[I,0];
  GColores.Py:=Colores[I,1];
  GColores.R:=Colores[I,2];
  GColores.G:=Colores[I,3];
  GColores.B:=Colores[I,4];
  write(GF3,GColores);
  end;
CloseFile(GF3);
Inicio();
Redibujar(Guardado);
end;
```

Figura 5. Algoritmo para guardar el dibujo.

Proceso de abrir

En la Figura 5, se muestra el algoritmo usado para cargar los datos del archivo guardado, al programa, para ello se tienen tres arreglos bidimensionales, en el primero se guardan los datos principales de cada figura, en el segundo, los atributos como tamaño, grueso de contorno, etcétera, y en el tercero los atributos de colores de cada figura.

```
procedure TForm8.Abr1Click(Sender: TObject);
var
GFiguras: TFigura;
GF1: File of TFigura;

GAtributos: TAtributos;
GF2: File of TAtributos;

GColores: TColores;
GF3: File of TColores;
  I: Integer;
begin
Open.Execute();

AssignFile( GF1 , Open.FileName + '.igc');
Reset(GF1);
NPuntos:= FileSize(GF1);
for I := 0 to FileSize(GF1)-1 do
  begin
  Seek(GF1,I);
  BlockRead(GF1,GFiguras,1);
  Figura[I,0]:=GFiguras.N;
  Figura[I,1]:=GFiguras.Px;
  Figura[I,2]:=GFiguras.Py;
  end;
CloseFile(GF1);


AssignFile( GF2 ,  Open.FileName + 'a.igc');
Reset(GF2);
NFigura:=FileSize(GF2);
for I := 0 to FileSize(GF2)-1 do
  begin
  Seek(GF2,I);
  BlockRead(GF2,GAtributos,1);
  Atributos[I,0]:=GAtributos.N;
  Atributos[I,1]:=GAtributos.R;
  Atributos[I,2]:=GAtributos.G;
  Atributos[I,3]:=GAtributos.B;
  Atributos[I,4]:=GAtributos.MX;
  Atributos[I,5]:=GAtributos.MY;
  Atributos[I,6]:=GAtributos.NX;
  Atributos[I,7]:=GAtributos.NY;
  Atributos[I,8]:=GAtributos.PI;
  Atributos[I,9]:=GAtributos.C;
  Atributos[I,10]:=GAtributos.PF;
  end;
CloseFile(GF2);


AssignFile( GF3 , Open.FileName + 'c.igc');
Reset(GF3);
NColor:=FileSize(GF3);
for I := 0 to FileSize(GF3)-1 do
  begin
  Seek(GF3,I);
  BlockRead(GF3,GColores,1);
  Colores[I,0]:=GColores.Px;
  Colores[I,1]:=GColores.Py;
  Colores[I,2]:=GColores.R;
  Colores[I,3]:=GColores.G;
  Colores[I,4]:=GColores.B;
  end;
CloseFile(GF3);
Redibujar(Guardado);
end;
```

Figura 6. Proceso de abrir el documento.

Una vez cargados los datos se procede a redibujar la figura, este proceso se muestra en la Figura 6. Para poder lograrlo se toma al identificador de cada figura, se obtiene el tipo de figura que es, es decir, línea, circulo, poli línea, y se llama al algoritmo correspondiente, aplicando los atributos y colores guardados, con ello se garantiza que el dibujo se reconstruye en el mismo orden en que fue creado originalmente. Una vez terminado este proceso el usuario puede modificar el archivo todo lo que desee, y posteriormente guardarlo, como un archivo nuevo o modificar el existente.

```
//Redibujar figuras
procedure Redibujar(Redibujar:TImage);
var
I,J,K,L:integer;
begin
Redibujar.Canvas.Brush.Color:=rgb(255,255,255);
Redibujar.Canvas.FloodFill( 0,0, Redibujar.picture.bitmap.canvas.pixels[0,0] , fsSurface );
Redibujar.Canvas.Fillrect(Redibujar.Canvas.ClipRect);
for I := 0 to NFigura-1 do
  begin
  if Atributos[I,0]>-1 then
    begin
    J:=Atributos[I,8];
    for K := 0 to Atributos[I,9] do
      begin
      for L := 0 to 3 do
        begin
          Puntos[L].X:=Figura[J,1];
          Puntos[L].Y:=Figura[J,2];
          J:=J+1;
        end;
      J:=J-1;
      Redibujar.Canvas.Pen.Color:=rgb(Atributos[I,1],Atributos[I,2],Atributos[I,3]);
      Redibujar.Canvas.PolyBezier(Puntos);
      end;
    end;
  end;
for L := 0 to NColor-1 do
  begin
    Redibujar.Canvas.Brush.Color:=rgb(Colores[L,2],Colores[L,3],Colores[L,4]);
    Redibujar.Canvas.FloodFill( Colores[L,0],Colores[L,1], Redibujar.picture.bitmap.
    canvas.pixels[Colores[L,0],Colores[L,1]] , fsSurface );
  end;
```

Figura 7. Reconstrucción de la figura.

CONCLUSIONES

Con los resultados mostrados de puede asegurar que el objetivo se cumplió completamente. Las metodologías usadas tanto en el proceso de desarrollo del software como en el proceso de enseñanza-aprendizaje, dieron como resultado, programas CAD funcionales que, si bien tienen errores en tiempo de ejecución, son solventadas, siguiendo los pasos que cada estudiante ideo para su sistema particular.

El objetivo fue, “Desarrollar un programa de dibujo CAD, usando algoritmos de figuras elementales como línea, circulo, elipse, Bézier, y aplicando transformaciones en 2D, tales como, traslación, escalación, rotación, corte y reflexión, a cada una. El programa guarda los datos vectoriales de la imagen y lo puede reconstruir”, el programa crea cada figura solicitada, lo guarda, lo abre y reconstruye la imagen, además que cada algoritmo de transformación, se le aplica indistintamente a cada figura.

BIBLIOGRAFIA

Hearn, D. & Baker, P. (2006). Gráficos por computadora con Open GL. España: Pearson Educación, S. A.

Gómez, F. & Rubio, J. (2017). Cognición contextualizada: una propuesta didáctica y psicopedagógica socioconstructivista para la enseñanza-aprendizaje del derecho. *Revista Pedagogía Universitaria y Didáctica del Derecho. Vol.* (4). Recuperado el 01 de febrero de 2019, de: https://revistas.uchile.cl/index.php/RPUD/article/view/47970

Maldonado, P. (2008). Aprendizaje Basado en Proyectos Colaborativos. Una experiencia en educación superior. *Laurus Revista de educación*. *vol.* (14). Recuperado el 01 de febrero de 2019, de https://www.redalyc.org/pdf/761/76111716009.pdf