24 de enero de 2022.
Dictamen C.I. 02/2022

DICTAMEN
QUE PRESENTA LA COMISIÓN DE INVESTIGACIÓN DE LA DIVISIÓN DE CIENCIAS DE LA COMUNICACIÓN Y DISEÑO

ANTECEDENTES

I. El Consejo Divisional de Ciencias de la Comunicación y Diseño, en la sesión 03.21, celebrada el 29 de enero de 2021, integró esta Comisión en los términos señalados en el artículo 55 de Reglamento Interno de los Órganos Colegiados Académicos.

II. El Consejo Divisional designó para esta Comisión a los siguientes integrantes:

a) Órganos personales:
- ✓ Dra. Margarita Espinosa Meneses, Jefa del Departamento de Ciencias de la Comunicación;
- ✓ Dra. Erika Cecilia Castañeda Arredondo, Jefa del Departamento de Teoría y Procesos del Diseño;
- ✓ Dr. Carlos Joel Rivero Moreno, Jefe del Departamento de Tecnologías de la Información.

b) Representantes propietarios:
- Personal académico:
- ✓ Departamento de Ciencias de la Comunicación;
- ✓ Dra. Lucero Fabiola García Franco, Departamento de Teoría y Procesos del Diseño.
- ✓ Dr. Alfredo Piero Mateos Papis, Departamento de Tecnologías de la Información.

CONSIDERACIONES

I. La Comisión recibió, para análisis y discusión, el reporte parcial de resultados del proyecto de investigación denominado "***Geometría en Movimiento 3***" presentado por la Dra. Dina Rochman Beer, aprobado en la Sesión 13.20 celebrada el 24 de noviembre de 2020, mediante el Acuerdo DCCD.CD.04.13.20.

II. La Comisión de Investigación sesionó el 24 de enero de 2022, fecha en la que concluyó su trabajo de análisis y evaluación del reporte parcial de resultados, con el presente Dictamen.

III. La Comisión tomó en consideración los siguientes elementos:

- *"Lineamientos para la creación de grupos de investigación y la presentación, seguimiento y evaluación de proyectos de investigación"* aprobados en la Sesión 06.16 del Consejo Divisional de Ciencias de la Comunicación y Diseño, celebrada el 6 de junio de 2016, mediante al acuerdo DCCD.CD.15.06.16.
- Protocolo de investigación.
- Relevancia para el Departamento.
- Objetivos planteados.
- Resultados obtenidos.

IV. **Objetivo general**:

Realizar un pantógrafo, al que se le llamará "Pantógrafo XYZ" con el cual se pueda reproducir ls curvas de las partes del cuerpo de los animales de pequeña escala para realizar su impresión 3D a mayor escala.

V. **Actividades realizadas durante el primer año:**

- Boceto del modelo,
- Simulación bidimensional del mecanismo,
- Primer modelo,
- Impresión 3D del primer modelo,
- Abstract para el congreso CAD ´22 que se realizará en China en julio 11 al 13 de 2022.

DICTAMEN

ÚNICO:

Tras evaluar el reporte parcial de resultados del proyecto de investigación denominado ***"Geometría en Movimiento 3"*** presentado por la Dra. Dina Rochman Beer, la Comisión de Investigación recomienda al Consejo Divisional de Ciencias de la Comunicación y Diseño aceptarlo.

VOTOS:

Integrantes	Sentido de los votos
Dra. Margarita Espinosa Meneses	A favor
Dra. Erika Cecilia Castañeda Arredondo	A favor
Dr. Carlos Joel Rivero Moreno	A favor
Dra. Lucero Fabiola García Franco	A favor
Dr. Alfredo Piero Mateos Papis	A favor
Total de los votos	**5 votos a favor**

Coordinadora

Mtra. Silvia Gabriela García Martínez
Secretaria del Consejo Divisional de Ciencias de la Comunicación y Diseño

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

Ciudad de México 13 de enero 2022

DTPD.023.22

Asunto:

Reporte de avance anual del proyecto: "Geometría en Movimiento 3"

Dra. Gloria Angélica Martínez de la Peña
Presidente del Consejo Divisional
División de Ciencias de la Comunicación y Diseño
Universidad Autónoma Metropolitana
Unidad Cuajimalpa
Presente

Por este medio hago de su conocimiento el reporte de avance anual del proyecto de investigación "Geometría en Movimiento 3", cuyo responsable es la Dra. Dina Rochman Beer, para su dictamen y aprobación.

El proyecto de investigación "Geometría en Movimiento 3" fue aprobado por el Consejo Divisional de la DCCD en la Sesión 13.20 del Consejo Divisional, mediante el acuerdo DCCD.CD.04.13.20 del 24 de noviembre de 2020, por un período de tres años, del 25-nov-20 al 24-nov-23.

Para su análisis y dictaminación, **se anexan los siguientes documentos**:

Reporte de avance anual de la Investigación.

Probatorios de los productos de investigación.

De igual forma, a continuación, enuncio los documentos que se anexan a la presente con la intención de dar un contexto del proyecto:

- Dictamen de la Comisión de Investigación.
    - Protocolo de Investigación

División de Ciencias de la Comunicación y Diseño
Unidad Cuajimalpa
DCCD | División de Ciencias de la Comunicación y Diseño
Jefatura del Departamento de Teoría y Procesos del Diseño
Torre III, 5to. piso. Av. Vasco de Quiroga 4871,
Colonia Santa Fe Cuajimalpa. Alcaldía Cuajimalpa de Morelos.
C.P. 05348, Ciudad de México.
Tel.: (+52) 55.5814.5348
http://dccd.cua.uam.mx

- Aprobación en el Consejo Divisional de CCD.

Sin más por el momento, quedo a sus órdenes para cualquier duda o aclaración y le envío un cordial saludo.

Atentamente

Casa abierta al tiempo

Dra. Erika Cecilia Castañeda Arredondo

Jefa del Departamento de Teoría y procesos del Diseño

*ccp. Archivo

Primer reporte del Proyecto de investigación “Geometría en Movimiento 3”

Los insectos y moluscos son parte de la cultura y del ecosistema de todo el mundo. Los biólogos para estudiarlos utilizan el método de la morfometría geométrica o el método de marca-ruptura para analizar su crecimiento y los cambios morfológicos que han sufrido tanto los insectos como los moluscos debido al cambio climático.

El uso de estos métodos implica mucho tiempo y esfuerzo para los biólogos porque el trabajo se realiza inicialmente de forma manual para marcar los puntos de referencia (Landmarks). Luego utilizan programas de computadora para su análisis, descripción y evaluación.

Mi interés es el estudio de la forma morfológica de los insectos y los moluscos desde la geometría descriptiva para reproducir físicamente su estructura en impresión 3D y estudiarla.

Cuando realicé mi estancia sabática en el Instituto de Ecología (INECOL) de Jalapa, Veracruz en el año 2015, inventé un nuevo método y realicé tres programas de cómputo para encontrar los puntos de referencia, es decir las coordenadas “x”, “y” y “z” de las formas biológicas de los insectos de pequeña escala para modelar su estructura geométrica con el programa de cómputo AutoCAD™.

En 2021, diseñé y construí un modelo tridimensional (en impresión 3D) para documentar el crecimiento periódico de los exoesqueletos. Dado que se tienen los valores numéricos de los puntos de referencia, es decir de las coordenadas "x" e "y" y el contorno de las curvas del exoesqueleto, se pueden realizar comparaciones matemáticas.

Tanto el método y el modelo de mi invención funcionan bien, pero tienen errores. Esto se debe a que las fotografías tienen perspectiva, por lo que es complicado encontrar la proyección de cada uno de los puntos en el espacio en cada una de las vistas de la proyección ortogonal. Decidí por lo tanto mejorar el método.

La primera prueba que realicé fue el de pegar la concha de un mejillón sobre una base y cocerle una malla de hilos en los dos sentidos, transversal y longitudinal. Aunque las fotografías tienen perspectiva la malla sirvió para encontrar más rápido las proyecciones de los puntos en el espacio en las fotografías y tener menos errores, por lo que el trabajo fue más eficiente.

El análisis de este trabajo me llevó a pensar que se podrían trazar las curvas de las partes de los animales de pequeña escala de otra manera sin utilizar los valores numéricos que se encuentran a partir de la morfometría geométrica siguiendo los fundamentos de la geometría descriptiva de las proyecciones de los puntos en el espacio en la proyección ortogonal.

Así fue como llegué a la idea de un pantógrafo para reproducir bidimensionalmente sobre una hoja de papel tanto los puntos de referencia o las coordenadas “x”, “y” como cada una

de las curvas de las partes de los animales de pequeña escala basándome en el invento de Christoph Scheiner.

El objetivo del proyecto de investigación “Geometría en Movimiento3” es el de realizar un pantógrafo, al que llamaré “Pantógrafo XYZ”, con el cual se pueda reproducir las curvas de de las partes del cuerpo de los animales de pequeña escala para realizar su impresión 3D a mayor escala.

El impacto del proyecto de investigación “Geometría en Movimiento 3” será muy importante debido al cambio de paradigma para encontrar los valores numéricos de los puntos de referencia, es decir las coordenadas x , y, y las curvas de las partes de los insectos y moluscos para establecer los cambios morfológicos que han sufrido tanto los insectos como los moluscos debido al cambio climático.

A partir de los valores numéricos los biólogos podrán describir, comprender y predecir los cambios que ocurren en la naturaleza. Y desde la geometría descriptiva se podrá modelar la estructura de los insectos y moluscos y crear modelos físicos tridimensionales para su estudio.

El Pantógrafo XYZ, se difundirá en los centros educativos, centros de investigación y en congresos para dar a conocer y universalizar el resultado del prototipo.

Acorde al cronograma del protocolo de investigación, a continuación, expondré el trabajo que realicé en el primer año.

Índice

1. Boceto del modelo
2. Simulación bidimensional del mecanismo
3. Primer modelo
4. Impresión 3D del primer modelo
5. Abstract para el congreso CAD´22 que se realzará en China, julio 11-13 del 2022.

BOCETO DEL MODELO

Para el desarrollo del pantógrafo tridimensional, me basé en la invención de Christoph Scheiner. Christoph Scheiner, matemático alemán, que nació el 25 de julio de 1573 en la localidad de Wald, cerca de Mindelheim, en la región de Suabia, en el suroeste de Alemania, inventó en 1603 el pantógrafo, que permite ampliar la copia de imágenes. Geométricamente, el pantógrafo de Scheiner se basa en la propiedad de los paralelogramos. Está compuesto por cuatro varillas conectadas de tal manera que se pueden mover con respecto a un punto o un pivote (Figura 1).

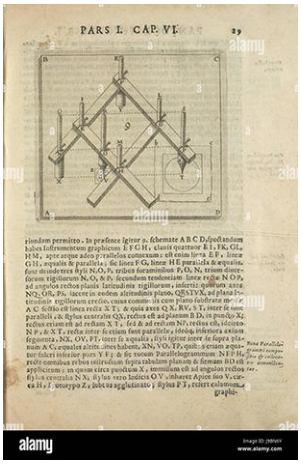
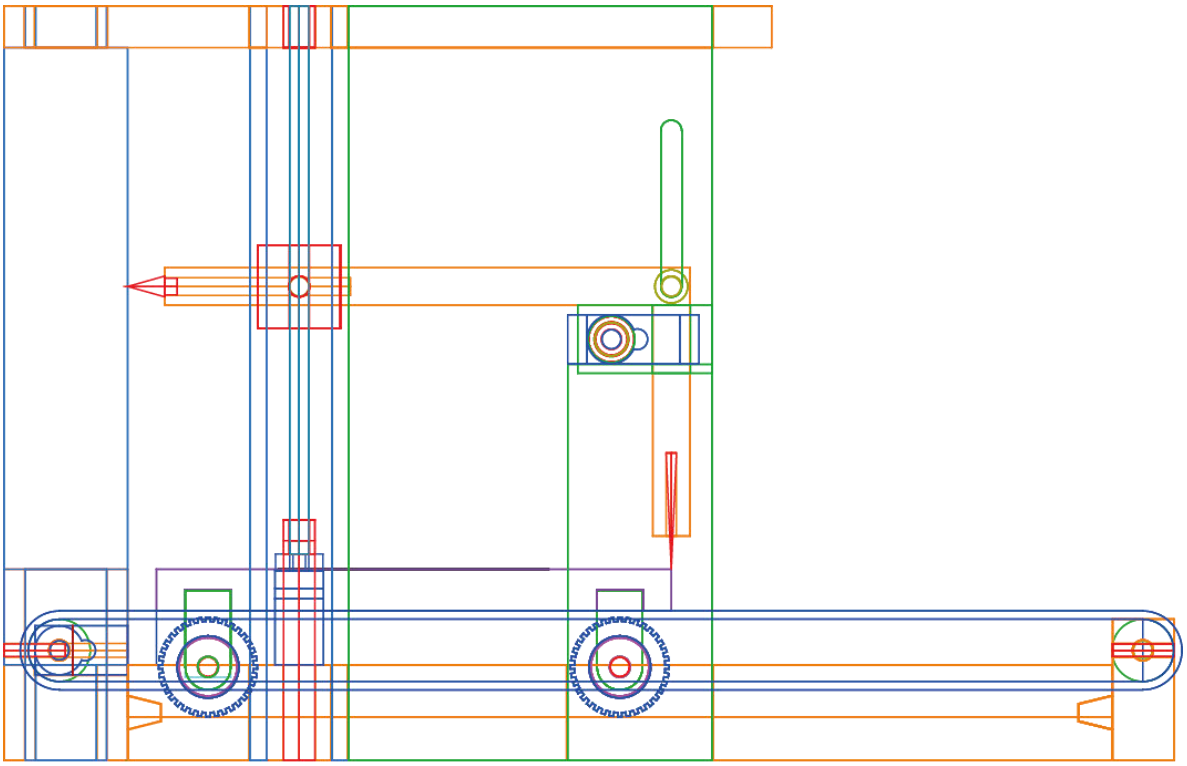


Figura 1. Pantógrafo de Christoph Scheiner.

Para dibujar, el pivote es fijo y el puntero de referencia se mueve sobre el dibujo original; un bolígrafo en el punto de copia reproduce la imagen a una escala mayor, que viene determinada por la relación de distancias P-RP y P-CP. P es el pivote; RP es el punto de referencia y CP es el punto de copia. Cambiar el puntero de referencia al punto de copia reproduce la imagen a menor escala.

Geométricamente, nuestro pantógrafo tridimensional se basa en el método de representación de Gaspard Monge. Con las siguientes partes: (1) una mesa con ruedas que se mueve hacia adelante y hacia atrás en el eje z a través de una banda; (2) una base rectangular sobre un riel que se mueve hacia la derecha y hacia la izquierda en el eje x a través de una varilla roscada Acme. Sobre esta base se encuentran el puntero y la punta del lápiz. Y (3) dos varillas Acme roscadas para mover el riel hacia arriba y hacia abajo en el eje z (Figura 2).



Proyecto “Geometría en Movimiento 3”
Pantógrafo XYZ
vista frontal
Dra. Dina Rochman Beer
2021

Figura 2.

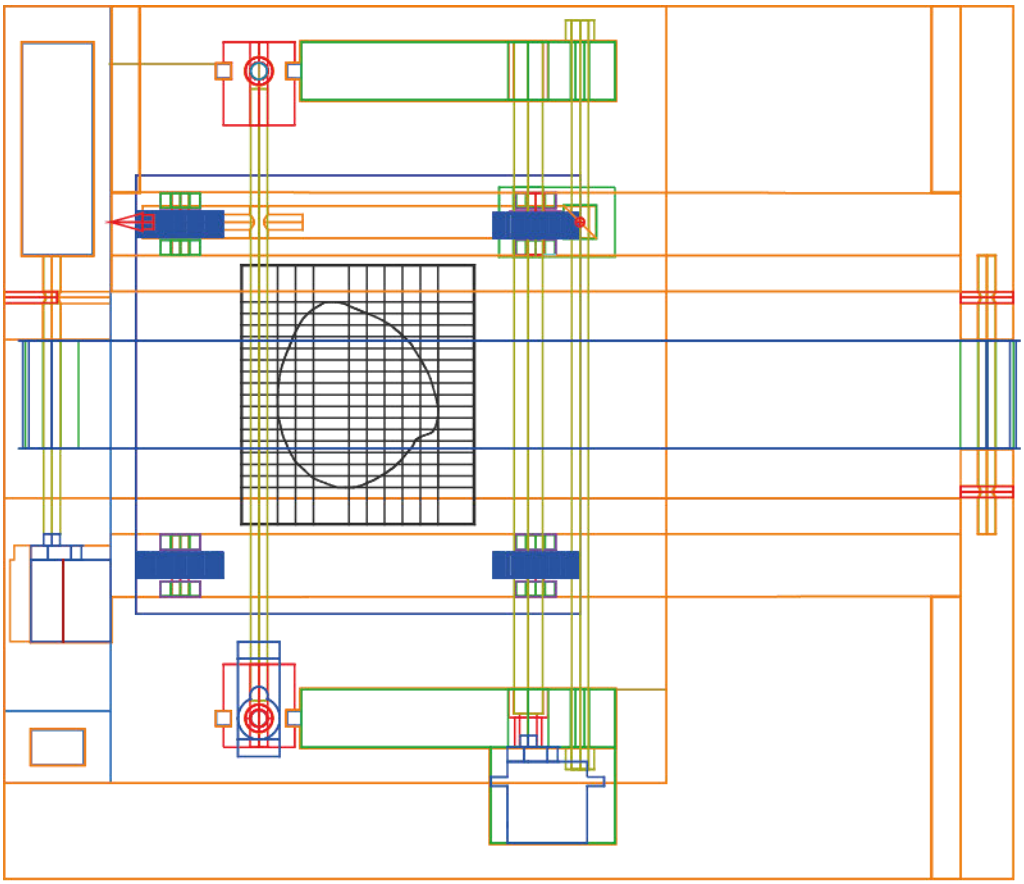
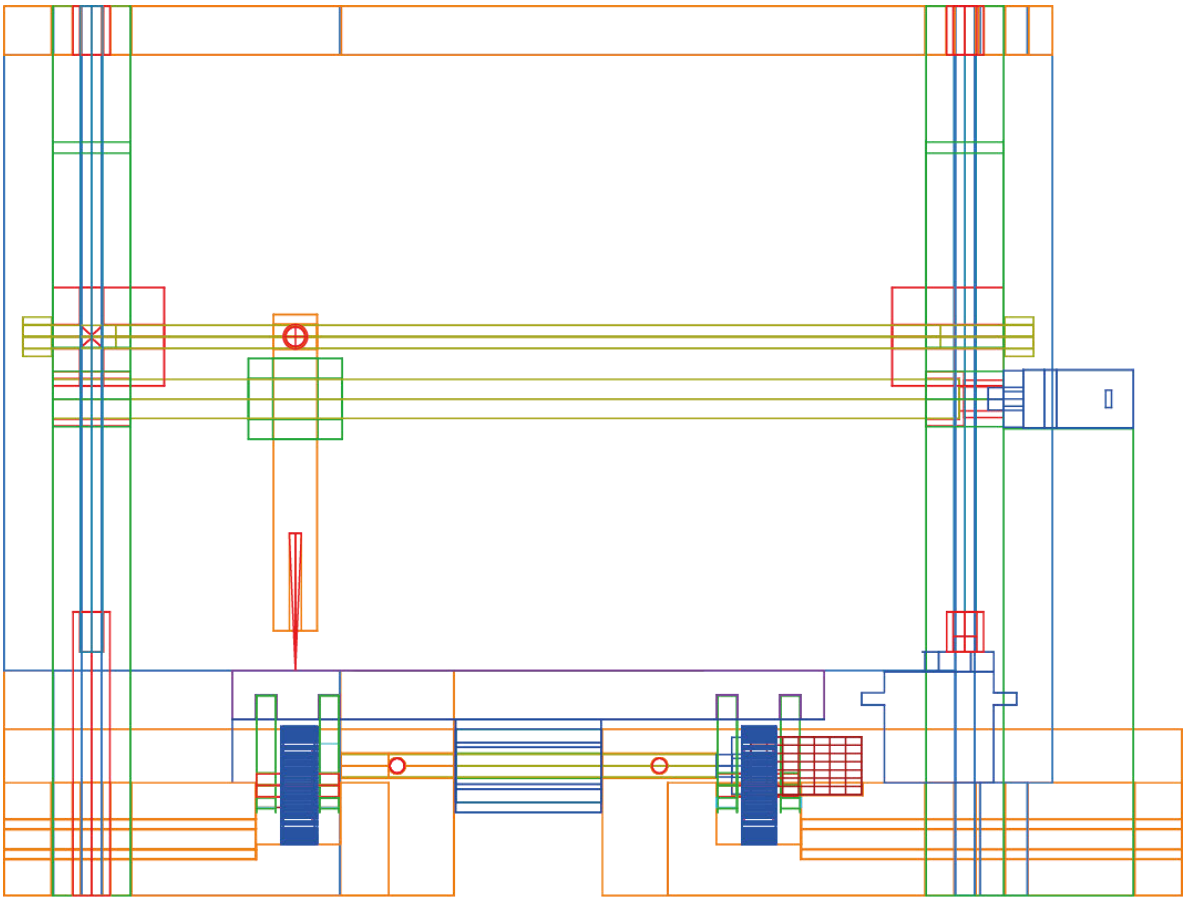
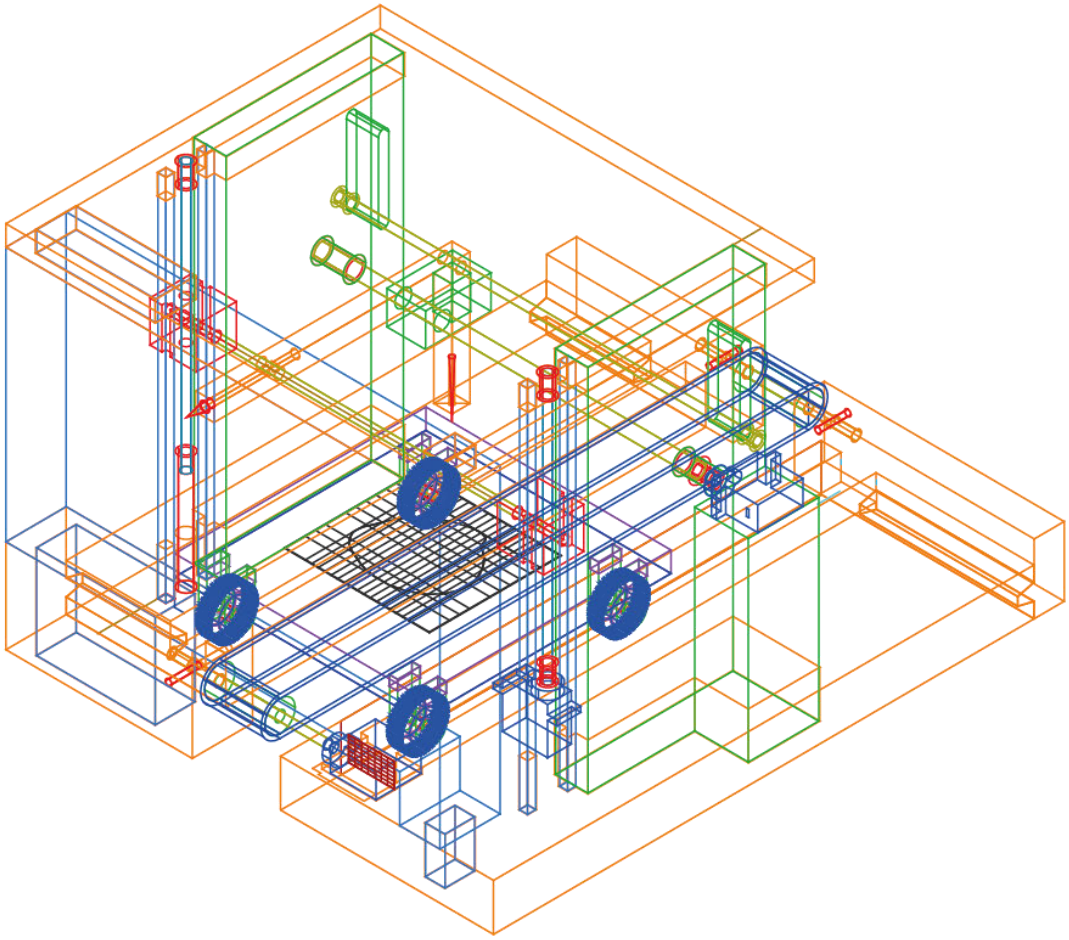

Proyecto “Geometría en Movimiento 3”
Pantógrafo XYZ
vista superior
Dra. Dina Rochman Beer
2021

Figura 3.



Proyecto “Geometría en Movimiento 3”
Pantógrafo XYZ
vista lateral
Dra. Dina Rochman Beer
2021

Figura 4.



Proyecto “Geometría en Movimiento 3”
Pantógrafo XYZ
vista isométrico
Dra. Dina Rochman Beer
2021

Figura 5.

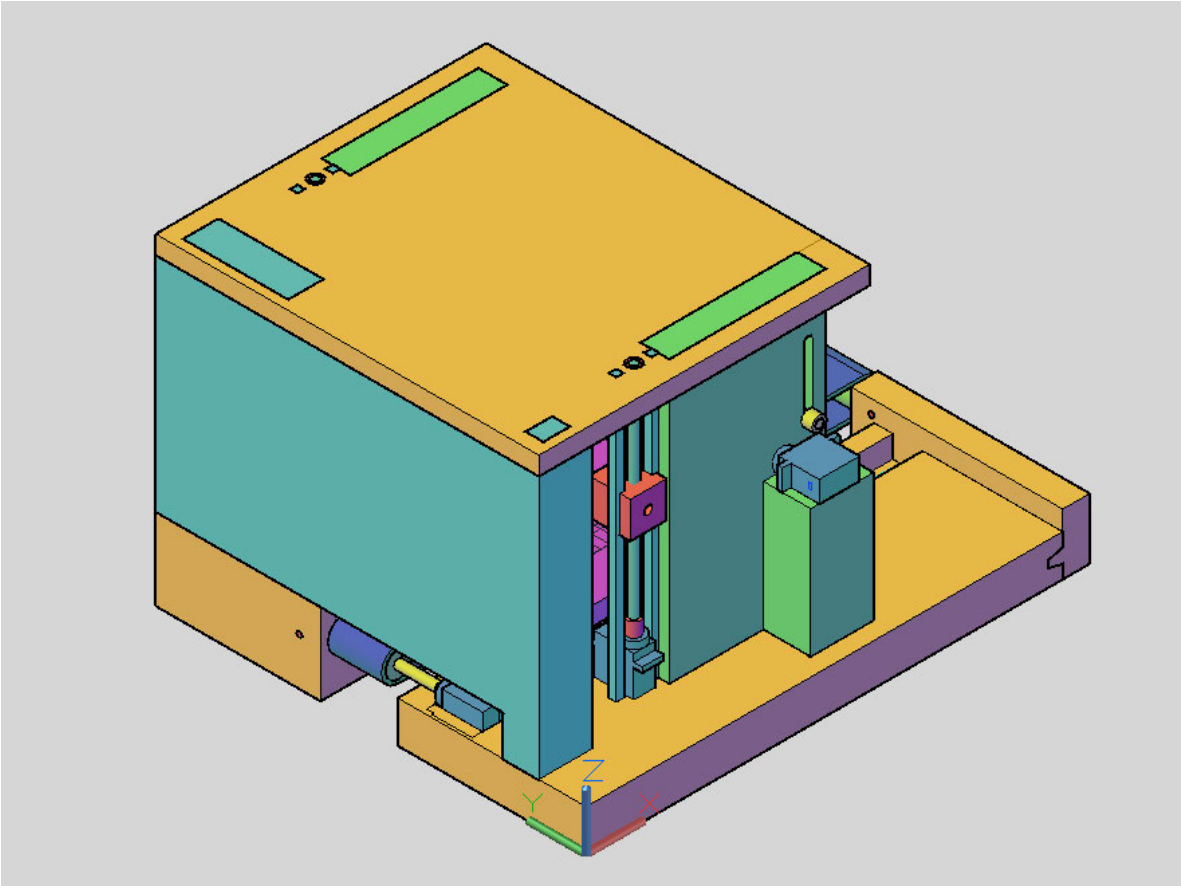


Figura 6. ISOMÉTRICO SUROESTE

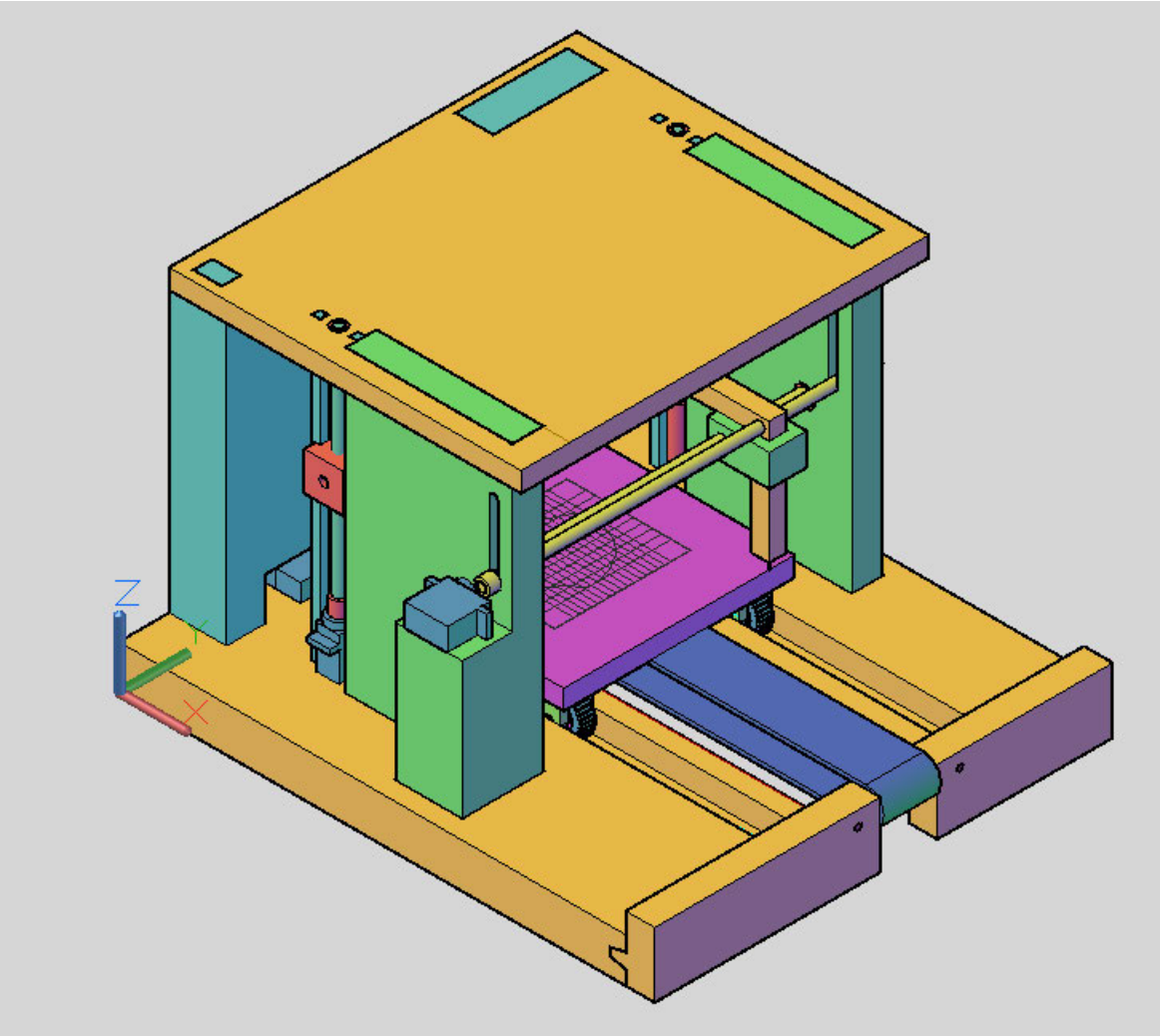


Figura 7. VISTA ISOMÉTRICO SURESTE

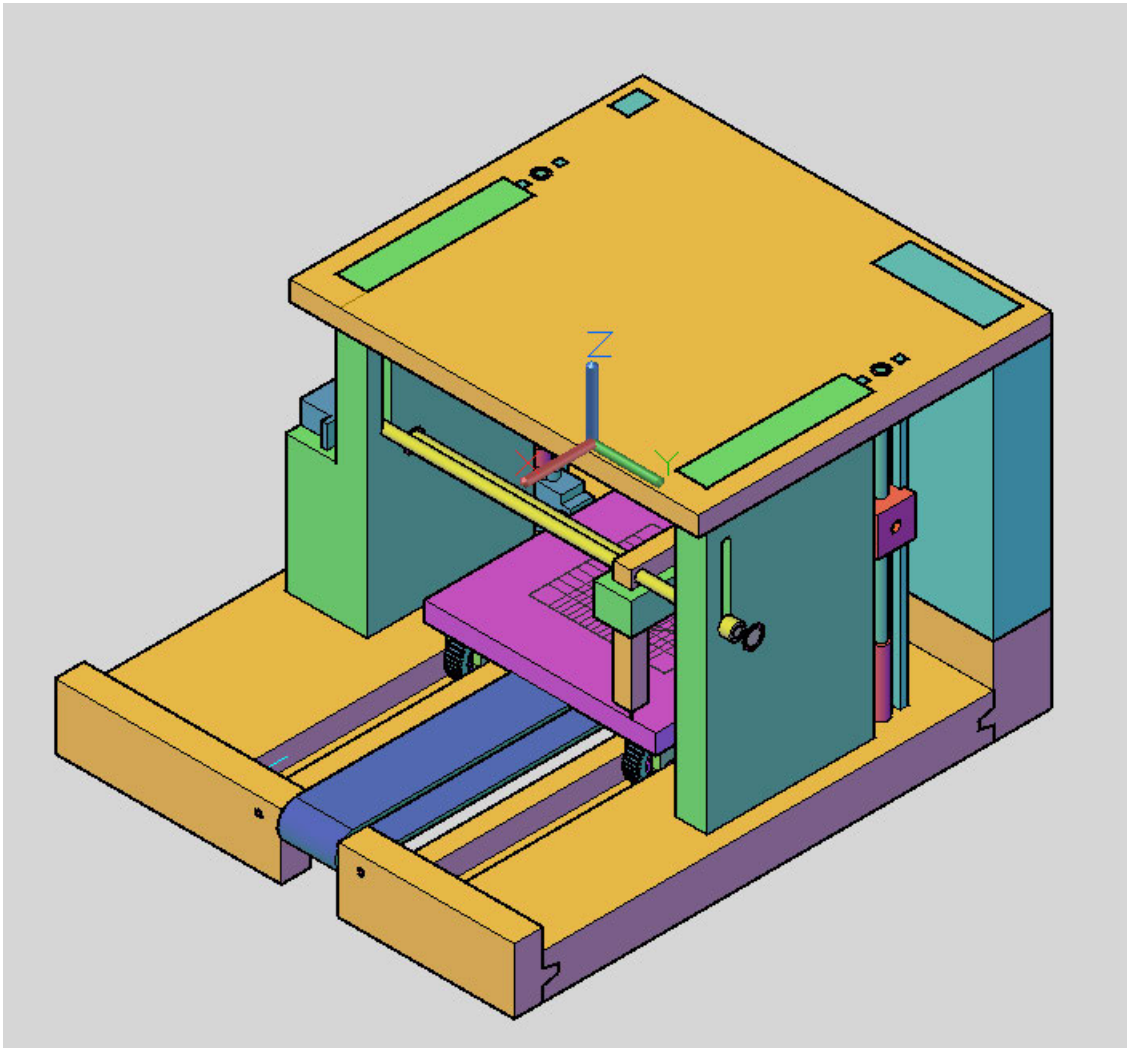


Figura 8. VISTA ISOMÉTRICO NORESTE

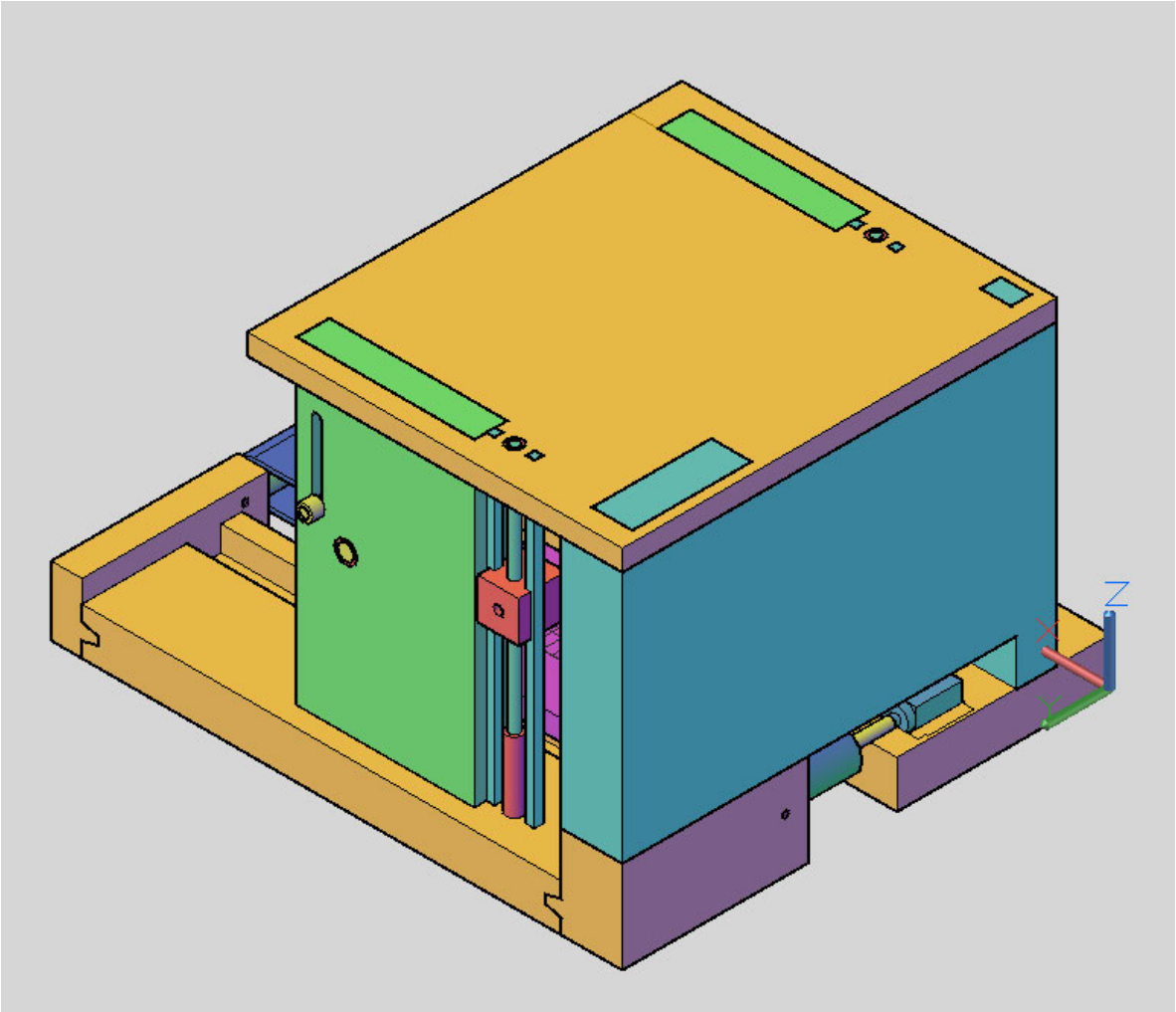
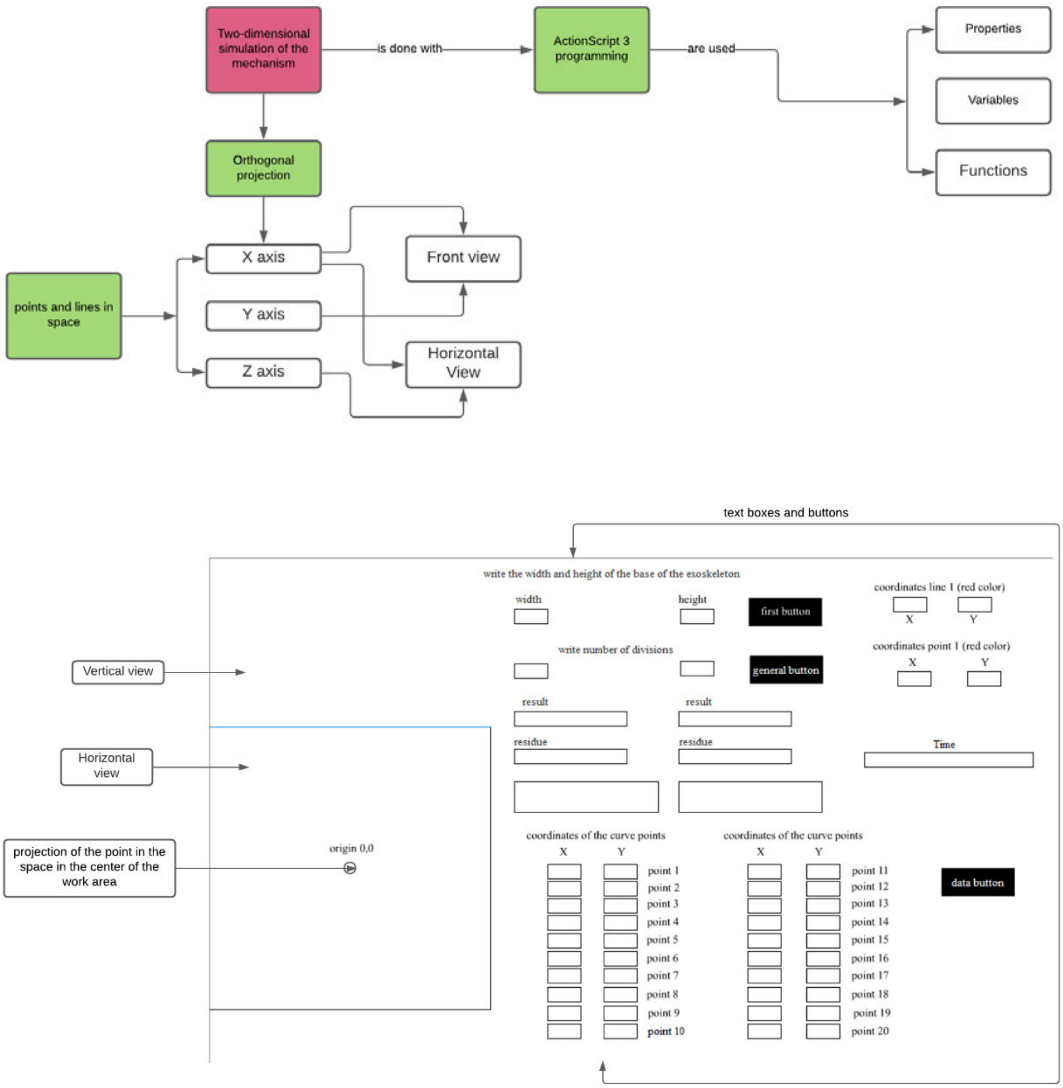


Figura 9. VISTA ISOMÉTRICO NOROESTE

SIMULACIÓN BIDIMENSIONAL DEL MECANISMO DEL BOCETO

Para comprobar el funcionamiento del Pantógrafo XYZ, realicé una simulación bidimensional en el programa Adobe Animate 2022™. En la figura que a continuación se presenta se puede observar la metodología, el algoritmo y el diseño gráfico de la aplicación que utilicé para realizar la simulación.



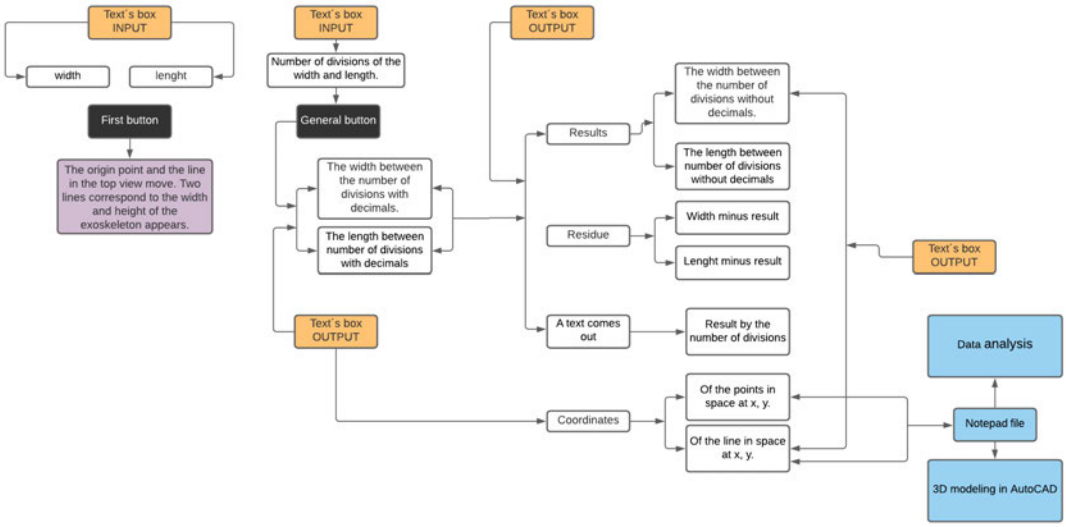


Figura 10. Metodología.

DISEÑO GRÁFICO DE LA PANTALLA

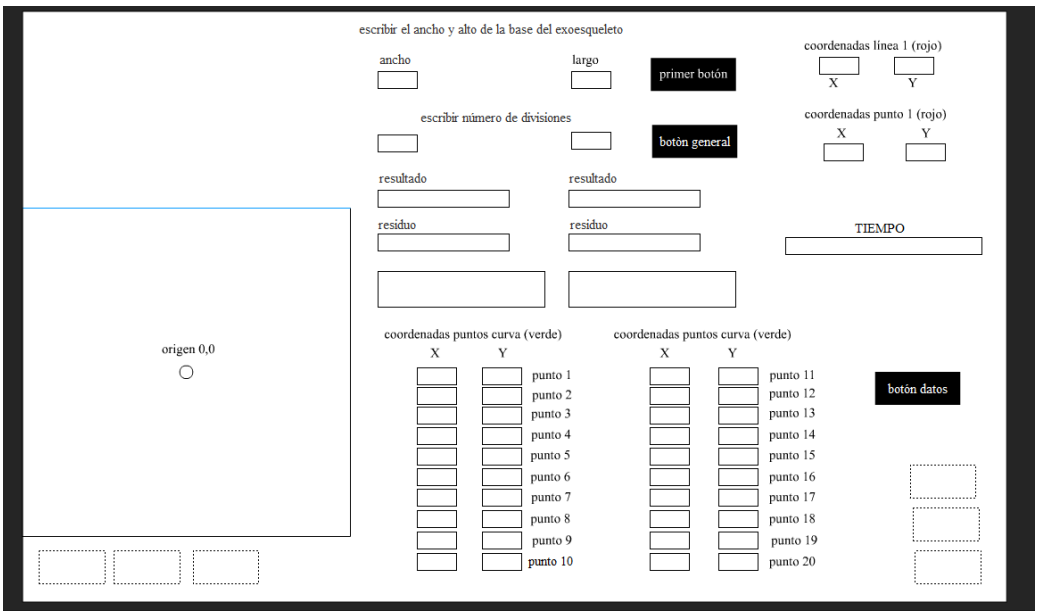


Figura 11. Diseño gráfico de la pantalla.

Cabe mencionar que, en la simulación que se presenta, el contorno geométrico de la curva del exoesqueleto es el encontrado en trabajos anteriores. En la programación, las distancias

de ancho, largo y alto de las curvas del exoesqueleto están en unidades. Los nombres de las variables están escritos en español.

Para poder realizar la programación, primero fue necesario hacer un diagrama de la curva que va del punto 0 al punto 300 para encontrar las distancias de la altura de la curva cada 25 unidades (Figura 12).

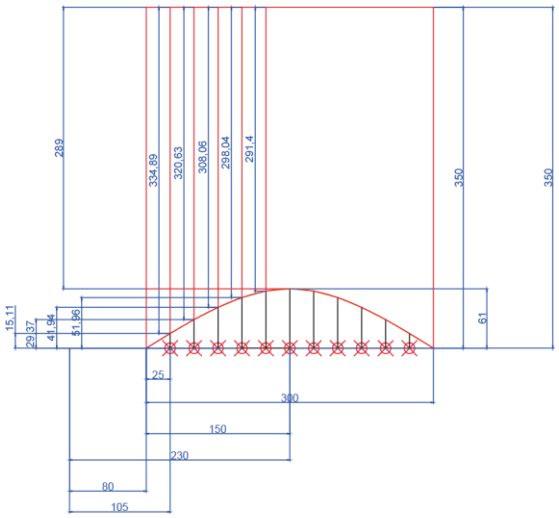


Figura 12. Distancias de la altura de la curva cada 25 unidades.

Se encontraron el porcentaje de las alturas con respecto a la altura del centro de la curva (Tabla 1).

Points	Distance	Percentage
1		0
2	334.8912	95.68
3	320.6296	91.61
4	308.0625	88.02
5	298.0370	85.15
6	291.4005	83.26
7	289.0000	82.57
8	291.4005	83.26
9	298.0370	85.15

10	308.0625	88.02
11	320.296	91.61
12	334.8912	95.68
13	350.0000	100.00

Tabla 1. Porcentaje de las alturas.

CÓDIGO

Utilicé el lenguaje de programación ActionScript 3.

```
import flash.events.MouseEvent;
import flash.globalization.NumberFormatter;
import flash.display.Sprite;
import flash.events.Event;
import flash.display.MovieClip;
import flash.utils.Timer;
import flash.events.TimerEvent;
import flash.display.Shape;

var nuevaLinea1 = new linea1;
addChild(nuevaLinea1);
nuevaLinea1.x = 250;
nuevaLinea1.y = 300;
valorXLinea1.text = String(Number(nuevaLinea1.x));
valorYLinea1.text = String(Number(nuevaLinea1.y));

var nuevoPunto1 = new punto1;
addChild(nuevoPunto1);
nuevoPunto1.x = 250;
nuevoPunto1.y = 550;
valorXPunto1.text = String(Number(nuevoPunto1.x));
valorYPunto1.text = String(Number(nuevoPunto1.y));

var largoLinea1: Number = 150;
var detenerArribaLinea1: Number = 150;
var inputAncho: Number = 0;
var inputLargo: Number = 0;

botonMovimientoPunto1.addEventListener(MouseEvent.MOUSE_DOWN,
movimientoPuntoLinea1);
function movimientoPuntoLinea1(pEvent: MouseEvent) {
        inputAncho = Number(anchoEntrada.text);
        inputLargo = Number(largoEntrada.text);
```

```
        nuevoPunto1.x -= inputAncho / 2;
        nuevoPunto1.y += inputLargo / 2;
        nuevaLinea1.x -= inputAncho / 2;
        valorXLinea1.text = String(Number(nuevaLinea1.x));
        valorYLinea1.text = String(Number(nuevaLinea1.y));
        valorXPunto1.text = String(Number(nuevoPunto1.x));
        valorYPunto1.text = String(Number(nuevoPunto1.y));
        var sprLineaVertical: Sprite = new Sprite();
        sprLineaVertical.graphics.lineStyle(1, 0x000000, 1);
        sprLineaVertical.graphics.moveTo(puntoOrigen.x - inputAncho / 2, puntoOrigen.y +
inputLargo / 2);
        sprLineaVertical.graphics.lineTo(puntoOrigen.x + inputAncho / 2, puntoOrigen.y +
inputLargo / 2);
        addChild(sprLineaVertical);
        var sprLineaHorizontal: Sprite = new Sprite();
        sprLineaHorizontal.graphics.lineStyle(1, 0x000000, 1);
        sprLineaHorizontal.graphics.moveTo(puntoOrigen.x - inputAncho / 2,
puntoOrigen.y + inputLargo / 2);
        sprLineaHorizontal.graphics.lineTo(puntoOrigen.x - inputAncho / 2, puntoOrigen.y
- inputLargo / 2);
        addChild(sprLineaHorizontal);
        botonMovimientoPunto1.removeEventListener(MouseEvent.MOUSE_DOWN,
movimientoPuntoLinea1);
}


botonGeneral.addEventListener(MouseEvent.MOUSE_UP, anchoLargoResultado);
function anchoLargoResultado(e: MouseEvent) {
        resultadoAncho.text = String(Number(anchoEntrada.text) /
Number(anchoDivisiones.text));
        var numAncho: Number = (Number(resultadoAncho.text));
        var valorAncho: Number = Number(int(numAncho));
        resultadoAnchoSinDecimales.text = ("distancia divisiones:" + valorAncho);
resultadoLargo.text = String(Number(largoEntrada.text) / Number(largoDivisiones.text));
        var numLargo: Number = (Number(resultadoLargo.text));
        var valorLargo: Number = Number(int(numLargo));
        resultadoLargoSinDecimales.text = ("distancia divisiones:" + valorLargo);

var numDivisionesAncho: Number = (Number(anchoDivisiones.text));
        var valorDivisionesAncho: Number = Number(int(numDivisionesAncho));

var numDivisionesLargo: Number = (Number(largoDivisiones.text));
        var valorDivisionesLargo: Number = Number(int(numDivisionesLargo));

residuoAncho.text = String(Number(anchoEntrada.text) % Number(anchoDivisiones.text));
```

```
        var numResiduoAncho: Number = (Number(residuoAncho.text));
        var valorResiduoAncho: Number = Number(int(numResiduoAncho));
        resultadoResiduoAnchoSinDecimales.text = ("distancia residuo:" +
valorResiduoAncho);

residuoLargo.text = String(Number(largoEntrada.text) % Number(largoDivisiones.text));
        var numResiduoLargo: Number = (Number(residuoLargo.text));
        var valorResiduoLargo: Number = Number(int(numResiduoLargo));
        resultadoResiduoLargoSinDecimales.text = ("distancia residuo:" +
valorResiduoLargo);

multiplicacionAnchoDivisiones.text = String(valorAncho * valorDivisionesAncho);
        var numMultiplicacionAnchoDivisionesAncho: Number =
(Number(multiplicacionAnchoDivisiones.text));
        var valorMultiplicacionAnchoDivisionesAncho: Number =
Number(int(numMultiplicacionAnchoDivisionesAncho));
        resultadoMultiplicacionAnchoDivisiones.text = ("valor distancia divisiones x
número de divisiones:" + valorMultiplicacionAnchoDivisionesAncho);

multiplicacionLargoDivisiones.text = String(valorLargo * valorDivisionesLargo);
        var numMultiplicacionLargoDivisionesLargo: Number =
(Number(multiplicacionLargoDivisiones.text));
        var valorMultiplicacionLargoDivisionesLargo: Number =
Number(int(numMultiplicacionLargoDivisionesLargo));
        resultadoMultiplicacionLargoDivisiones.text = ("valor distancia divisiones x número
de divisiones:" + valorMultiplicacionLargoDivisionesLargo);

var distanciaAncho: Number = valorAncho;
        var distanciaResiduoAncho: Number = valorResiduoAncho;
        var limiteLinea1: Number = valorMultiplicacionAnchoDivisionesAncho;
        var limitePunto1: Number = valorMultiplicacionLargoDivisionesLargo;

var distanciaLargo: Number = valorLargo;
        var distanciaResiduoLargo: Number = valorResiduoLargo;
        var velocidadLinea1: Number = 1;

//linea del lado izquierdo de la linea 1 en el eje Y
        var nuevaLinea2 = new linea2;
        nuevaLinea2.x = nuevoPunto1.x;
        nuevaLinea2.y = lineaSuperiorMesa.y;

//linea del lado derecho de la linea 1 en el eje Y
        var nuevaLinea3 = new linea3;
        nuevaLinea3.x = nuevaLinea2.x + inputAncho;
        nuevaLinea3.y = lineaSuperiorMesa.y;
```

```
var posicionXLinea2 = nuevaLinea2.x;
        var posicionYLinea2 = lineaSuperiorMesa.y;
        var posicionXLinea3 = nuevaLinea2.x + inputAncho;
        var posicionYLinea3 = lineaSuperiorMesa.y;

var valorPunto2: Number = 95.6832; //porcentaje de 334.89 que es la distancia de 0 a la curva en el centro
        var valor2Punto2 = (valorPunto2 * posicionYLinea2 / 100);
        var porcentajeLinea2 = (valor2Punto2 * 100 / posicionYLinea2);

var valorPunto3: Number = 91.6085; //porcentaje de 320.63 que es la distancia de 0 a la curva en el centro
        var valor2Punto3 = (valorPunto3 * posicionYLinea2 / 100);
        var porcentajeLinea3 = (valor2Punto3 * 100 / posicionYLinea2);

var valorPunto4: Number = 88.0179; //porcentaje de 320.63 que es la distancia de 0 a la curva en el centro
        var valor2Punto4 = (valorPunto4 * posicionYLinea2 / 100);
        var porcentajeLinea4 = (valor2Punto4 * 100 / posicionYLinea2);

var valorPunto5: Number = 85.1534; //porcentaje de 298.04 que es la distancia de 0 a la curva en el centro
        var valor2Punto5 = (valorPunto5 * posicionYLinea2 / 100);
        var porcentajeLinea5 = (valor2Punto5 * 100 / posicionYLinea2);

var valorPunto6: Number = 83.2573; //porcentaje de 291.4 que es la distancia de 0 a la curva en el centro
        var valor2Punto6 = (valorPunto6 * posicionYLinea2 / 100);
        var porcentajeLinea6 = (valor2Punto6 * 100 / posicionYLinea2);

var valorPunto7: Number = 82.5714; //porcentaje de 289 que es la distancia de 0 a la curva en el centro
        var valor2Punto7 = (valorPunto7 * posicionYLinea2 / 100);
        var porcentajeLinea7 = (valor2Punto7 * 100 / posicionYLinea2);

var valorPunto8: Number = 83.2573; //porcentaje de 291.4 que es la distancia de 0 a la curva en el centro
        var valor2Punto8 = (valorPunto8 * posicionYLinea2 / 100);
        var porcentajeLinea8 = (valor2Punto8 * 100 / posicionYLinea2);

var valorPunto9: Number = 85.1534; //porcentaje de 298.04 que es la distancia de 0 a la curva en el centro
        var valor2Punto9 = (valorPunto9 * posicionYLinea2 / 100);
        var porcentajeLinea9 = (valor2Punto9 * 100 / posicionYLinea2);
```

```
var valorPunto10: Number = 88.0179; //porcentaje de 308.06 que es la distancia de 0 a la curva en el centro
	var valor2Punto10 = (valorPunto10 * posicionYLinea2 / 100);
	var porcentajeLinea10 = (valor2Punto10 * 100 / posicionYLinea2);

var valorPunto11: Number = 91.6085; //porcentaje de 320.63 que es la distancia de 0 a la curva en el centro
	var valor2Punto11 = (valorPunto11 * posicionYLinea2 / 100);
	var porcentajeLinea11 = (valor2Punto11 * 100 / posicionYLinea2);

var valorPunto12: Number = 95.6832; //porcentaje de 334.89 que es la distancia de 0 a la curva en el centro
	var valor2Punto12 = (valorPunto12 * posicionYLinea2 / 100);
	var porcentajeLinea12 = (valor2Punto12 * 100 / posicionYLinea2);

var detenerArribaLinea: Number = 100;
	var velocidadArribaLinea: Number = 2;
	var velocidadDerechaLinea: Number = 2;
	var velocidadAbajoLinea: Number = 2;

var tiempo = 0;
	var timer: Timer = new Timer(1000, 200);
	timer.addEventListener(TimerEvent.TIMER, contarTiempo);
	timer.start();
	function contarTiempo(e: TimerEvent) {
		tiempo++;
		cajaTiempo.text = tiempo.toString();
	}

addEventListener(MouseEvent.MOUSE_DOWN, movimientoLineas);
	function movimientoLineas(e: MouseEvent) {
		botonGeneral.removeEventListener(MouseEvent.MOUSE_UP, anchoLargoResultado);
		nuevaLinea1.x += distanciaAncho;
		nuevoPunto1.x += distanciaAncho;
		var sprAncho: Sprite = new Sprite();
		sprAncho.graphics.lineStyle(1, 0x000000, 1);
		sprAncho.graphics.moveTo(nuevoPunto1.x, nuevoPunto1.y);
		sprAncho.graphics.lineTo(nuevoPunto1.x, nuevoPunto1.y - Number(largoEntrada.text));
		addChild(sprAncho);
		valorXLinea1.text = String(Number(nuevaLinea1.x));
		valorYLinea1.text = String(Number(nuevaLinea1.y));
		valorXPunto1.text = String(Number(nuevoPunto1.x));
```

```
                valorYPunto1.text = String(Number(nuevoPunto1.y));
if (nuevoPunto1.x == limiteLinea1 + (puntoOrigen.x - inputAncho / 2)) {
                        removeEventListener(MouseEvent.MOUSE_DOWN,
movimientoLineas);
                        addEventListener(MouseEvent.MOUSE_DOWN,
movimientoResiduoAncho);
                        function movimientoResiduoAncho(e: MouseEvent) {
                                nuevaLinea1.x += distanciaResiduoAncho;
                                nuevoPunto1.x += distanciaResiduoAncho;
                                var sprAnchoResiduo: Sprite = new Sprite();
                                sprAnchoResiduo.graphics.lineStyle(1, 0x000000, 1);
                                sprAnchoResiduo.graphics.moveTo(nuevoPunto1.x,
nuevoPunto1.y);
                                sprAnchoResiduo.graphics.lineTo(nuevoPunto1.x,
nuevoPunto1.y - Number(largoEntrada.text));
                                addChild(sprAnchoResiduo);
                                valorXLinea1.text = String(Number(nuevaLinea1.x));
                                valorYLinea1.text = String(Number(nuevaLinea1.y));
                                valorXPunto1.text = String(Number(nuevoPunto1.x));
                                valorYPunto1.text = String(Number(nuevoPunto1.y));
if (nuevoPunto1.x == (limiteLinea1 + (puntoOrigen.x - inputAncho / 2) +
distanciaResiduoAncho)) {
                                        removeEventListener(MouseEvent.MOUSE_DOWN,
movimientoResiduoAncho);
                                        addEventListener(MouseEvent.MOUSE_DOWN,
movimientoPuntoArriba);
                                        function movimientoPuntoArriba(e: MouseEvent) {
                                                nuevoPunto1.y -= distanciaLargo;
                                                var sprLargo: Sprite = new Sprite();
                                                sprLargo.graphics.lineStyle(1, 0x000000, 1);
                                                sprLargo.graphics.moveTo(nuevoPunto1.x,
nuevoPunto1.y);
                                                sprLargo.graphics.lineTo(nuevoPunto1.x -
inputAncho, nuevoPunto1.y);
                                                addChild(sprLargo);
                                                valorXPunto1.text =
String(Number(nuevoPunto1.x));
                                                valorYPunto1.text =
String(Number(nuevoPunto1.y));
if (nuevoPunto1.y == ((puntoOrigen.y + inputLargo / 2) - limitePunto1)) {

        removeEventListener(MouseEvent.MOUSE_DOWN, movimientoPuntoArriba);

        addEventListener(MouseEvent.MOUSE_DOWN, movimientoResiduoLargo);
```

```
                                                    function movimientoResiduoLargo(e:
MouseEvent) {
                                                        nuevoPunto1.y -=
distanciaResiduoLargo;
                                                        var sprLargoResiduo: Sprite =
new Sprite();

        sprLargoResiduo.graphics.lineStyle(1, 0x000000, 1);

        sprLargoResiduo.graphics.moveTo(nuevoPunto1.x, nuevoPunto1.y);

        sprLargoResiduo.graphics.lineTo(nuevoPunto1.x - inputAncho, nuevoPunto1.y);
                                                        addChild(sprLargoResiduo);
                                                        valorXPunto1.text =
String(Number(nuevoPunto1.x));
                                                        valorYPunto1.text =
String(Number(nuevoPunto1.y));

        removeEventListener(MouseEvent.MOUSE_DOWN, movimientoResiduoLargo);

        addEventListener(MouseEvent.MOUSE_DOWN, moverPuntosCurva);
function moverPuntosCurva(e: MouseEvent) {
                                                    if (nuevoPunto1.y ==
((puntoOrigen.y + inputLargo / 2) - (limitePunto1 + distanciaResiduoLargo))) {
                                                        nuevoPunto1.x -=
inputAncho;
                                                        nuevoPunto1.y
+= inputLargo;

        valorXPunto1.text = String(Number(nuevoPunto1.x));

        valorYPunto1.text = String(Number(nuevoPunto1.y));
                                                        nuevaLinea1.x -=
inputAncho;

        addChild(nuevaLinea2);

        addChild(nuevaLinea3); //curva acorde al largo de 300
                                                        //CIRCULOS
ROJOS
                                                        var circulo1:
Sprite = new Sprite();

        circulo1.graphics.beginFill(0x990000);
```

```
        circulo1.graphics.drawCircle(posicionXLinea2, posicionYLinea2, 5);

        circulo1.graphics.endFill();

        addChild(circulo1);
var circulo2: Sprite = new Sprite();

        circulo2.graphics.beginFill(0x990000);

        circulo2.graphics.drawCircle(posicionXLinea2 + distanciaAncho, (porcentajeLinea2
* posicionYLinea2) / 100, 5);

        circulo2.graphics.endFill();

        addChild(circulo2);
                                                          var        circulo3:
Sprite = new Sprite();

        circulo3.graphics.beginFill(0x990000);

        circulo3.graphics.drawCircle(posicionXLinea2    +    (distanciaAncho    *    2),
(porcentajeLinea3 * posicionYLinea2) / 100, 5);

        circulo3.graphics.endFill();

        addChild(circulo3);
                                                          var        circulo4:
Sprite = new Sprite();

        circulo4.graphics.beginFill(0x990000);

        circulo4.graphics.drawCircle(posicionXLinea2    +    (distanciaAncho    *    3),
(porcentajeLinea4 * posicionYLinea2) / 100, 5);

        circulo4.graphics.endFill();

        addChild(circulo4);
var circulo5: Sprite = new Sprite();

        circulo5.graphics.beginFill(0x990000);

        circulo5.graphics.drawCircle(posicionXLinea2    +    (distanciaAncho    *    4),
(porcentajeLinea5 * posicionYLinea2) / 100, 5);
```

```
        circulo5.graphics.endFill();

        addChild(circulo5);
                                                                    var circulo6:
Sprite = new Sprite();

        circulo6.graphics.beginFill(0x990000);

        circulo6.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 5),
(porcentajeLinea6 * posicionYLinea2) / 100, 5);

        circulo6.graphics.endFill();

        addChild(circulo6);
var circulo7: Sprite = new Sprite();

        circulo7.graphics.beginFill(0x990000);

        circulo7.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 6),
(porcentajeLinea7 * posicionYLinea2) / 100, 5);

        circulo7.graphics.endFill();

        addChild(circulo7);
                                                                    var circulo8:
Sprite = new Sprite();

        circulo8.graphics.beginFill(0x990000);

        circulo8.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 7),
(porcentajeLinea8 * posicionYLinea2) / 100, 5);

        circulo8.graphics.endFill();

        addChild(circulo8);
                                                                    var circulo9:
Sprite = new Sprite();

        circulo9.graphics.beginFill(0x990000);

        circulo9.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 8),
(porcentajeLinea9 * posicionYLinea2) / 100, 5);

        circulo9.graphics.endFill();
```

```
        addChild(circulo9);
var circulo10: Sprite = new Sprite();

        circulo10.graphics.beginFill(0x990000);

        circulo10.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 9),
(porcentajeLinea10 * posicionYLinea2) / 100, 5);

        circulo10.graphics.endFill();

        addChild(circulo10);
                                                        var circulo11:
Sprite = new Sprite();

        circulo11.graphics.beginFill(0x990000);

        circulo11.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 10),
(porcentajeLinea11 * posicionYLinea2) / 100, 5);

        circulo11.graphics.endFill();

        addChild(circulo11);
                                                        var circulo12:
Sprite = new Sprite();

        circulo12.graphics.beginFill(0x990000);

        circulo12.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 11),
(porcentajeLinea12 * posicionYLinea2) / 100, 5);

        circulo12.graphics.endFill();

        addChild(circulo12);
var circulo13: Sprite = new Sprite();

        circulo13.graphics.beginFill(0x990000);

        circulo13.graphics.drawCircle(posicionXLinea3, posicionYLinea3, 5);

        circulo13.graphics.endFill();

        addChild(circulo13);
removeEventListener(MouseEvent.MOUSE_DOWN, moverPuntosCurva);
```

```
        addEventListener(Event.ENTER_FRAME, subirLineaY);
                                                                var circuloVerde:
Sprite = new Sprite();

        circuloVerde.graphics.beginFill(0x605858);

        circuloVerde.graphics.drawCircle(nuevaLinea1.x, (nuevaLinea1.y - 155), 5);

        circuloVerde.graphics.endFill();

        addChild(circuloVerde);
var curva: Sprite = new Sprite();

        curva.graphics.lineStyle(1, 0x605858, 1);

        curva.graphics.moveTo(posicionXLinea2, posicionYLinea2); //circulo1

        curva.graphics.lineTo(posicionXLinea2 + distanciaAncho, (porcentajeLinea2 *
posicionYLinea2) / 100); //circulo2

        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 2), (porcentajeLinea3 *
posicionYLinea2) / 100); //circulo3

        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 3), (porcentajeLinea4 *
posicionYLinea2) / 100); //circulo4

        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 4), (porcentajeLinea5 *
posicionYLinea2) / 100); //circulo5

        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 5), (porcentajeLinea6 *
posicionYLinea2) / 100); //circulo6

        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 6), (porcentajeLinea7 *
posicionYLinea2) / 100); //circulo7

        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 7), (porcentajeLinea8 *
posicionYLinea2) / 100); //circulo8

        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 8), (porcentajeLinea9 *
posicionYLinea2) / 100); //circulo9

        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 9), (porcentajeLinea10
* posicionYLinea2) / 100); //circulo10
```

```
        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 10), (porcentajeLinea11
* posicionYLinea2) / 100); //circulo11

        curva.graphics.lineTo(posicionXLinea2 + (distanciaAncho * 11), (porcentajeLinea12
* posicionYLinea2) / 100); //circulo12

        curva.graphics.lineTo(posicionXLinea3, posicionYLinea3); //circulo13
                                                                    addChild(curva);
function subirLineaY(e: Event) {

        nuevaLinea1.y -= velocidadArribaLinea;

        valorXLinea1.text = String(Number(nuevaLinea1.x));

        valorYLinea1.text = String(Number(nuevaLinea1.y));
                                                                    if
(nuevaLinea1.y <= detenerArribaLinea) {

        nuevaLinea1.y = detenerArribaLinea;

        removeEventListener(Event.ENTER_FRAME, subirLineaY);

        addEventListener(Event.ENTER_FRAME, moverLineaX);

        function moverLineaX(e: Event) {

        nuevaLinea1.x += distanciaAncho;

        valorXLinea1.text = String(Number(nuevaLinea1.x));

        valorYLinea1.text = String(Number(nuevaLinea1.y));

        removeEventListener(Event.ENTER_FRAME, moverLineaX);

        addEventListener(Event.ENTER_FRAME, bajarLineaY);
function bajarLineaY(e: Event) {

                nuevoPunto1.x = nuevaLinea1.x;

                valorXPunto1.text = String(Number(nuevoPunto1.x));

                valorYPunto1.text = String(Number(nuevoPunto1.y));
```

```
var circulos: Array = new Array(circulo1, circulo2, circulo3, circulo4, circulo5, circulo6, circulo7, circulo8, circulo9, circulo10, circulo11, circulo12, circulo13);

nuevaLinea1.y += velocidadAbajoLinea;

valorXLinea1.text = String(Number(nuevaLinea1.x));

valorYLinea1.text = String(Number(nuevaLinea1.y));

for each(var circulo in circulos) {
if (nuevaLinea1.hitTestObject(circulo)) {

nuevaLinea1.y == circulo.y;

//CIRCULOS GRISES ARRIBA

var circuloVerde: Sprite = new Sprite();

circuloVerde.graphics.beginFill(0x605858);

circuloVerde.graphics.drawCircle(nuevaLinea1.x, (nuevaLinea1.y - 150), 5);

circuloVerde.graphics.endFill();

addChild(circuloVerde);

removeEventListener(Event.ENTER_FRAME, bajarLineaY);

addEventListener(Event.ENTER_FRAME, subirLineaY);
if (nuevaLinea1.x == nuevaLinea3.x) {

removeEventListener(Event.ENTER_FRAME, subirLineaY);

var circulo1Gris: Sprite = new Sprite();

circulo1Gris.graphics.beginFill(0x605858);

circulo1Gris.graphics.drawCircle(posicionXLinea2, ((posicionYLinea2) - 155), 5);

circulo1Gris.graphics.endFill();
```

```
var circulo2Gris: Sprite = new Sprite();

circulo2Gris.graphics.beginFill(0x605858);

circulo2Gris.graphics.drawCircle(posicionXLinea2 + distanciaAncho, ((porcentajeLinea2 * posicionYLinea2) / 100) - 155, 5);

circulo2Gris.graphics.endFill();

var circulo3Gris: Sprite = new Sprite();

circulo3Gris.graphics.beginFill(0x605858);

circulo3Gris.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 2), ((porcentajeLinea3 * posicionYLinea2) / 100) - 155, 5);

circulo3Gris.graphics.endFill();

var circulo4Gris: Sprite = new Sprite();

circulo4Gris.graphics.beginFill(0x605858);

circulo4Gris.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 3), ((porcentajeLinea4 * posicionYLinea2) / 100) - 155, 5);

circulo4Gris.graphics.endFill();

var circulo5Gris: Sprite = new Sprite();

circulo5Gris.graphics.beginFill(0x605858);

circulo5Gris.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 4), ((porcentajeLinea5 * posicionYLinea2) / 100) - 155, 5);

circulo5Gris.graphics.endFill();

var circulo6Gris: Sprite = new Sprite();

circulo6Gris.graphics.beginFill(0x605858);

circulo6Gris.graphics.drawCircle(posicionXLinea2 + (distanciaAncho * 5), ((porcentajeLinea6 * posicionYLinea2) / 100) - 155, 5);

circulo6Gris.graphics.endFill();
```

```
                    var circulo7Gris: Sprite = new Sprite();

                    circulo7Gris.graphics.beginFill(0x605858);

                    circulo7Gris.graphics.drawCircle(posicionXLinea2 +
(distanciaAncho * 6), ((porcentajeLinea7 * posicionYLinea2) / 100) - 155, 5);

                    circulo7Gris.graphics.endFill();

                    var circulo8Gris: Sprite = new Sprite();

                    circulo8Gris.graphics.beginFill(0x605858);

                    circulo8Gris.graphics.drawCircle(posicionXLinea2 +
(distanciaAncho * 7), ((porcentajeLinea8 * posicionYLinea2) / 100) - 155, 5);

                    circulo8Gris.graphics.endFill();

                    var circulo9Gris: Sprite = new Sprite();

                    circulo9Gris.graphics.beginFill(0x605858);

                    circulo9Gris.graphics.drawCircle(posicionXLinea2 +
(distanciaAncho * 8), ((porcentajeLinea9 * posicionYLinea2) / 100) - 155, 5);

                    circulo9Gris.graphics.endFill();

                    var circulo10Gris: Sprite = new Sprite();

                    circulo10Gris.graphics.beginFill(0x605858);

                    circulo10Gris.graphics.drawCircle(posicionXLinea2 +
(distanciaAncho * 9), ((porcentajeLinea10 * posicionYLinea2) / 100) - 155, 5);

                    circulo10Gris.graphics.endFill();

                    var circulo11Gris: Sprite = new Sprite();

                    circulo11Gris.graphics.beginFill(0x605858);

                    circulo11Gris.graphics.drawCircle(posicionXLinea2 +
(distanciaAncho * 10), ((porcentajeLinea11 * posicionYLinea2) / 100) - 155, 5);

                    circulo11Gris.graphics.endFill();
```

```
                    var circulo12Gris: Sprite = new Sprite();

                    circulo12Gris.graphics.beginFill(0x605858);

                    circulo12Gris.graphics.drawCircle(posicionXLinea2 +
(distanciaAncho * 11), ((porcentajeLinea12 * posicionYLinea2) / 100) - 155, 5);

                    circulo12Gris.graphics.endFill();

                    var circulo13Gris: Sprite = new Sprite();

                    circulo13Gris.graphics.beginFill(0x605858);

                    circulo13Gris.graphics.drawCircle(posicionXLinea3,
((posicionYLinea3) - 155), 5);

                    circulo13Gris.graphics.endFill();

                    addChild(circulo1Gris);

                    var curva2: Sprite = new Sprite();

                    curva2.graphics.lineStyle(1, 0x605858, 1);

                    curva2.graphics.moveTo(posicionXLinea2,
((posicionYLinea2) - 155)); //circulo1

                    curva2.graphics.lineTo(posicionXLinea2          +
distanciaAncho, ((porcentajeLinea2 * posicionYLinea2) / 100) - 155); //circulo2

                    curva2.graphics.lineTo(posicionXLinea2          +
(distanciaAncho * 2), ((porcentajeLinea3 * posicionYLinea2) / 100) - 155); //circulo3

                    curva2.graphics.lineTo(posicionXLinea2          +
(distanciaAncho * 3), ((porcentajeLinea4 * posicionYLinea2) / 100) - 155); //circulo4

                    curva2.graphics.lineTo(posicionXLinea2          +
(distanciaAncho * 4), ((porcentajeLinea5 * posicionYLinea2) / 100) - 155); //circulo5

                    curva2.graphics.lineTo(posicionXLinea2          +
(distanciaAncho * 5), ((porcentajeLinea6 * posicionYLinea2) / 100) - 155); //circulo6

                    curva2.graphics.lineTo(posicionXLinea2          +
(distanciaAncho * 6), ((porcentajeLinea7 * posicionYLinea2) / 100) - 155); //circulo7
```

```
                curva2.graphics.lineTo(posicionXLinea2 +
(distanciaAncho * 7), ((porcentajeLinea8 * posicionYLinea2) / 100) - 155); //circulo8

                curva2.graphics.lineTo(posicionXLinea2 +
(distanciaAncho * 8), ((porcentajeLinea9 * posicionYLinea2) / 100) - 155); //circulo9

                curva2.graphics.lineTo(posicionXLinea2 +
(distanciaAncho * 9), ((porcentajeLinea10 * posicionYLinea2) / 100) - 155); //circulo10

                curva2.graphics.lineTo(posicionXLinea2 +
(distanciaAncho * 10), ((porcentajeLinea11 * posicionYLinea2) / 100) - 155); //circulo11

                curva2.graphics.lineTo(posicionXLinea2 +
(distanciaAncho * 11), ((porcentajeLinea12 * posicionYLinea2) / 100) - 155); //circulo12

                curva2.graphics.lineTo(posicionXLinea3,
(posicionYLinea3) - 155); //circulo13

                addChild(curva2);

                valorXPunto1Verde.text =
String(Number(posicionXLinea2));

                valorYPunto1Verde.text =
String(Number((posicionYLinea2) - 155));

                addChild(circulo2Gris);

                valorXPunto2Verde.text =
String(Number(posicionXLinea2 + distanciaAncho));

                valorYPunto2Verde.text =
String(Number((porcentajeLinea2 * posicionYLinea2) / 100) - 155);

                addChild(circulo3Gris);

                valorXPunto3Verde.text =
String(Number(posicionXLinea2 + (distanciaAncho * 2)));

                valorYPunto3Verde.text =
String(Number((porcentajeLinea3 * posicionYLinea2) / 100) - 155);

                addChild(circulo4Gris);
```

```
                    valorXPunto4Verde.text =
String(Number(posicionXLinea2 + (distanciaAncho * 3)));

                    valorYPunto4Verde.text =
String(Number((porcentajeLinea4 * posicionYLinea2) / 100) - 155);

                    addChild(circulo5Gris);

                    valorXPunto5Verde.text =
String(Number(posicionXLinea2 + (distanciaAncho * 4)));

                    valorYPunto5Verde.text =
String(Number((porcentajeLinea5 * posicionYLinea2) / 100) - 155);

                    addChild(circulo6Gris);

                    valorXPunto6Verde.text =
String(Number(posicionXLinea2 + (distanciaAncho * 5)));

                    valorYPunto6Verde.text =
String(Number((porcentajeLinea6 * posicionYLinea2) / 100) - 155);

                    addChild(circulo7Gris);

                    valorXPunto7Verde.text =
String(Number(posicionXLinea2 + (distanciaAncho * 6)));

                    valorYPunto7Verde.text =
String(Number((porcentajeLinea7 * posicionYLinea2) / 100) - 155);

                    addChild(circulo8Gris);

                    valorXPunto8Verde.text =
String(Number(posicionXLinea2 + (distanciaAncho * 7)));

                    valorYPunto8Verde.text =
String(Number((porcentajeLinea8 * posicionYLinea2) / 100) - 155);

                    addChild(circulo9Gris);

                    valorXPunto9Verde.text =
String(Number(posicionXLinea2 + (distanciaAncho * 8)));
```

```
                                        valorYPunto9Verde.text                  =
String(Number((porcentajeLinea9 * posicionYLinea2) / 100) - 155);

                                        addChild(circulo10Gris);

                                        valorXPunto10Verde.text                 =
String(Number(posicionXLinea2 + (distanciaAncho * 9)));

                                        valorYPunto10Verde.text                 =
String(Number((porcentajeLinea10 * posicionYLinea2) / 100) - 155);

                                        addChild(circulo11Gris);

                                        valorXPunto11Verde.text                 =
String(Number(posicionXLinea2 + (distanciaAncho * 10)));

                                        valorYPunto11Verde.text                 =
String(Number((porcentajeLinea11 * posicionYLinea2) / 100) - 155);

                                        addChild(circulo12Gris);

                                        valorXPunto12Verde.text                 =
String(Number(posicionXLinea2 + (distanciaAncho * 11)));

                                        valorYPunto12Verde.text                 =
String(Number((porcentajeLinea12 * posicionYLinea2) / 100) - 155);

                                        addChild(circulo13Gris);

                                        valorXPunto13Verde.text                 =
String(Number(posicionXLinea3));

                                        valorYPunto13Verde.text                 =
String(Number((posicionYLinea3) - 155));

                                        addEventListener(Event.ENTER_FRAME,
tiempoTotal);
function tiempoTotal(e: Event) {

                                                if (nuevaLinea1.x == nuevaLinea3.x) {

                                                        timer.stop();
```

```
                                                    cajaTiempo.text = "tiempo transcurrido"
+ " " + tiempo.toString() + " " + "segundos";

                                                    var textToSave: String;

                                                    var fileRef: FileReference;


        botonDatos.addEventListener(MouseEvent.CLICK, savefunction);

                                                    function savefunction(e: MouseEvent) {

                                                            textToSave                    =
valorXPunto1Verde.text + "," + valorYPunto1Verde.text + "\n" +

                                                                    valorXPunto2Verde.text
+ "," + valorYPunto2Verde.text + "\n" +

                                                                    valorXPunto3Verde.text
+ "," + valorYPunto3Verde.text + "\n" +

                                                                    valorXPunto4Verde.text
+ "," + valorYPunto4Verde.text + "\n" +

                                                                    valorXPunto5Verde.text
+ "," + valorYPunto5Verde.text + "\n" +

                                                                    valorXPunto6Verde.text
+ "," + valorYPunto6Verde.text + "\n" +

                                                                    valorXPunto7Verde.text
+ "," + valorYPunto7Verde.text + "\n" +

                                                                    valorXPunto8Verde.text
+ "," + valorYPunto8Verde.text + "\n" +

                                                                    valorXPunto9Verde.text
+ "," + valorYPunto9Verde.text + "\n" +


        valorXPunto10Verde.text + "," + valorYPunto10Verde.text + "\n" +


        valorXPunto11Verde.text + "," + valorYPunto11Verde.text + "\n" +
```

```
        valorXPunto12Verde.text + "," + valorYPunto12Verde.text + "\n" +

        valorXPunto13Verde.text + "," + valorYPunto13Verde.text + "\n";
                                                        fileRef = new FileReference;
                                                        fileRef.save(textToSave,
"matematicas5_4_5");
        }
                                                }
                                        }
                                }
                        }
                }
        }
                                                                                }
                                                                        }
                                                                }
                                                        }
```

La simulación del mecanismo funciona de la siguiente manera:

Se traza el tamaño de la base del primer boceto, marcando el punto 0,0 que es el punto centro de la base y una línea azul que indica el cambio de planos el horizontal y el vertical.

Los valores que se escribirán son un ejemplo de cómo funciona la programación y la simulación.

(1) Escribimos el ancho y el largo de la base del exoesqueleto, 300 unidades de ancho y 200 unidades de largo.
(2) La línea y el punto rojos están en las coordenadas 0,0. Cuando se presiona el "primer botón", cambian su posición a las siguientes coordenadas: punto x = 100, y = 650, y línea x = 100, y = 300. Los valores numéricos se pueden ver en los recuadros

correspondientes. Aparecen dos líneas negras en la vista horizontal de la proyección ortogonal, que indican el ancho y el largo del exoesqueleto.
(3) Escribimos el número de divisiones, 12 de ancho y 6 de largo.
(4) Cuando se presiona el "botón general", en cada una de las casillas se ve el resultado y residuo de cada uno de los ejes. También se ven las divisiones de valor de distancia por número de divisiones. El tiempo comienza a correr (Figura13).

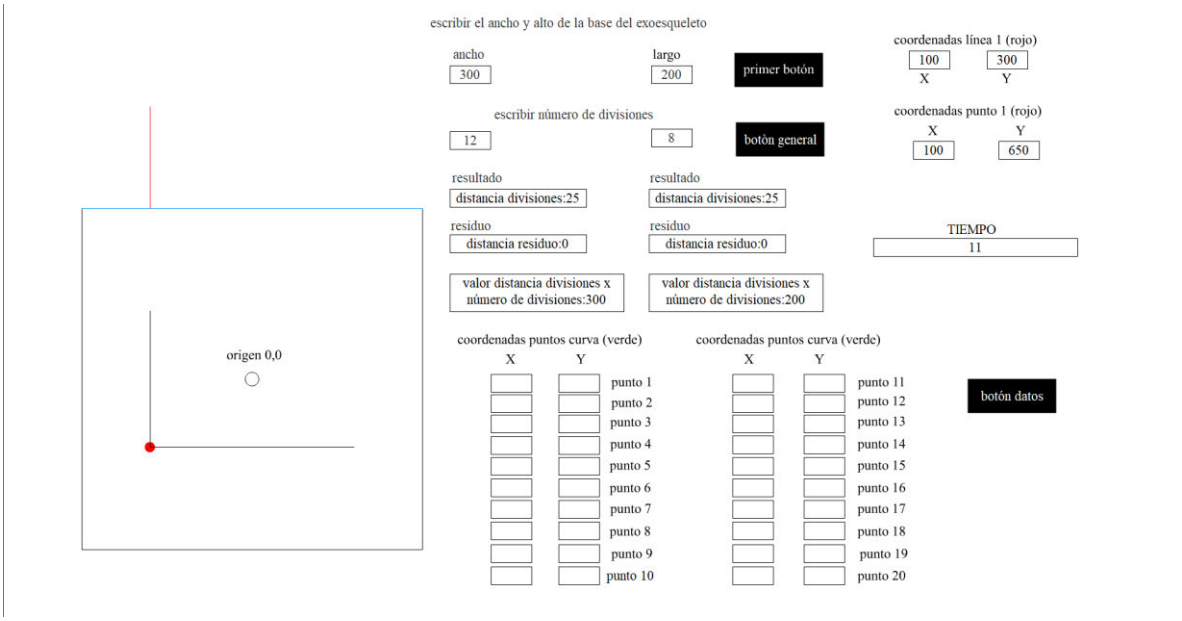


Figura 13. Resultados y residuos.

(5) Pulsamos de nuevo el "botón general" el mismo número de divisiones, 12 veces para el ancho y 6 veces para el largo. La línea y el punto rojos se mueven a la coordenada correspondiente a cada división. Al mismo tiempo, se puede ver las líneas de cuadrícula en la vista horizontal de la proyección ortogonal. La curva del exoesqueleto aparece con los puntos que indican las 12 divisiones en la vista vertical de la proyección ortogonal.

(6) Comienza la simulación. El punto y la línea roja simulan el puntero. La línea roja se mueve de abajo hacia arriba, el punto rojo se mueve hacia la derecha tantas veces como el número de divisiones. Cada vez que el puntero toca un punto de la curva del exoesqueleto, simula los puntos marcados en el papel.

(7) Una vez finalizada la animación, en los recuadros de coordenadas del punto de la curva aparecen los valores numéricos de las coordenadas x , y de cada uno de los puntos (Figura 14).

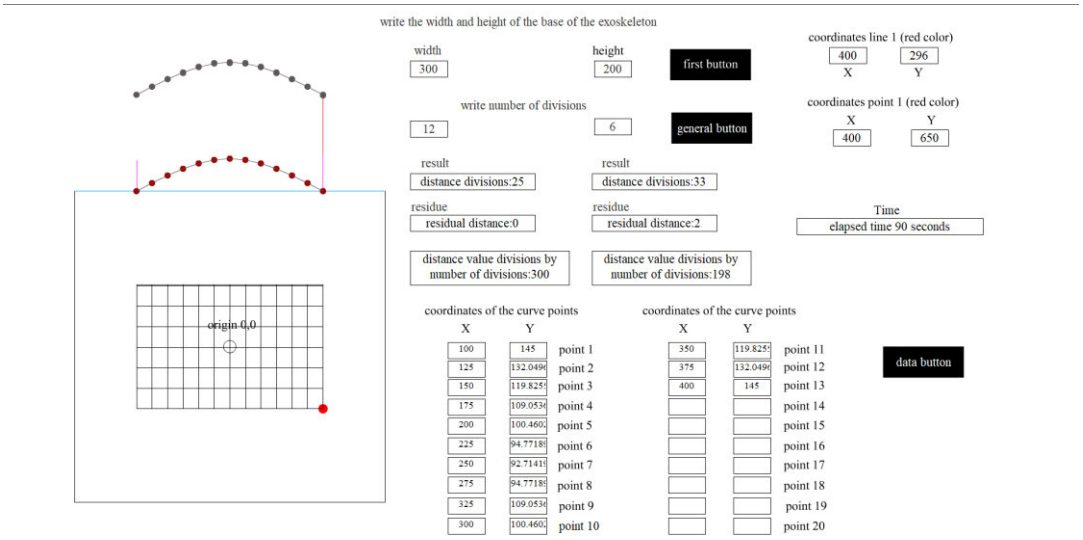


Figura 14. Resultados de las coordenadas x e y de los puntos de referencia.

(8) Al pulsar el “botón de datos" se guardan automáticamente en el bloc de notas los valores numéricos de las coordenadas x, y de cada uno de los puntos. Quedan dos valores después del punto decimal (Figura 15).

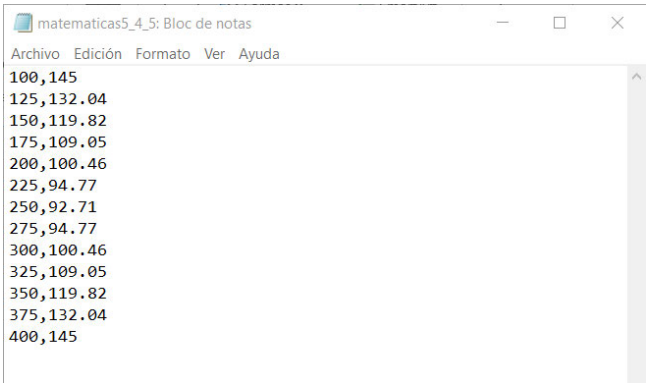


Figura 15. Valores numéricos de las coordenadas x, y, en el block de notas.

(9) Los valores numéricos de las coordenadas x, y se copian y pasan a AutoCAD™ para dibujar la curva. Los puntos de la curva están al revés porque el origen del programa Adobe Animate 2022™ está en la parte superior y en AutoCAD™ está en la parte inferior. Entonces, los puntos se giran 180° para posicionarlos correctamente en la proyección isométrica. Dibujamos la curva en la primera línea de la cuadrícula (Figura 16).

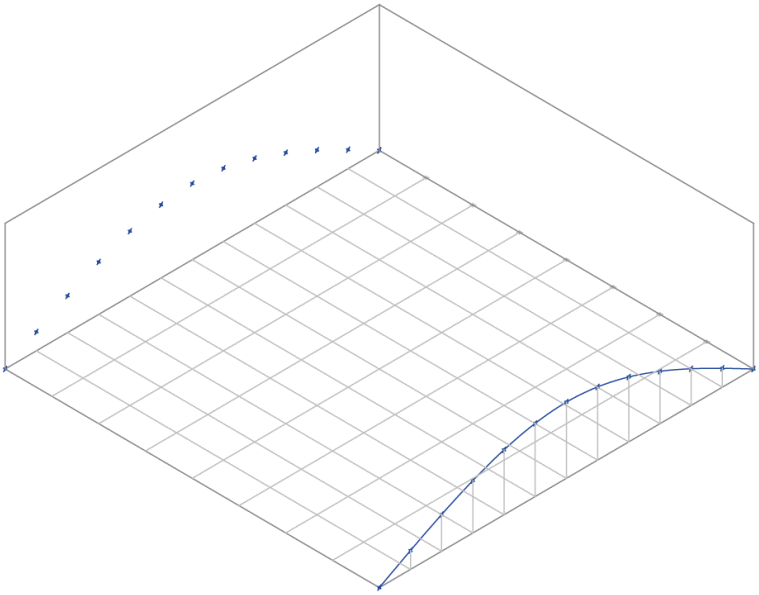


Figura 16. Proyección isométrica de la curva del exoesqueleto.

SE ANEXA EL ARCHIVO MP4 PARA VER LA SIMULACIÓN.

PRIMER MODELO

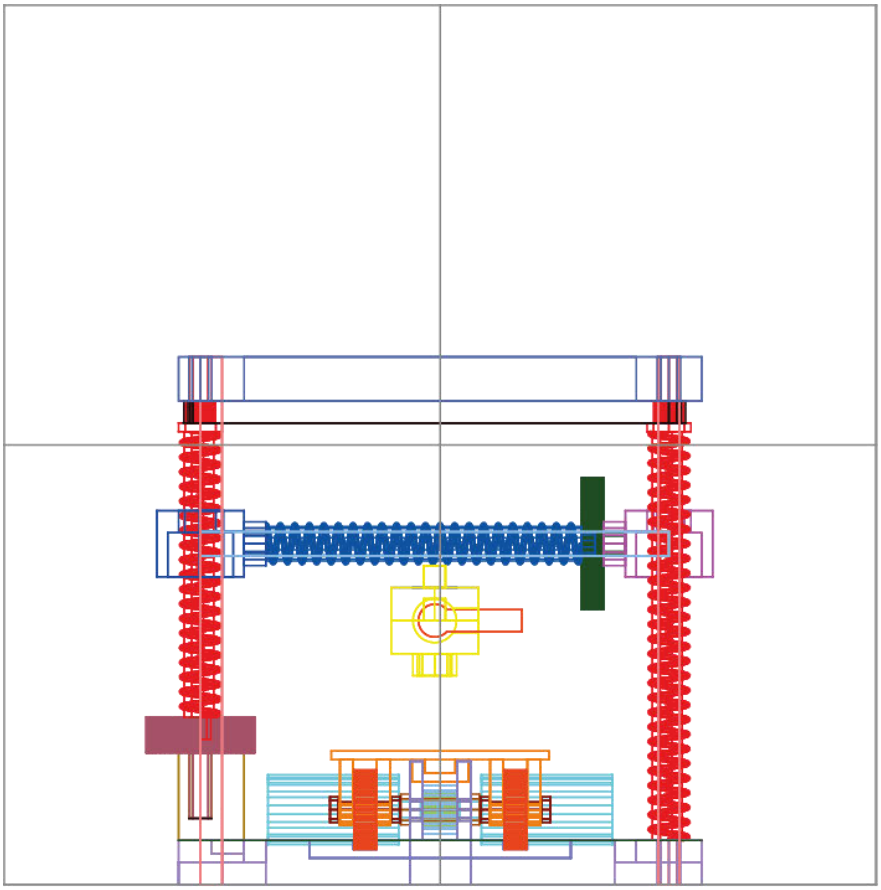


Figura 17.

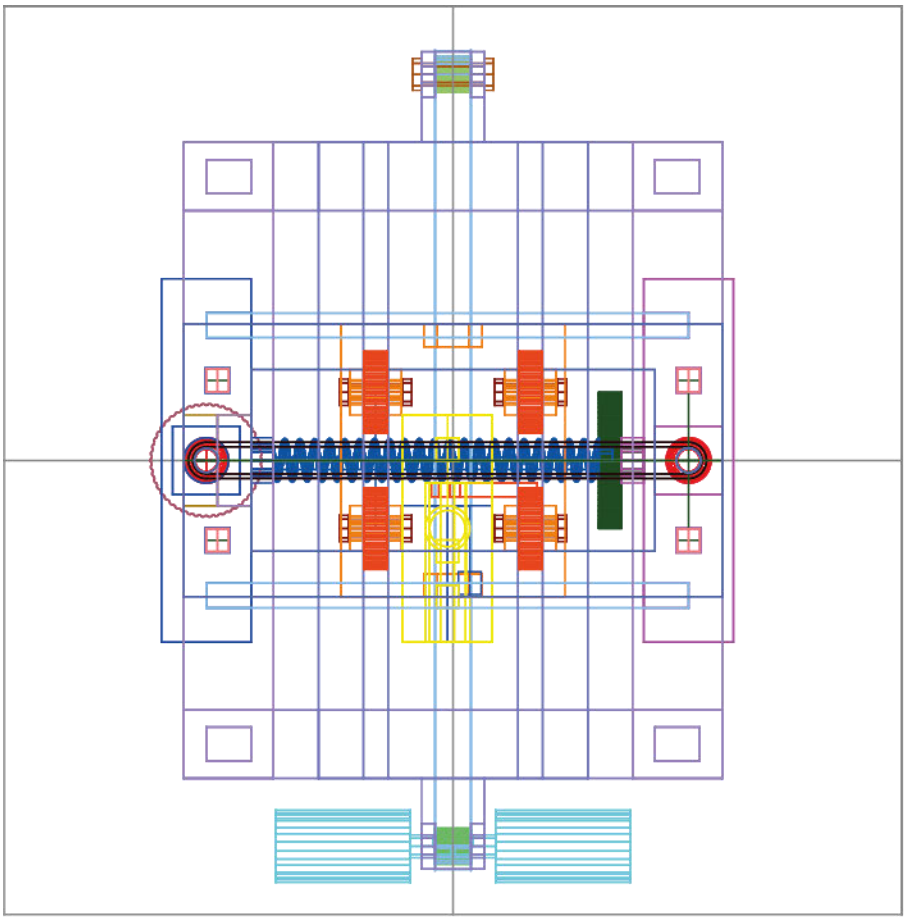

Proyecto “Geometría en Movimiento 3”
Pantógrafo XYZ
vista superior modelo 1
Dra. Dina Rochman Beer
2021

Figura 18.

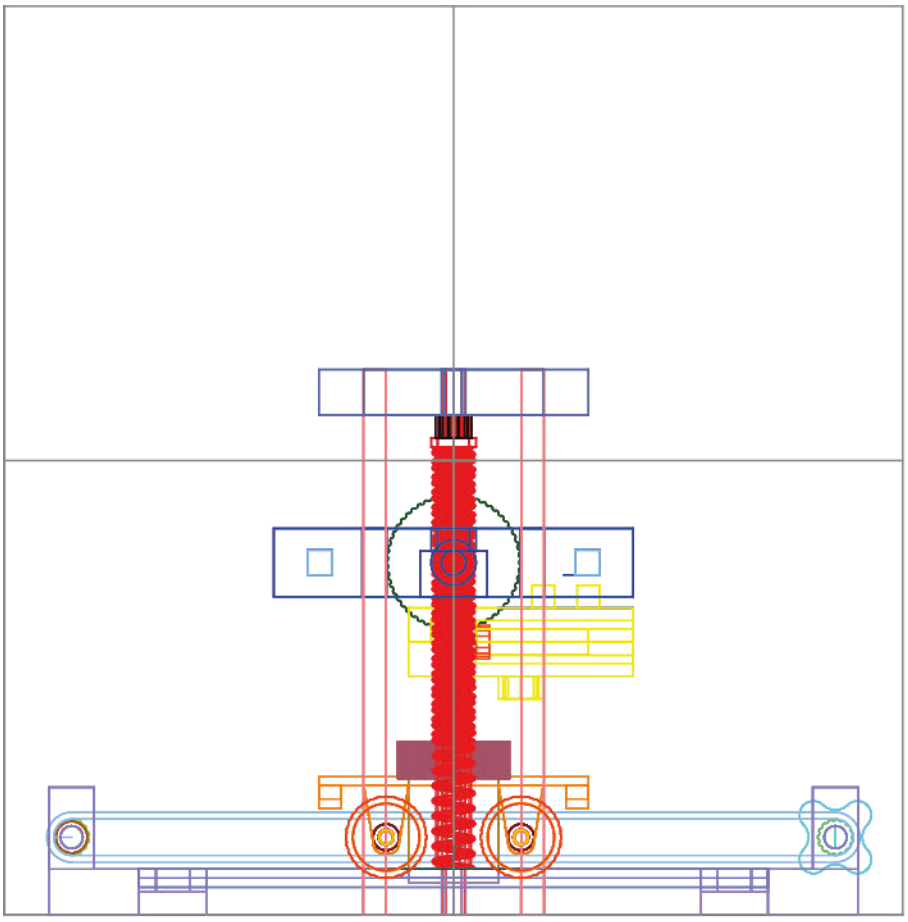

Proyecto “Geometría en Movimiento 3”
Pantógrafo XYZ
vista lateral modelo 1
Dra. Dina Rochman Beer
2021

Figura 19.

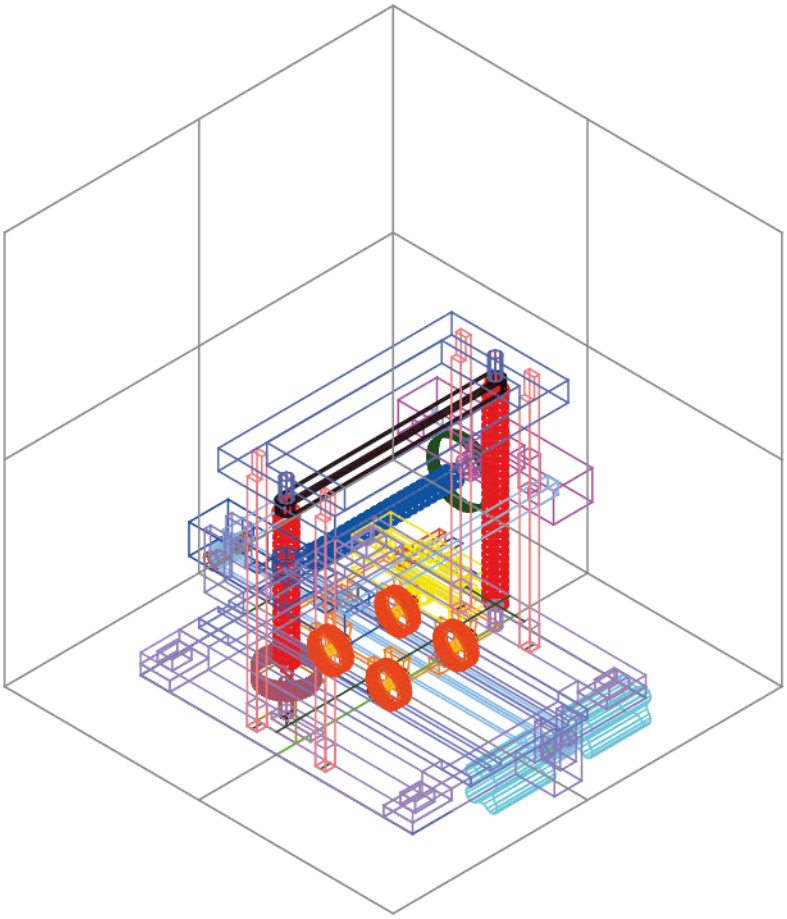

Proyecto “Geometría en Movimiento 3”
Pantógrafo XYZ
vista isométrico modelo 1
Dra. Dina Rochman Beer
2021

Figura 20.

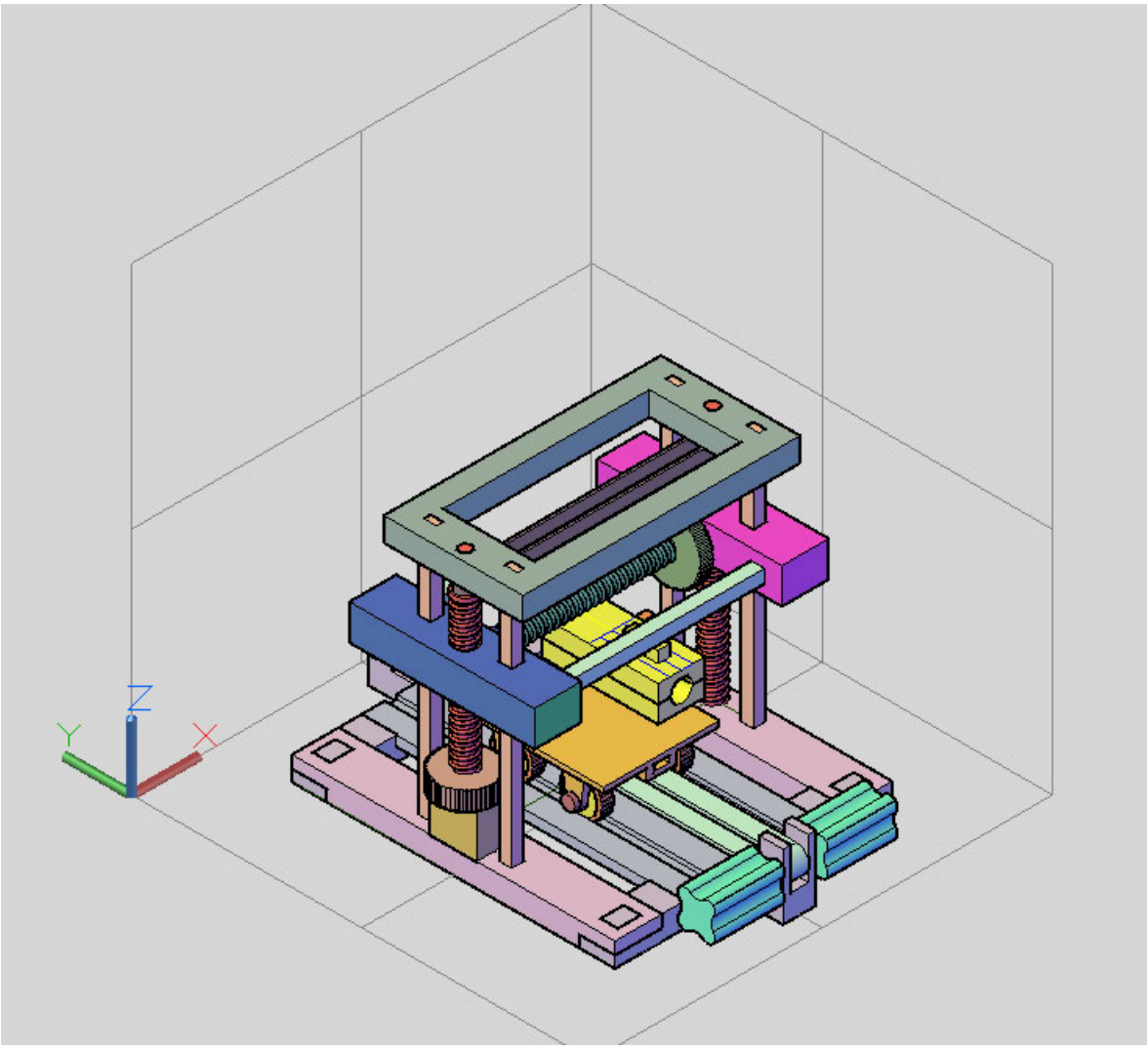


Figura 21. VISTA SUROESTE DEL MODELO 1

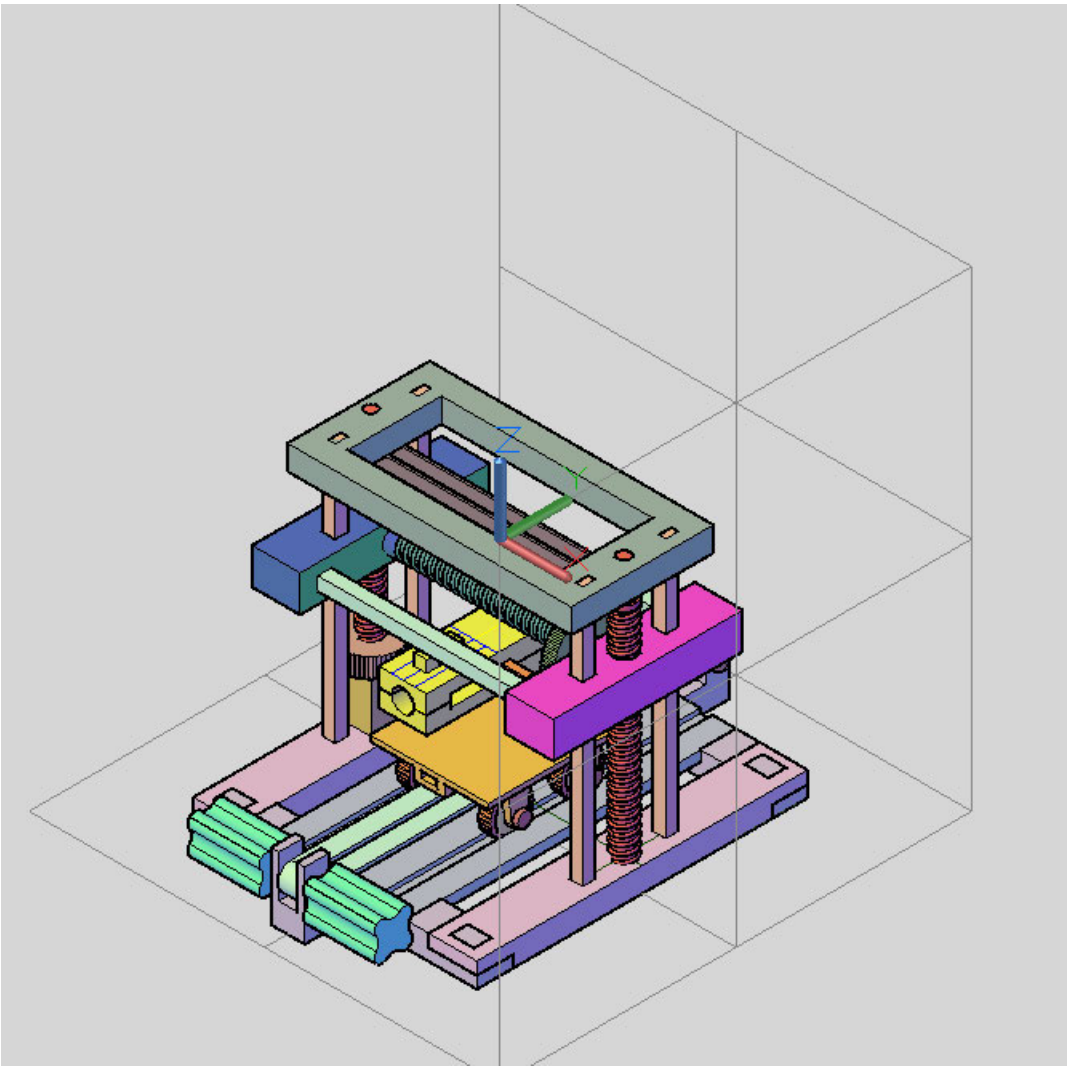


Figura 22. VISTA SURESTE DEL MODELO 1

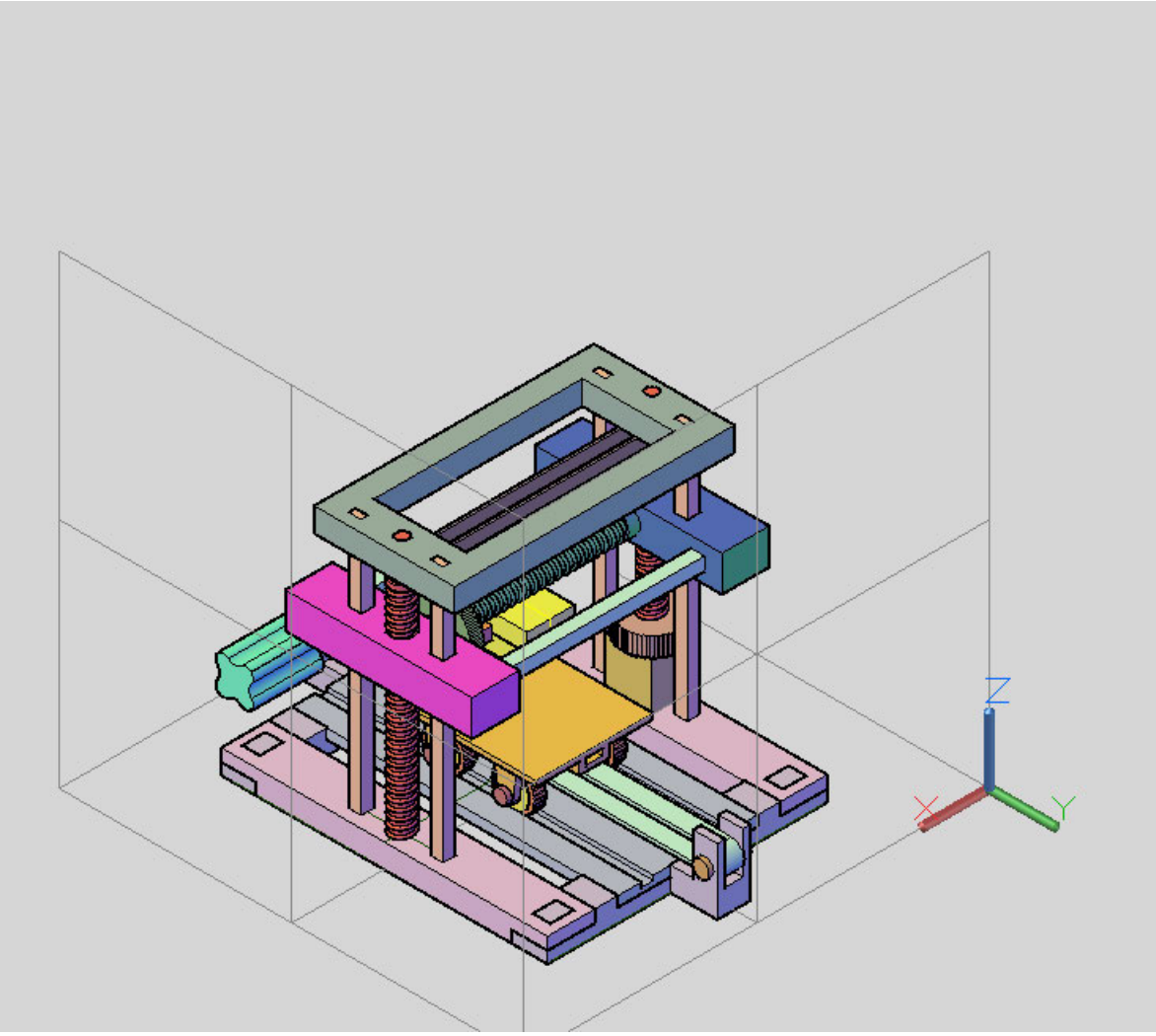


Figura 23. VISTA NORESTE DEL MODELO 1

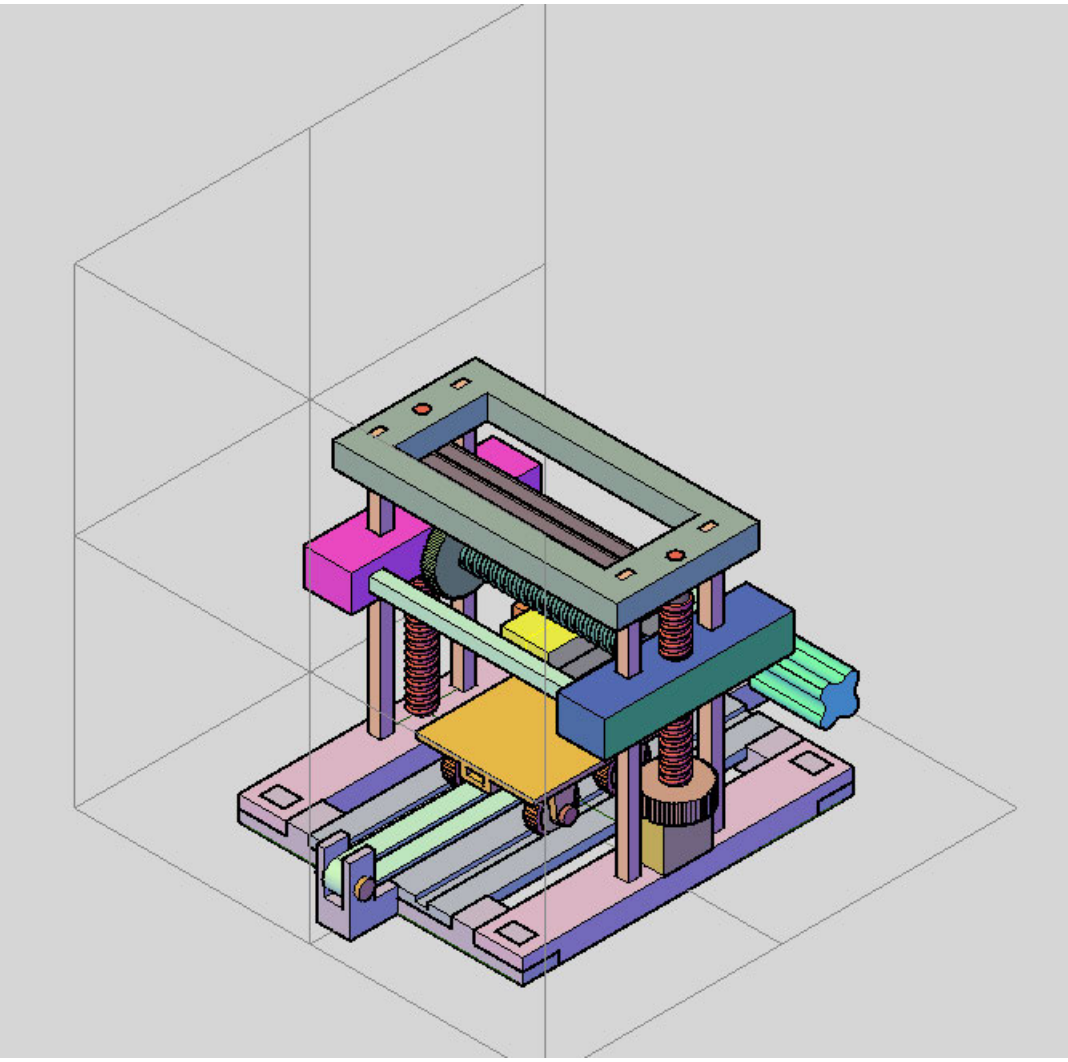
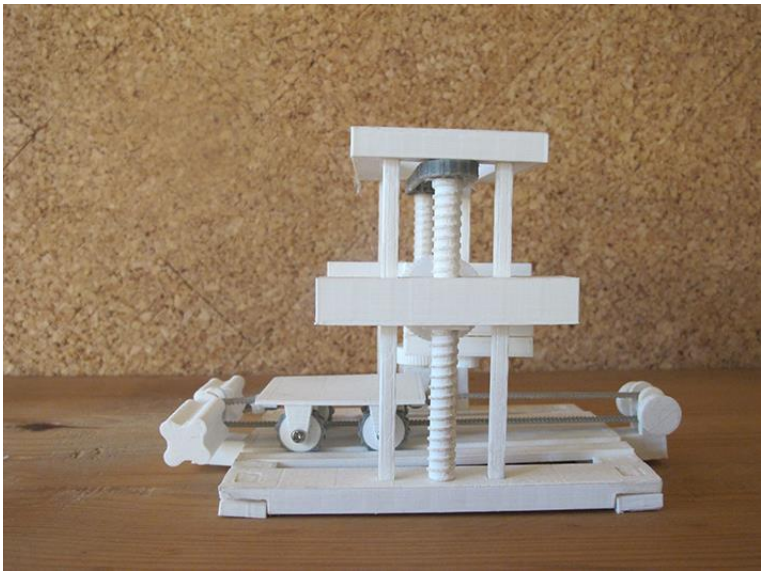
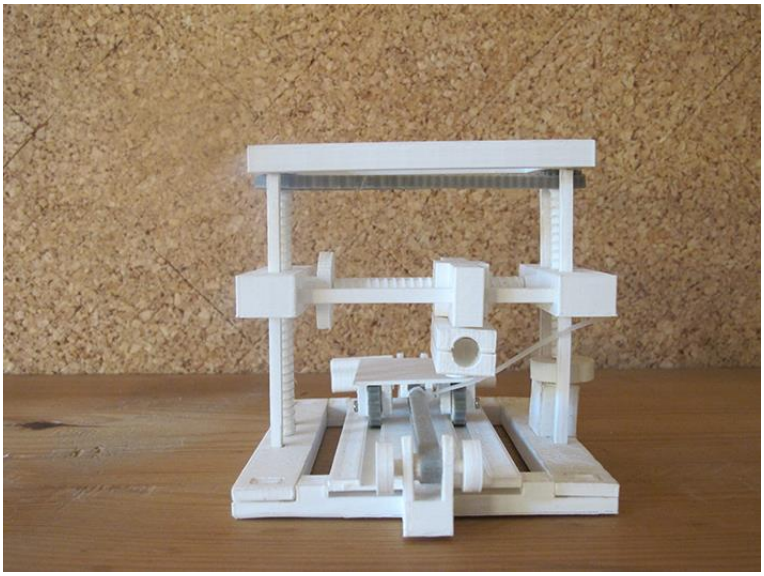
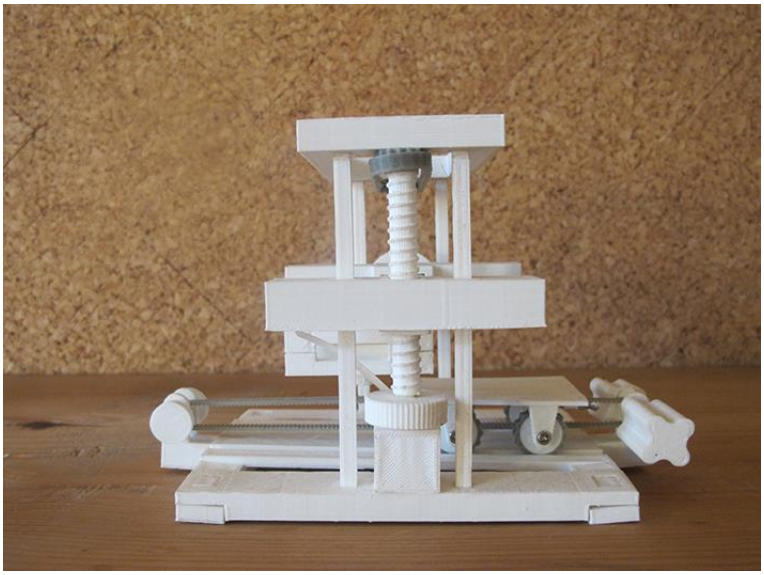


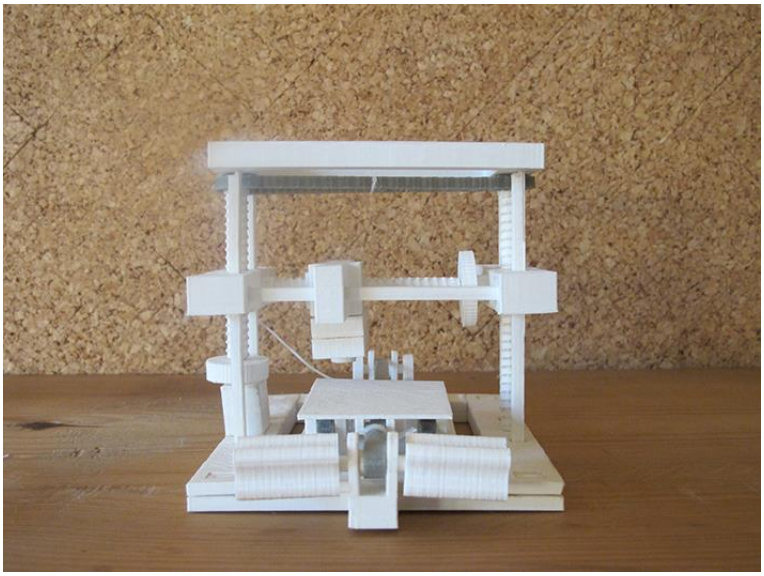
Figura 24. VISTA NOROESTE DEL MODELO1

IMPRESIÓN 3D DEL PRIMER MODELO









RESULTADOS

Los pantógrafos, que habían existido de una forma u otra durante cientos de años, estaban a punto de sacudir el mundo del grabado, que hasta ese momento había estado dominado por un grupo pequeño, pero ferozmente independiente de grabadores manuales . Durante siglos, el pantógrafo se había utilizado principalmente para fines bidimensionales o "planos", lo que significa que el trazador y el cortador viajaban en un plano bidimensional básicamente plano. Pero las máquinas europeas cambiaron todo eso. Conocidas como máquinas de grabado industrial de "servicio pesado", eran literalmente pesadas, con un peso de más de 1000 libras. Las máquinas industriales se utilizaron principalmente para aplicaciones a gran escala, como la fabricación de moldes para neumáticos vulcanizados, zapatos, platería, hebillas de cinturón, diales, latas, troqueles de extrusión y acuñación.

El pantógrafo tridimensional expuesto es para el estudio de insectos y moluscos. Geométricamente, utilizamos el método de representación de Gaspard Monge en su diseño (Figura 25).

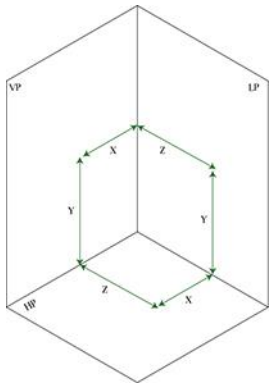


Figura 25. Método de representación de Gaspard Monge.

En su solución consideramos las etapas de síntesis y simulación. En la etapa de síntesis, nos enfocamos en la funcionalidad del mecanismo, que hasta ahora es manual. La mesa con ruedas se mueve hacia adelante y hacia atrás en el eje z a través de una banda. La base rectangular con el puntero y la punta del lápiz está en un riel que se mueve hacia la izquierda y hacia la derecha en el eje x usando una varilla roscada Acme, y las dos varillas roscadas Acme mueven el riel hacia arriba y hacia abajo en el eje y.

En la etapa de simulación, nos enfocamos en encontrar los valores numéricos de las coordenadas x, y de los puntos de referencia de las curvas del exoesqueleto usando un programa de computadora que diseñamos.

En el programa Adobe Animate 2022™ se realizó tanto la animación como la programación, utilizando el lenguaje orientado a objetos ActionScript 3.0 para encontrar los puntos de la curva del exoesqueleto.

El tiempo que tarda el programa en encontrar los valores numéricos de las coordenadas x, y de la primera curva del exoesqueleto es de 90 segundos, 1,5 minutos. Entonces, en 1200 segundos, es decir, 15 minutos, tendríamos todas las coordenadas del exoesqueleto. Tomando los valores numéricos de las coordenadas x, y en el bloc de notas, podemos modelar el exoesqueleto en AutoCAD™ y analizar el crecimiento y cambios morfológicos que tanto insectos como moluscos han sufrido debido al cambio climático.

CONCLUSIONES

El deterioro de los insectos y moluscos que se ven afectados por el cambio climático cada día es más rápido. Entonces, en esta investigación, consideramos la importancia del tiempo para encontrar los valores numéricos de los puntos de referencia de las curvas del exoesqueleto.

En la prueba que realizamos, el resultado indica que en aproximadamente 15 minutos estarían disponibles todos los valores numéricos de las coordenadas x, y del exoesqueleto para que los biólogos puedan describir, comprender y predecir los cambios que ocurren en la naturaleza. Y a partir de la geometría descriptiva, se puede modelar la estructura de insectos y moluscos para su estudio.

Seguiremos trabajando en el pantógrafo tridimensional hasta que pueda ser una máquina de control numérico.

Agradecimientos Quiero expresar mi agradecimiento a Alfredo Almaraz por su apoyo en la realización de la impresión 3D del modelo para esta investigación.

REFERENCIAS

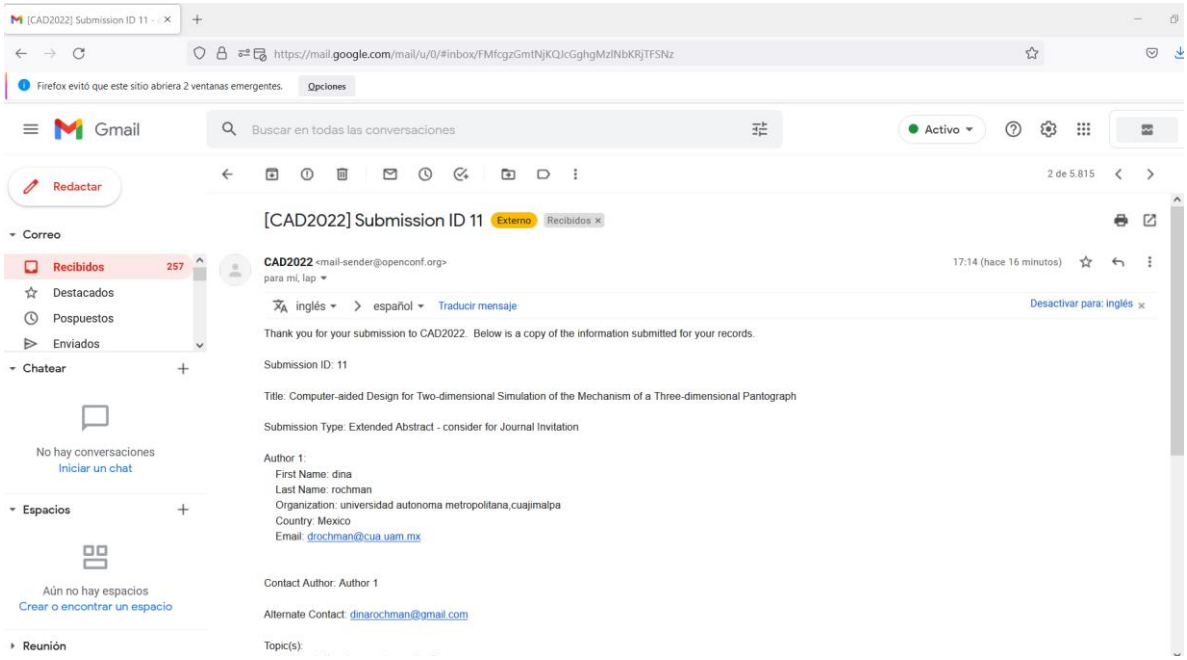
[1] Rohlf, F. J.: Geometric morphometrics simplified: Trends In Ecology and Evolution, USA, DOI:10.1016/j.tree.2004.08.005

[2] Searcy, R.; Lugo, J.; Beltrán, C.: Periodicity of internal growth ring deposition in the Pismo clam (Tivela stultorum) from Playa San Ramon, BC, Mexico, Ciencias marinas, México, 1989. http://dx.doi.org/10.7773/cm.v15i3.655

[3] Schreck, K.: Monge´s Legacy of descriptive and differential geometry, Docent Press, Boston, Massachusetts, USA, 2016Eaton, J. A.: Layered Manufacturing Methods for Reconstructing Bone Structures, Ph.D. Thesis, University of Minnesota, Twin Cities, MN, 1998.

[4] Pantograph,https://www.alamy.es/foto-pantografo-por-christoph-scheiner-143472979.html, Figure 1.

[5] R&I Industry Scrapbook, https://www.engraversjournal.com/legacyarticles/2207/, Part2: The Pantograph Era.



SE ANEXA EL ESCRITO DEL ABSTRACT QUE SE ENVIÓ PARA EL CONGRESO CAD´22.

SE ANEXA EL ESCRITO DELPAPER QUE SE ENVIARÁ AL CONGRESO CAD´22 CUANDO SE ME SOLICITE.

www.cadanda.com
Computer-Aided Design AND APPLICATIONS™

Computer-aided design for two-dimensional simulation of the mechanism of a three-dimensional pantograph

Dina Rochman

Universidad Autónoma Metropolitana, Cuajimalpa, drochman@cua.uam.mx

Corresponding author: Dina Rochman, drochman@cua.uam.mx

Abstract. This document describes the two-dimensional simulation of a three-dimensional pantograph. We present the graphical user and the programming of the program that we carry out. In the test we perform, the result indicates that the simulation takes 1.5 minutes to find the numerical values of the reference points of an exoskeleton curve. Therefore, in approximately 15 minutes, all the numerical values of the x and y coordinates of the exoskeleton would be available so that biologists can describe, understand, and predict the changes that occur in nature. And from the descriptive geometry, the structure of insects and mollusks can be modeled for study.

Keywords: Three-dimensional Pantograph. Two-dimensional simulation. Numerical value of the x and y coordinates.
DOI: https://doi.org/10.14733/cadaps.2023.aaa-bbb

1 INTRODUCTION

Insects and mollusks are part of the culture and ecosystem around the world. Biologists to study them use the method of geometric morphometry [1] or the mark-break [2] method to analyze their growth and the morphological changes that both insects and mollusks have undergone due to climate change.

Using these two methods involves a lot of time and effort for biologists because the work is initially done manually by marking the landmarks. They then use computer programs for their analysis, description, and evaluation.

The analysis of these two methods led us to think that the curves of the parts of small-scale animals can be drawn in another way without using the numerical values found from geometric morphometry, following the fundamentals of descriptive geometry of the projections of the points in space in the orthogonal projection.

Therefore, we propose a paradigm shift to find the numerical values of the reference points, that is, the x and y coordinates and the curves of the insect and mollusk parts using a three-dimensional pantograph.

The specific objective of the research project called "Geometry in Motion 3" is to develop and carry out a two-dimensional simulation of the mechanism of a three-dimensional pantograph.

The three-dimensional pantograph that we developed will be used to reproduce on a sheet of paper the reference points of each of the curves of the parts of insects and mollusks.

And in the two-dimensional simulation of the pantograph mechanism, the numerical values of the reference points are found, that is, the x and y coordinates.

From the numerical values, biologists will be able to describe, understand and predict the changes that occur in nature. And from descriptive geometry, the structure of insects and mollusks can be modeled for study.

This article has the following sections: Section 2, talks about the pantograph, Section 3, describes the methodology we followed. Section 4 explain the graphical user interface. Section 5 explains program programming. Section 6 list the results and, finally Section 7 presents the conclusions. All the figures in this document are original and created by the author that works at the Universidad Autónoma Metropolitana Campus Cuajimalpa in Mexico City

2 THREE-DIMENSIONAL PANTOGRAPH

For the development of the three-dimensional pantograph, we relied on the invention of Christoph Scheiner. Christoph Scheiner, a German mathematician, that was born on July 25, 1573, in the town of Wald, near Mindelheim, in the Swabia region, in southwestern Germany, invented in 1603 the pantograph, which allows enlarging the copying of images.

Geometrically, Scheiner's pantograph is based on the property of parallelograms. It is made up of four rods connected in such a way that they can be moved relative to a point or a pivot (Figure 1).

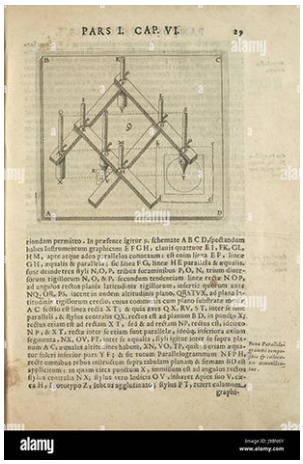


Figure 1. Christoph Scheiner pantograph [4].

To draw, the pivot is fixed, and the reference pointer is moved over the original drawing; a pen at the point of copying reproduces the image on a larger scale, which is determined by the ratio of distances P-RP and P-CP. P is the pivot; RP is the reference point and CP is the copy point. Changing the reference pointer to the copy point reproduces the image at a smaller scale.

Geometrically our three-dimensional pantograph is based on Gaspard Monge's method of representation [5]. Having the following parts: (1) a table with wheels that moves back and forth on the z-axis through a band; (2) a rectangular base on a rail that moves right and left on the x-

axis through an Acme threaded rod. On this base are the pointer and the tip of the pencil. And (3) two threaded Acme rods to move the rail up and down on the y-axis (Figure 2).

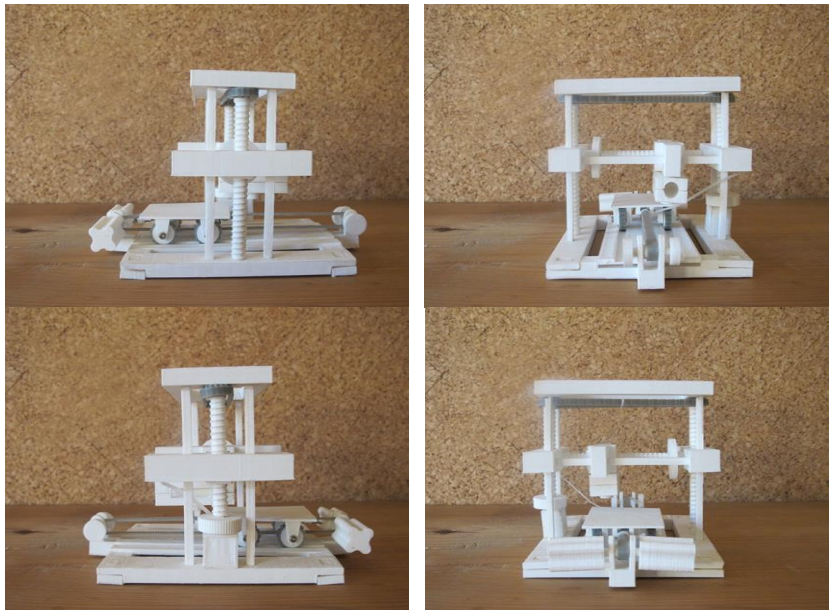


Figure 2. Photographs of the three-dimensional pantograph prototype.

3 METHODOLOGY

The methodology used to perform the two-dimensional simulation of the three-dimensional pantograph mechanism consists of three parts. The first part was defined the views, axes, and programming language (Figure 3).

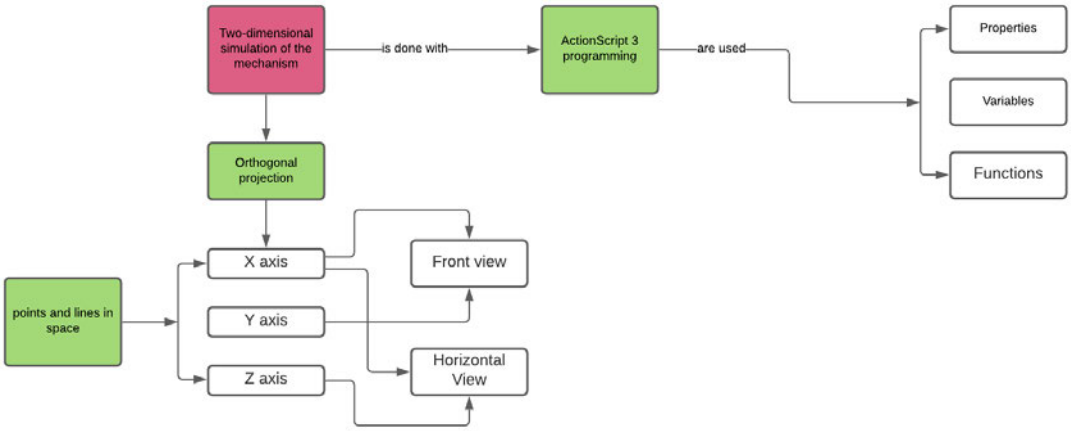


Figure 3. First part of the methodology.

In the second part, the graphic part of the screen was defined (Figure 4).

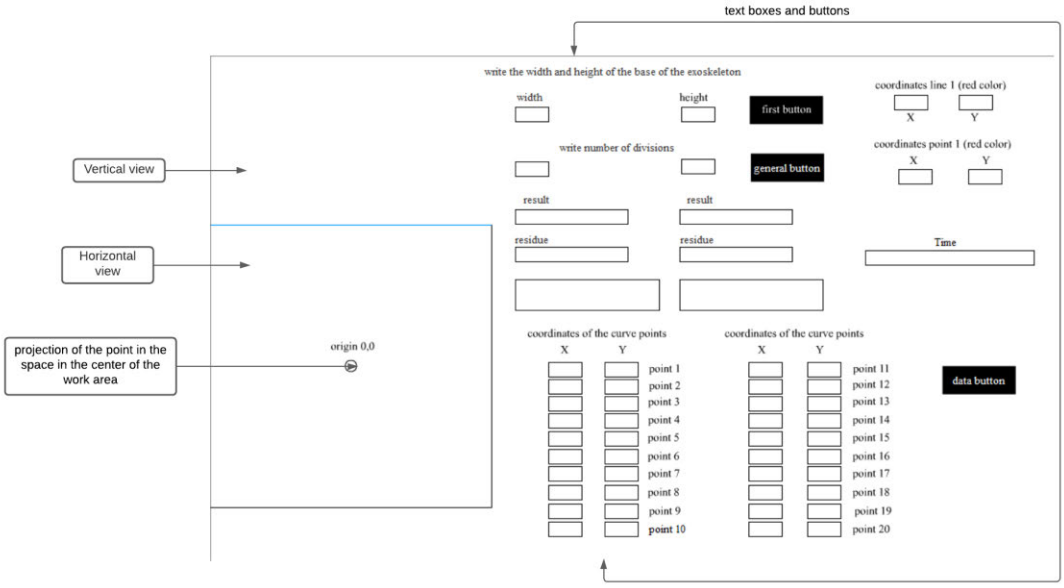


Figure 4. Second part of the methodology.

And in the third part, the algorithm was defined (Figure 5).

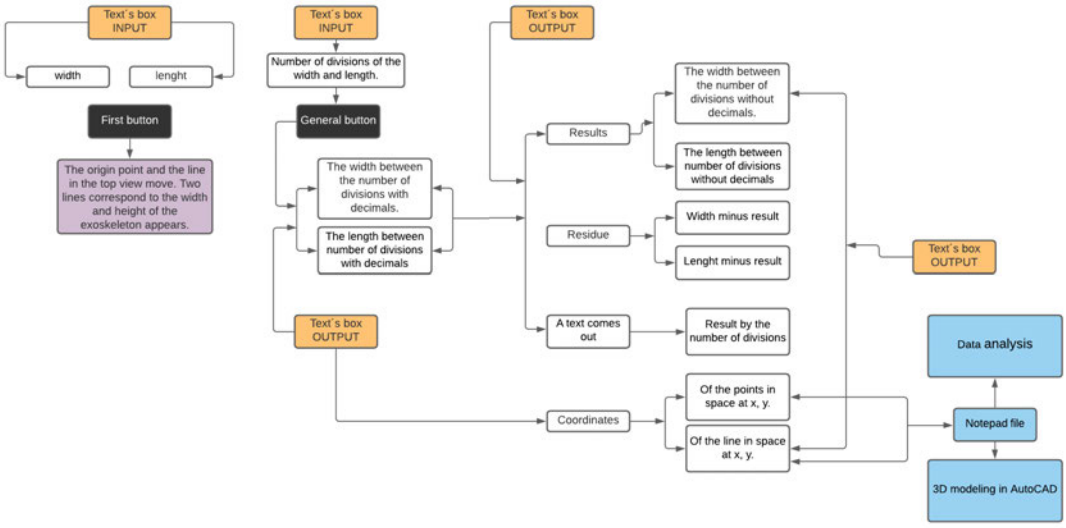


Figure 5. Third part of the methodology.

4 GRAPHICAL USER INTERFACE

In this section, we explain how to find the numerical values of the reference points of an exoskeleton curve. The values to be written are an example of how programming and simulation work.

(1) We write the width and length of the base of the exoskeleton, 300 units wide and 200 units long.
(2) The red line and the red point are in the coordinates 0,0. When is pressed the "First button", they change their position to the following coordinates: point x = 100, y = 650, and line x = 100, y = 300. The numerical values can see in the corresponding boxes. Two black lines appear in the horizontal view of the orthogonal projection, indicating the width and length of the exoskeleton.
(3) We write the number of divisions, 12 in the width and 6 in the length.
(4) When the "General button" is pressed, in each of the boxes the result and residue of each of the axes are seen. The distance value divisions by number of divisions also are seen. Time begins to run (Figure 6).

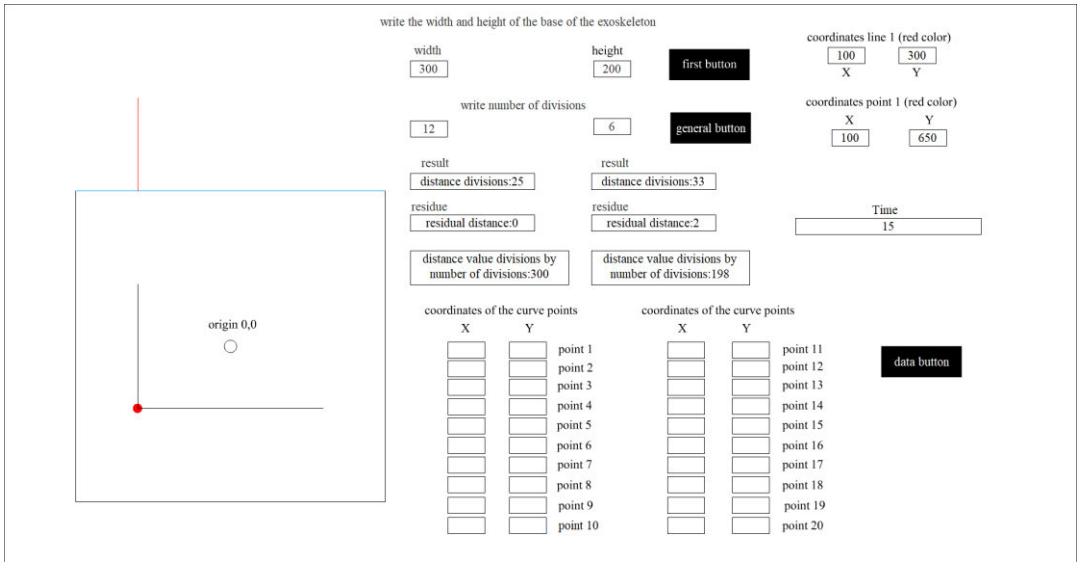


Figure 6. Results and residues.

(5) We press the "General button" again the same number of divisions, 12 times for the width and 6 times for the length. The red line and point move to the coordinate corresponding to each division. At the same time, it can see the gridlines in the horizontal view of the orthogonal projection. The exoskeleton curve appears with the points indicating the 12 divisions in the vertical view of the orthogonal projection.
(6) The simulation begins. The point and the red line simulate the pointer. The red line moves from bottom to top, the red point moves to the right as many times as the number of divisions. Each time the pointer touches a point on the exoskeleton curve, it simulates the points marked on the paper.
(7) Once the animation ends, the numerical values of the coordinates x and y of each of the points appear in the boxes of the coordinates of the curve point (Figure 7).

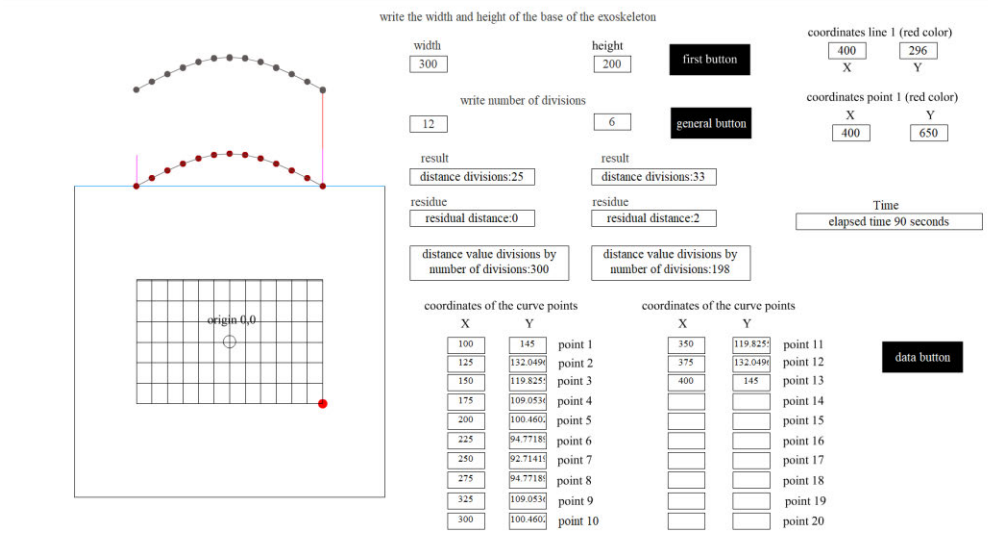


Figure 7. Numerical values of the coordinates x and y of each of the points.

(8) By pressing the "Data button" the numerical values of the coordinates x and y of each of the points are automatically saved in the notepad. Two values are left after the decimal point (Figure 8).

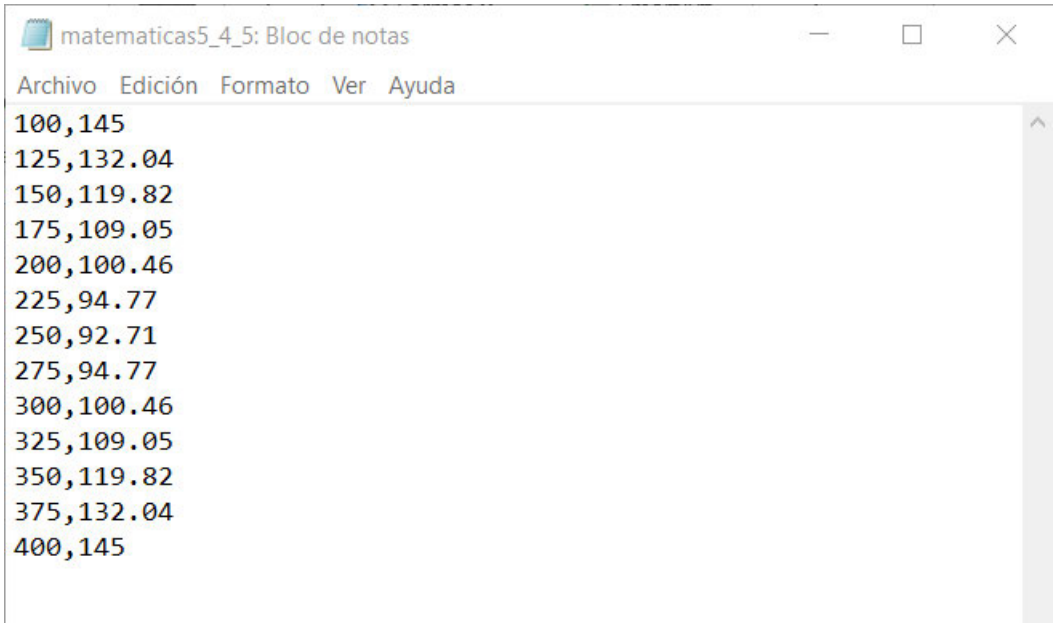


Figure 8. Numerical values of the coordinates x and y of each of the points

(9) The numerical values of the coordinates x and y are copied and passed to the AutoCAD™ to draw the curve. The points on the curve are backward because the origin of the Adobe

Animate 2022™ program is at the top, and in AutoCAD™, it is at the bottom. So, the points are rotated 180° to position them correctly in the isometric projection. We draw the curve on the first line of the grid (Figure 9).

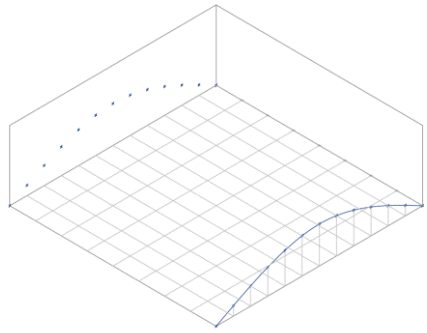


Figure 9. Outline of the first curve of the exoskeleton.

5 COMPUTER PROGRAM

It is worth mentioning that in the simulation presented, the geometric outline of the exoskeleton curve is the one found in previous works.

In programming, the width, length, and height distances of the exoskeleton curves are in units. We write the variable names in Spanish.

First, it was necessary to make a diagram of the curve that goes from point 0 to point 300 to find the distances of the height of the curve at every 25 units (Figure 11).

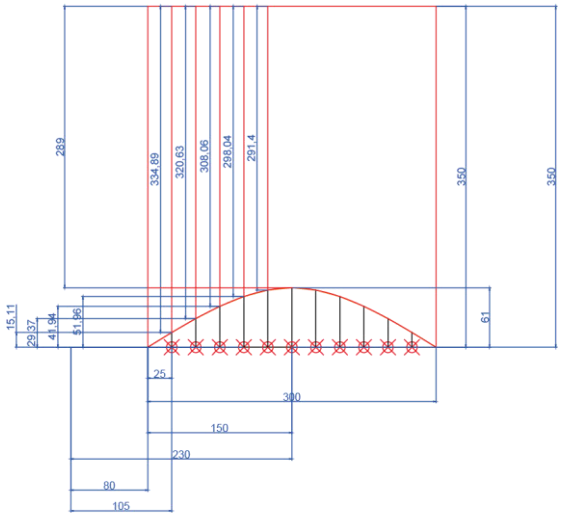


Figure 11. Distances of the height of the curve at every 25 units.

We find the percentage of the heights concerning the height of the center of the curve (Table 1).

Points	Distance	Percentage
1		0
2	334.8912	95.68
3	320.6296	91.61
4	308.0625	88.02
5	298.0370	85.15
6	291.4005	83.26
7	289.0000	82.57
8	291.4005	83.26
9	298.0370	85.15
10	308.0625	88.02
11	320.296	91.61
12	334.8912	95.68
13	350.0000	100.00

Table 1. Percentage of the heights.

The input data is the width and length of the exoskeleton and the number of parts it is divided into vertically and horizontally to create a grid.

```
inputAncho = Number(anchoEntrada.text);
inputLargo = Number(largoEntrada.text);
inpuAnchotDivisiones = Number(anchoDivisiones.text);
inputLargoDivisiones = Number(largoDivisiones.text);
```

The program performs the mathematical operations of the result of the distance of the divisions, the remainder and the distance value divisions by number of divisions. The remainder appears when the result has decimal numbers.

```
resultadoAncho.text = String(Number(anchoEntrada.text) / Number(anchoDivisiones.text));
var numAncho: Number = (Number(resultadoAncho.text));
var valorAncho: Number = Number(int(numAncho));
resultadoAnchoSinDecimales.text = ("distance divisions:" + valorAncho);

residuoAncho.text = String(Number(anchoEntrada.text) % Number(anchoDivisiones.text));
var numResiduoAncho: Number = (Number(residuoAncho.text));
var valorResiduoAncho: Number = Number(int(numResiduoAncho));
resultadoResiduoAnchoSinDecimales.text = ("residual distance:" + valorResiduoAncho);

multiplicacionAnchoDivisiones.text = String(valorAncho * valorDivisionesAncho);
var
numMultiplicacionAnchoDivisionesAncho:Number=(Number(multiplicacionAnchoDivisiones.text));
var
valorMultiplicacionAnchoDivisionesAncho:Number=Number(int(numMultiplicacionAnchoDivisionesA
ncho));
```

```
resultadoMultiplicacionAnchoDivisiones.text = ("distance value divisions by number of divisions:" +
valorMultiplicacionAnchoDivisionesAncho);
```

In the animation, the red line goes up and down. Every time the red line touches the curve of the exoskeleton, a dot appears that simulates the trace on the paper. An Array is used in programming to show all the points that simulate the trace on the paper.

```
var circulos: Array = new Array(circulo1, circulo2, circulo3, circulo4, circulo5, circulo6, circulo7,
circulo8, circulo9, circulo10, circulo11, circulo12, circulo13);

for each(var circulo in circulos) {
    if (nuevaLinea1.hitTestObject(circulo)) {
        nuevaLinea1.y == circulo.y;
        var circuloVerde: Sprite = new Sprite();
        circuloVerde.graphics.beginFill(0x605858);
        circuloVerde.graphics.drawCircle(nuevaLinea1.x, (nuevaLinea1.y - 150), 5);
        circuloVerde.graphics.endFill();
        addChild(circuloVerde);
        removeEventListener(Event.ENTER_FRAME, bajarLineaY);
```

The Timer indicates the time it will take for the three-dimensional pantograph to complete the journey.

```
var tiempo = 0;
        var timer: Timer = new Timer(1000, 200);
        timer.addEventListener(TimerEvent.TIMER, contarTiempo);
        timer.start();
        function contarTiempo(e: TimerEvent) {
                tiempo++;
                cajaTiempo.text = tiempo.toString();
```

We finish the programming with the file reference. The numerical values of the x, and y coordinates of each point of the exoskeleton curve are in the notepad.

```
        fileRef = new FileReference;
        fileRef.save(textToSave, "matematicas5_4_5");
```

6 RESULTS

Pantographs, which had been around in one form or another for hundreds of years, were about to shake up the world of engraving, which up to that time had been dominated by a small but fiercely independent group of hand engravers [5].

For centuries, the pantograph had been mostly used for two-dimensional or “flat” purposes, meaning the tracer and cutter traveled in a basically flat, two-dimensional plane. But the European machines changed all that. Known as “heavy-duty” industrial engraving machines, they were literally heavy, weighing over 1000 lbs. The industrial machines were used mostly for large-scale applications, such as making molds for vulcanized tires, shoes, silverware, belt buckles, dials, cans, extrusion and coining dies [5].

The three-dimensional pantograph exposed is for the study of insects and mollusks. Geometrically, we use Gaspard Monge's method of representation in its design (Figure 10).

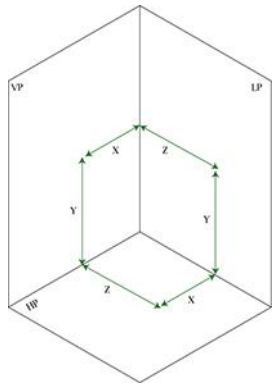


Figure 10. Gaspard Monge's method of representation.

In its solution, we consider the synthesis and simulation stages.

In the synthesis stage, we focus on the functionality of the mechanism, which until now is manual. The table with wheels moves back and forth on the z-axis through a band. The rectangular base with the pointer and pencil tip is on a rail moves left and right on the x-axis using an Acme threaded rod, and the two threaded Acme rods move the rail up and down on the y-axis (Figure 11).

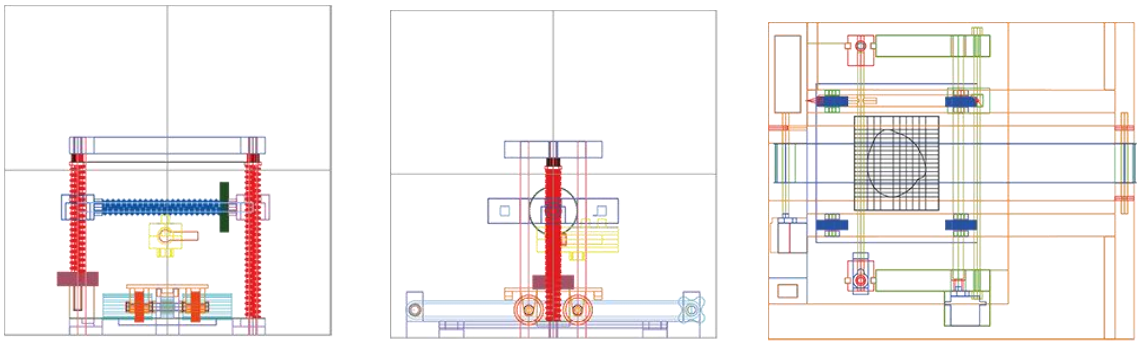


Figure 11. Side, front and top view of the model.

In the simulation stage, we focus on finding the numerical values of the x and y coordinates of the reference points of the exoskeleton curves using a computer program we designed.

In the Adobe Animate 2022™ program, both animation and programming were performed, using the object-oriented ActionScript 3.0 language to find the points of the exoskeleton curve.

The time it takes for the program to find the numerical values of the x and y coordinates of the first curve of the exoskeleton is 90 seconds, 1.5 minutes. So, in 1200 seconds, that is, 15 minutes, we would have all the coordinates of the exoskeleton.

Taking the numerical values of the coordinates x and y in the notepad, we can model the exoskeleton in AutoCAD™ and analyze the growth and morphological changes that both insects and mollusks have suffered due to climate change.

7 CONCLUSIONS

The deterioration of insects and mollusks that are affected by climate change every day is faster. So, in this research, we consider the importance of time to find the numerical values of the reference points of the exoskeleton curves.

In the test we perform, the result indicates that in approximately 15 minutes, all the numerical values of the x and y coordinates of the exoskeleton would be available so that biologists can describe, understand, and predict the changes that occur in nature. And from the descriptive geometry, the structure of insects and mollusks can be modeled for study.

We will keep working on the three-dimensional pantograph until it can be a numerical control machine.

Acknowledgements
I want to express my gratitude to Alfredo Almaraz for his support in the realization of the 3D printed of the model for this research.

Dina Rochman, http://orcid.org/0000-0001-8902-3513

REFERENCES

[1] Rohlf, F. J.: Geometric morphometrics simplified: Trends In Ecology and Evolution, USA, DOI:10.1016/j.tree.2004.08.005
[2] Searcy, R.; Lugo, J.; Beltrán, C.: Periodicity of internal growth ring deposition in the Pismo clam (Tivela stultorum) from Playa San Ramon, BC, Mexico, Ciencias marinas, México, 1989. http://dx.doi.org/10.7773/cm.v15i3.655
[3] Schreck, K.: Monge´s Legacy of descriptive and differential geometry, Docent Press, Boston, Massachusetts, USA, 2016Eaton, J. A.: Layered Manufacturing Methods for Reconstructing Bone Structures, Ph.D. Thesis, University of Minnesota, Twin Cities, MN, 1998.
[4] Pantograph,https://www.alamy.es/foto-pantografo-por-christoph-scheiner-143472979.html, Figure 1.
[5] R&I Industry Scrapbook, https://www.engraversjournal.com/legacyarticles/2207/, Part2: The Pantograph Era.

Title:
Computer-aided Design for Two-dimensional Simulation of the Mechanism of a Three-dimensional Pantograph

Authors:
Dina Rochman, drochman@cua.uam.mx, Universidad Autónoma Metropolitana, Cuajimalpa

Keywords:
Three-dimensional Pantograph. Two-dimensional simulation. Numerical value of the x and y coordinates.

DOI: 10.14733/cadconfP.2022.xxx-yyy

Introduction:
Insects and mollusk are part of the culture and ecosystem around the world. Biologists to study them use the method of geometric morphometry [1] or the mark-break [2] method to analyze their growth and the morphological changes that both insects and mollusks have undergone due to climate change.

Using these two methods involves a lot of time and effort for biologists because the work is initially done manually by marking the landmarks. They then use computer programs for their analysis, description, and evaluation.

Therefore, we propose a paradigm shift to find the numerical values of the reference points, that is, the x and y coordinates and the curves of the insect and mollusk parts using a three-dimensional pantograph.

The three-dimensional pantograph that we developed will be used to reproduce on a sheet of paper the reference points of each of the curves of the parts of insects and mollusks.

And in the two-dimensional simulation of the pantograph mechanism, the numerical values of the reference points are found, that is, the x, y coordinates.

Main Sections:
Geometrically our three-dimensional pantograph is based on Gaspard Monge's method of representation [3]. Having the following parts: (1) a table with wheels that moves back and forth on the z-axis through a band; (2) a rectangular base on a rail that moves right and left on the x-axis through an Acme threaded rod. On this base are the pointer and the tip of the pencil. And (3) two threaded Acme rods to move the rail up and down on the y-axis (Fig 1).

Fig. 1: Photographs of the three-dimensional pantograph prototype.

In the synthesis stage, we focus on the functionality of the mechanism, which until now is manual. (Fig 2).

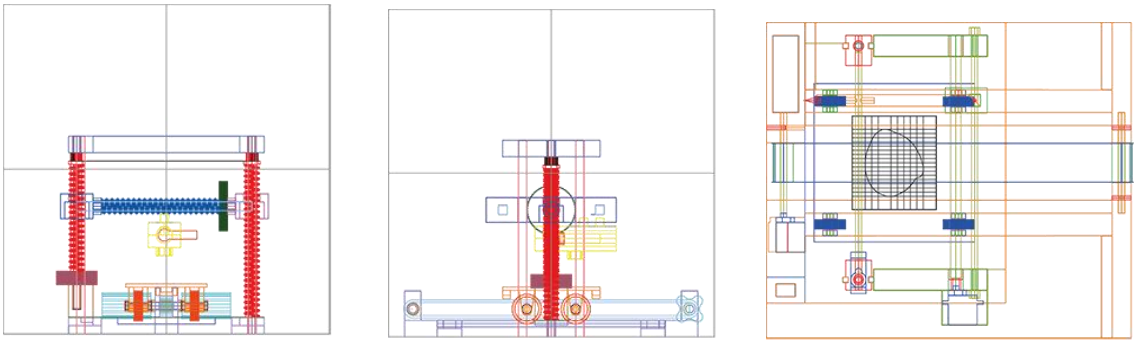


Fig. 2: Side, front and top view of the model.

In the simulation stage, we focus on finding the numerical values of the x and y coordinates of the reference points of the exoskeleton curves using a computer program we designed.

In the Adobe Animate 2022™ program, both animation and programming were performed, using the object-oriented ActionScript 3.0 language to find the points of the exoskeleton curve.

Methodology

The methodology used to perform the two-dimensional simulation of the three-dimensional pantograph mechanism consists of three parts. The first part was defined the views, axes, and programming language. In the second part, the graphic user interface was defined (Fig 3).

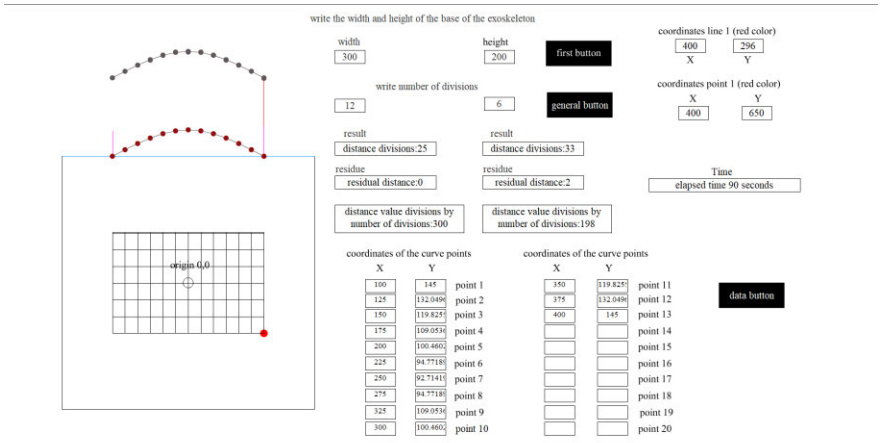


Fig. 3: Graphical user iterface.

In the graphical user interface are a series of boxes where the user enters the width, length, and the number of divisions into which they will split the exoskeleton vertically and horizontally. There are three buttons. Pressing the "first button" the point and the red line change position. Pressing the "general button" performs the mathematical operations to find the numerical values of the reference

points and the animation begins. Pressing the "data button" saves the numerical values of the x and y coordinates of the reference points in the notepad.

We made an animation to recreate the two-dimensional simulation of the three-dimensional pantograph. In the two-dimensional simulation, the red line moves from bottom to top; the point moves to the right many times as the number of divisions. Each time the pointer touches a point on the exoskeleton curve, it simulates the points marked on the paper.

Once the animation ends, the numerical values of the coordinates x and y of each of the points appear in the boxes of the coordinates of the curve point.

The numerical values of the coordinates x and y which are in the notepad are copied and passed to the AutoCAD™. We draw the curve on the first line of the grid (Fig 4).

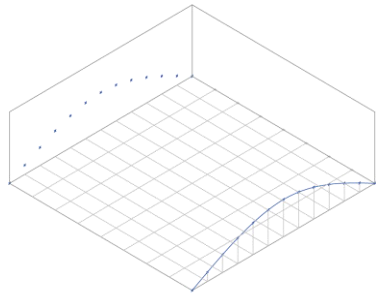


Fig. 4: Outline of the first curve of the exoskeleton.

And in the third part, the algorithm was defined (Fig 5).

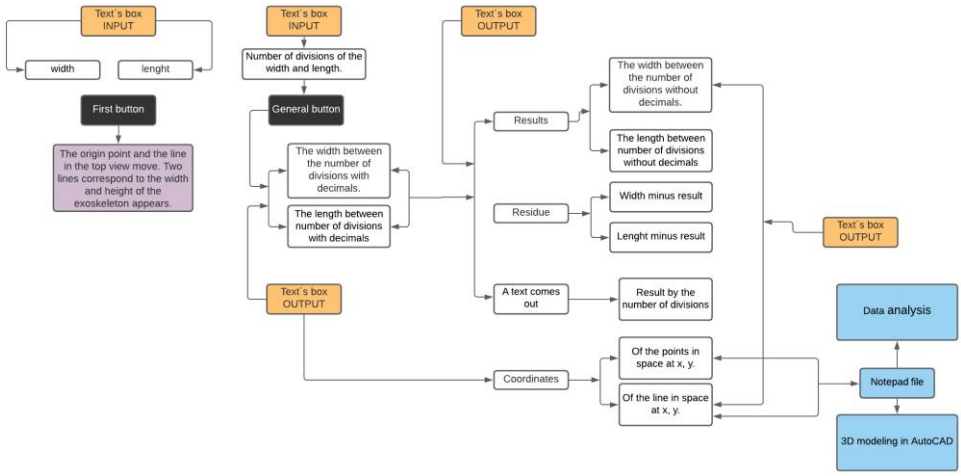


Fig. 5: Algorithm.

It is worth mentioning that in the simulation presented, the geometric outline of the exoskeleton curve is the one found in previous works.

In programming, the width, length, and height distances of the exoskeleton curves are in units. We write the variable names in Spanish.

In the computer program, the input data is the width and length of the exoskeleton and the number of parts it is divided into vertically and horizontally to create a grid.

```
inputAncho = Number(anchoEntrada.text);
inputLargo = Number(largoEntrada.text);
inpuAnchotDivisiones = Number(anchoDivisiones.text);
inputLargoDivisiones = Number(largoDivisiones.text);
```

The program performs the mathematical operations of the result of the distance of the divisions, the remainder and the distance value divisions by number of divisions. The remainder appears when the result has decimal numbers.

```
resultadoAncho.text = String(Number(anchoEntrada.text) / Number(anchoDivisiones.text));
var numAncho: Number = (Number(resultadoAncho.text));
var valorAncho: Number = Number(int(numAncho));
resultadoAnchoSinDecimales.text = ("distance divisions:" + valorAncho);
```

In the animation, the red line goes up and down. Every time the red line touches the curve of the exoskeleton, a dot appears that simulates the trace on the paper. An Array is used in programming to show all the points that simulate the trace on the paper.

```
var circulos: Array = new Array(circulo1, circulo2, circulo3, circulo4, circulo5, circulo6, circulo7, circulo8, circulo9, circulo10, circulo11, circulo12, circulo13);

for each(var circulo in circulos) {
    if (nuevaLinea1.hitTestObject(circulo)) {
        nuevaLinea1.y == circulo.y;
        var circuloVerde: Sprite = new Sprite();
        circuloVerde.graphics.beginFill(0x605858);
        circuloVerde.graphics.drawCircle(nuevaLinea1.x, (nuevaLinea1.y - 150), 5);
        circuloVerde.graphics.endFill();
        addChild(circuloVerde);
        removeEventListener(Event.ENTER_FRAME, bajarLineaY);
```

The Timer indicates the time it will take for the three-dimensional pantograph to complete the journey.

```
var tiempo = 0;
        var timer: Timer = new Timer(1000, 200);
        timer.addEventListener(TimerEvent.TIMER, contarTiempo);
        timer.start();
        function contarTiempo(e: TimerEvent) {
            tiempo++;
            cajaTiempo.text = tiempo.toString();
```

We finish the programming with the file reference. The numerical values of the x, and y coordinates of each point of the exoskeleton curve are in the notepad.

```
    fileRef = new FileReference;
      fileRef.save(textToSave, "matematicas5_4_5");
```

Conclusions:

The deterioration of insects and mollusks that are affected by climate change every day is faster. So, in this research, we consider the importance of time to find the numerical values of the reference points of the exoskeleton curves.

In the test we perform, the result indicates that in approximately 15 minutes, all the numerical values of the x and y coordinates of the exoskeleton would be available so that biologists can describe, understand, and predict the changes that occur in nature. And from the descriptive geometry, the structure of insects and mollusks can be modeled for study.

We will keep working on the three-dimensional pantograph until it can be a numerical control machine.

Acknowledgements

I want to express my gratitude to Alfredo Almaraz for his support in the realization of the 3D printed of the model for this research.

References:

[1] Rohlf, F. J.: Geometric morphometrics simplified: Trends In Ecology and Evolution, USA, DOI:10.1016/j.tree.2004.08.005

[2] Searcy, R.; Lugo, J.; Beltrán, C.: Periodicity of internal growth ring deposition in the Pismo clam (Tivela stultorum) from Playa San Ramon, BC, Mexico, Ciencias marinas, México, 1989. http://dx.doi.org/10.7773/cm.v15i3.655

[3] Schreck, K.: Monge´s Legacy of descriptive and differential geometry, Docent Press, Boston, Massachusetts, USA, 2016Eaton, J. A.: Layered Manufacturing Methods for Reconstructing.

Dina Rochman, http://orcid.org/0000-0001-8902-351