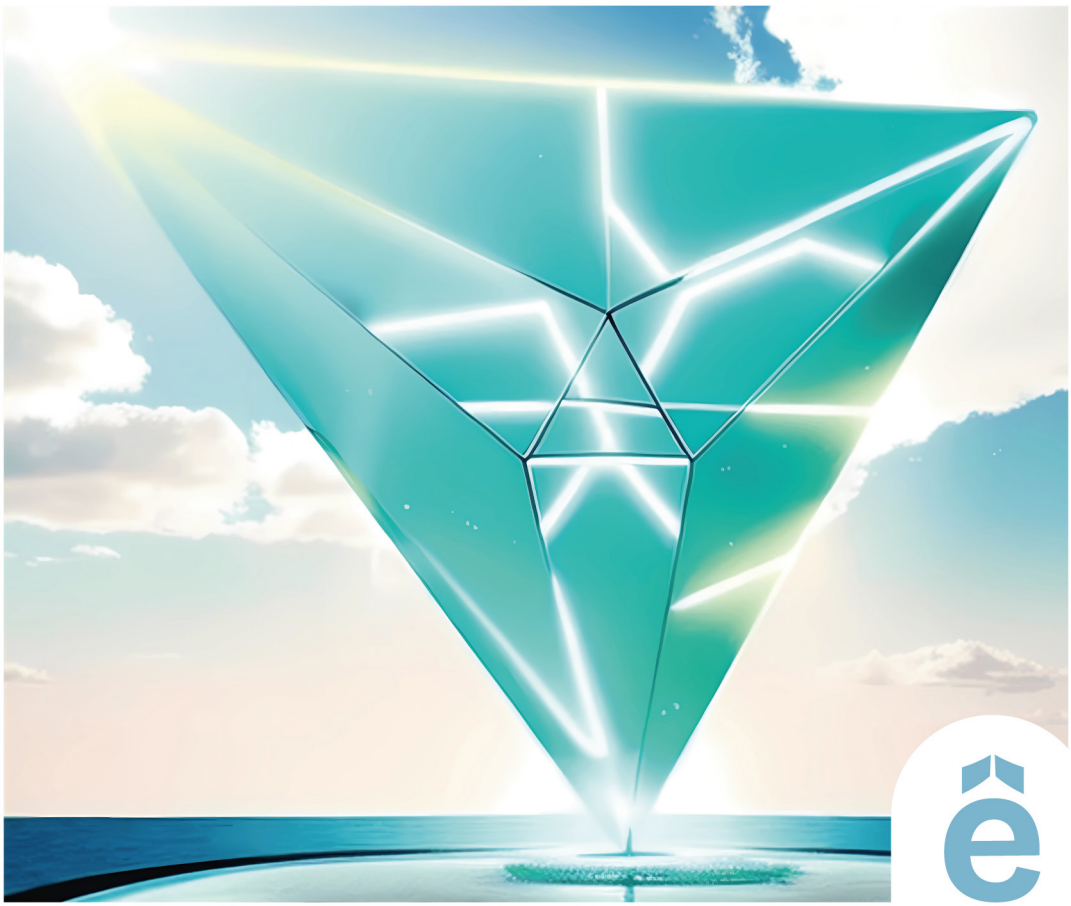
ASPECTOS TEÓRICOS Y PRÁCTICOS DEL TEOREMA DE MORLEY

Edwin Giovanni Insuasty Portilla
Saulo Mosquera López



Editorial
Universidad de Nariño

êditorial
Universidad de **Nariño**

Aspectos teóricos y prácticos del Teorema de Morley

Aspectos teóricos y prácticos del Teorema de Morley

Edwin Giovanni Insuasty Portilla
Saulo Mosquera López

êditorial
Universidad de Nariño

Insuasty Portilla, Edwin Giovanni
Aspectos teóricos y prácticos del teorema de Morley / Edwin Giovanni Insuasty Portilla, Saulo Mosquera López—1ª. ed. – San Juan de Pasto : Editorial Universidad de Nariño, 2025

115 páginas : ilustraciones, gráficos, tablas

Incluye referencias bibliográficas p. 108-109 y reseña de los autores p. 113
ISBN: 978-628-7771-33-8 Digital

1. Teorema de Morley 2. Morley, Frank—1860-1937 3. Triangulo de Morley--Origami 4. Triangulo de Morley—Software. I. Mosquera López, Saulo

516.154 I599a – SCDD-Ed. 22

SECCIÓN DE BIBLIOTECA

Aspectos teóricos y prácticos del Teorema de Morley

ISBN (digital): 978-628-7771-33-8
Primera edición

Corrección de estilo: Manuel Enrique Martínez Riascos
Diseño de cubiertas y diagramación: Liseth Motta Realpe

Fecha de publicación: Abril 2025
San Juan de Pasto - Nariño - Colombia

CONTENIDO

PRESENTACIÓN

El **Teorema de Morley** es uno de esos temas ocultos de la geometría, descubierto a fines del siglo XIX, que revela un hecho inesperado: al trisecar los ángulos de cualquier triángulo, los puntos de intersección de las trisecciones interiores forman un **triángulo equilátero**. Esta sorprendente conexión entre la trisección de ángulos y la simetría del triángulo equilátero ha fascinado a matemáticos durante décadas. Sin embargo, más allá de la demostración formal, el Teorema de Morley invita a explorar cómo la matemática puede combinarse con la creatividad para construir objetos tangibles y experiencias interactivas.

Este libro tiene como objetivo brindar un recorrido integral por el Teorema de Morley, abarcando no solo su demostración rigurosa, sino también aplicaciones innovadoras mediante técnicas de **origami computacional** y **desarrollo de software**. La intención es ofrecer a los lectores una experiencia multidisciplinaria que entrelace las matemáticas puras, el arte del origami y la programación, demostrando que la teoría abstracta puede convertirse en una experiencia lúdica y educativa.

En el **Capítulo 1**, se aborda el Teorema de Morley desde su contexto teórico, proporcionando una introducción clara y accesible. Se detallan tres demostraciones de este resultado, así: la demostración de **M. T. Naraniengar** de 1909, explicando los lemas previos necesarios para comprender los fundamentos del teorema, la de **A. Letac** de 1939 y la del profesor Colombiano **A. Chaves** de 1993. Esta sección se enfoca en

presentar los argumentos de manera gradual, facilitando que el lector entienda cada paso sin perderse en tecnicismos, al tiempo que destaca la elegancia matemática que caracteriza este resultado.

El **Capítulo 2** explora el Teorema de Morley desde una perspectiva más visual y experimental mediante la técnica del **origami matemático**. Aquí, el lector descubrirá cómo la precisión del plegado de papel puede ser una herramienta poderosa para representar conceptos geométricos complejos. Se introduce el concepto de **origami computacional**, una disciplina que combina algoritmos y arte para resolver problemas matemáticos de manera tangible. En esta sección se describe el **Método de Abe** para trisecar ángulos, un enfoque práctico para construir el Triángulo de Morley con papel. Además, se ofrece una guía detallada sobre la preparación del papel y la aplicación del **Axioma 6 de Huzita**, proporcionando instrucciones paso a paso para lograr cada pliegue de forma precisa.

El **Capítulo 3** se enfoca en llevar la experiencia del Teorema de Morley al ámbito de las aplicaciones digitales mediante la creación de un **rompecabezas interactivo** basado en este triángulo especial. Aquí, los lectores aprenderán sobre las diferencias entre **mapas de bits y gráficos vectoriales**, identificando las ventajas y desventajas de cada enfoque para el desarrollo de aplicaciones. Se explica cómo elegir entre herramientas como **Visual Studio con C#**, **Android Studio con Java** y **Adobe Animate con ActionScript 3** para diseñar y desarrollar un rompecabezas funcional y atractivo.

Además de los aspectos técnicos, este capítulo presenta una **metodología de desarrollo**, guiando al lector en cada paso del proceso de creación del rompecabezas. Desde la declaración de variables y objetos hasta las consideraciones técnicas para optimizar la experiencia del usuario, se ofrecen buenas prácticas que aseguran un desarrollo eficiente. Este capítulo también incluye instrucciones para descargar la aplicación y poner en marcha el proyecto, permitiendo a los lectores interactuar con el rompecabezas en diferentes plataformas.

Este libro no es solo una exploración teórica, sino una invitación a combinar disciplinas para crear nuevas experiencias. Al entrelazar matemáticas, origami y programación, se muestra cómo conceptos abstractos pueden convertirse en herramientas para la creatividad y el aprendizaje interactivo. Esta obra está dirigida a una amplia audiencia: matemáticos interesados en resultados sorprendentes, entusiastas del origami que buscan nuevos desafíos, y programadores que desean explorar proyectos educativos y recreativos.

En un mundo cada vez más interdisciplinario, esta propuesta busca demostrar que la matemática no solo es rigurosa, sino también **creativa, tangible y divertida**. Con la combinación de teoría, arte y tecnología, este libro invita a los lectores a disfrutar del camino desde la abstracción del Teorema de Morley hasta la satisfacción de construir con sus propias manos —y también con códigos— algo que ejemplifique la belleza de las matemáticas.

1. EL TEOREMA DE MORLEY

El teorema de Morley es un resultado geométrico que nos afirma que los puntos de intersección de las semirrectas adyacentes que dividen los ángulos de un triángulo en tres ángulos congruentes forman un triángulo equilátero. Aunque existe una amplia literatura al respecto, este resultado no es lo suficientemente conocido en los ámbitos de la enseñanza media o universitaria. El propósito de este capítulo es el presentar algunas cuestiones referentes a la evolución de este resultado, desde su formulación hasta la actualidad, así como también presentar tres demostraciones del mismo. La primera tiene carácter geométrico y se debe al matemático indio M. Y. Naraniengar, la segunda es de naturaleza trigonométrica y se atribuye al matemático francés A. Letac y, la tercera se debe al ingeniero civil colombiano A. Chaves.

1.1 INTRODUCCIÓN

Hacia el año 1897 el matemático británico Frank Morley, se radicó E.E.U.U. para trabajar como como profesor del Haverford College y en 1900 se trasladó a la universidad Johns Hopkins para ser el director del Departamento de Matemáticas de esta prestigiosa institución.

En ese año publicó, en el primer número de la revista *American Society Mathematical Translations* el artículo "*On the Metrics Geometry of the Plane n-lines*", en el cual se tratan, usando argumentos geométricos de la teoría

de funciones, resultados generales sobre el comportamiento de n - rectas en el plano y sus características constantes.

En su forma más simple el teorema de Morley, que se ilustra en la Figura 1, nos expresa que:

"Los puntos de intersección de las trisectrices interiores adyacentes de los ángulos de cualquier triángulo determinan un triángulo equilátero."

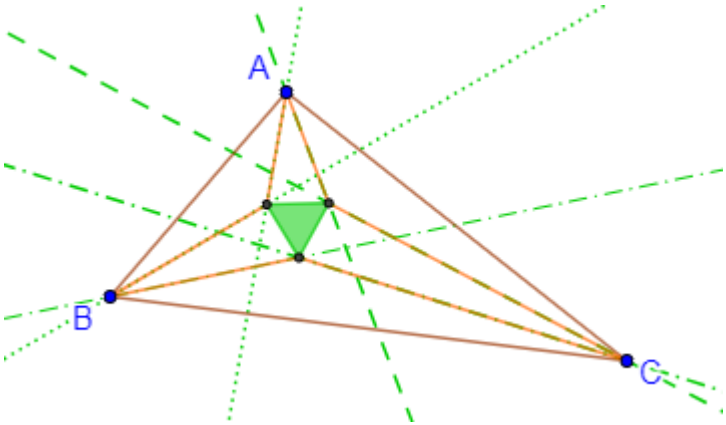


Figura 1 El primer triángulo de Morley. Fuente propia.

Aunque en su artículo Morley, no considera de manera explícita este resultado las referencias consultadas indican que Morley estaba consciente del mismo y de sus consecuencias, sin embargo, no se preocupó en su divulgación ya que para él solo era una pequeña parte de una teoría más general. Taylor y Marr (1913) afirman que este hecho fue comunicado por Morley a algunos de sus amigos entre los cuales es posible mencionar a Richmond en Cambridge y a Whittaker en Edimburgo, y que en 1904 este resultado ya se había hecho público.

El teorema de Morley se considera uno de los resultados más inesperados de las matemáticas elementales que extrañamente pasó desapercibido durante siglos, a pesar de que expresa una propiedad para los trisectores análoga a las bisectrices. ¿por qué los antiguos griegos, aunque estudiaron muchos resultados relacionados con el triángulo, no fueron capaces de descubrir este resultado? No está claro, parece que la imposibilidad de trisectar un ángulo con regla y compás les limitó en la percepción de este.

De acuerdo con Oakley y Baker (1978) los primeros enunciados formales de esta proposición se encuentran como el problema 1631 en el volumen 61-81 de la revista "The Educational times" de febrero de 1908 propuesto por E. J. Ebden y como el problema 1655, propuesto por T. Delahaye y H. Lez en el número 8 de la revista Mathesis del mismo año, sin hacer referencia a Morley en ambas publicaciones.

La primera solución a este problema fue desarrollada por M. Satyanarama y publicada como "Solution to problems 1655" en el volumen 61-308 de "The Educational times" de julio del mismo año. La segunda solución conocida fue desarrollada por el matemático indio, M. T. Naraniengar, y publicada en la revista Mathematical Questions and Solutions, en el The Educational Times de 1909, la cual presentaremos con detalle en la siguiente sección. Desde entonces se han desarrollado numerosas demostraciones y generalizaciones de este teorema que utilizan diferentes técnicas: geométricas, trigonométricas y algebraicas,

en particular, presentamos en las secciones 1.3 y 1.4 dos demostraciones de carácter trigonométrico. La primera se debe a A. Letac. Ver, por ejemplo, Bogomolny (s.f.) y la segunda, es un aporte del profesor de la Universidad Nacional (sede Manizales) A. Chaves, a la amplia literatura sobre este resultado.

La siguiente tabla nos presenta un listado de diferentes demostraciones de este teorema, algunas de las cuales utilizan sólo matemáticas elementales, pero pocas de ellas pueden considerarse que sean sencillas.

Tabla 1 Clase de demostración, autor y año. Fuente propia.

	Autor	**Tipo de demostración**	**Año**
1	M. Satyanarama	Trigonométrica	1909
2	M. T. Naraniengar	Sintética	1909
3	J. M. Child	Sintética	1922
4	R. Bricard	Sintética	1922
5	E. Borel	Sintética	1930
6	W. J. Dobbs	Sintética	1938
4	A. Letac	Trigonométrica	1939
5	T. W. Andrews	Sintética	1966
6	H. S. M. Coxeter	Sintética	1969
7	E. W. Dijkstra	Trigonométrica	1975
8	A. Chaves	Trigonométrica	1993
9	D. J. Newman	Trigonométrica	1996

10	A. Connes	Teoría de grupos	1998
11	Eric l. Grinberg y Mehmet Orhon	Trigonométrica	2003
12	B. Bollobas	Trigonométrica	2006
13	Jhon Conway	Sintética	2014
14	Mehmet Kilić	Sintética	2015
15	I. Gasteratos, S. Kuruklis, T. Kuruklis	Trigonométrica	2017
16	J. Arioni	Sintética	2017
17	P. Andrew	Geometría analítica compleja	2018
18	Stéphane Peigné	Sintética	2020
19	Man Yiu-Kwong	Sintética	2022
20	H. T. Quang	Sintética	2024

Es un hecho conocido que, si se consideran las trisectrices de los ángulos exteriores se obtiene un nuevo triángulo de Morley el cual se ilustra en la Figura 2.

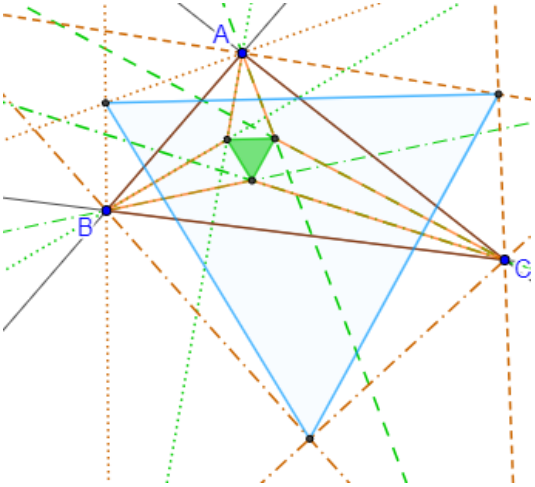


Figura 2 El segundo triángulo de Morley. Fuente propia.

Si ahora se consideran las intersecciones entre las trisectrices interiores y exteriores se obtienen tres nuevos triángulos de Morley los cuales se ilustran en la figura 3.

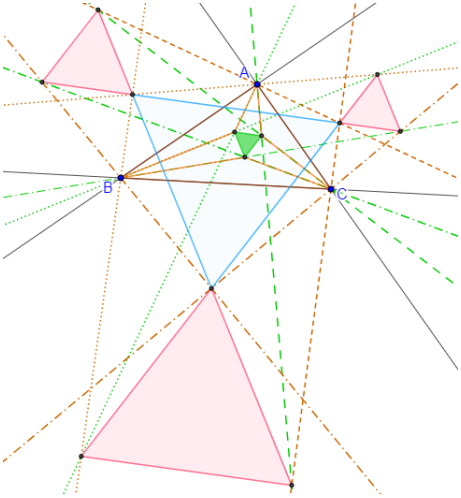


Figura 3 Cinco triángulos de Morley. Fuente propia.

¿Existen más triángulos de Morley? La respuesta es afirmativa y se encuentra en el artículo de Taylor y Marr (1913), quienes fueron los primeros en dar una solución completa al problema. Ellos demuestran que, si se considera el hecho de que cada ángulo A, B, C del triángulo $\measuredangle ABC$, se puede trisecar en tres ángulos distintos, de manera diferente usando A, A+2π, A+4π, y de manera análoga para los ángulos en B y C, se obtienen 27 triángulos equiláteros, de los cuales 18 son triángulos de Morley.

Los detalles de esta solución no son sencillos, utilizan conceptos no usuales de geometría tales como: coordenadas trilineales y conjugado isogonal y no se considerarán en este trabajo. Si Ud. desea información adicional al respecto puede consultar el artículo de Taylor y Marr (1913).

Todos los triángulos de Morley tienen lados paralelos y la longitud de un lado de cada uno es de la forma

$$8r\ sen\ (\alpha + \theta) sen(\gamma + \rho)\ sen(\beta + \varphi)$$

donde cada valor de $\theta, \rho,\ \varphi$ es algún arreglo particular de 0, $\frac{\pi}{6}$, $\frac{\pi}{3}$ y r es el radio de la circunferencia circunscrita al triángulo ABC. (Letac, 1938-1939).

Por último, conviene señalar que, el teorema de Morley está relacionado con muchos temas de geometría euclídea y con configuraciones

notables de punto-línea-plano-círculo-polígono con nombres como Apolonio, Brianchon, Ceva, Desargues, Feuerbach, Hesse, Lemoine, Menelao, Pascal, Ptolomeo, Simson, Spieker, Steiner, et., Ver, por ejemplo, (Ghiocas, 1934).

La última demostración conocida del teorema de Morley se debe a Quang Hung Tran de la Universidad de Vietnam y se encuentra en la revista Elem. Math. de abril de 2024. Ver Quang (2024).

1.2 LA DEMOSTRACIÓN GEOMÉTRICA DE M. T. NARANIENGAR.

En esta sección se desarrolla la demostración del teorema de Morley propuesta por M. T. Naraniengar de 1909. Esta prueba sigue los lineamientos de la versión que se encuentra en Coxeter y Greitzer (1967).

Esta demostración del teorema de Morley requiere del siguiente lema.

1.2.1 Lema: Si cuatro puntos Y', X, Y, X' satisfacen que

1. $Y'X = XY = YX'$
2. $\measuredangle YXY' = \measuredangle X'YX = 180° - 2\gamma > 60°$

entonces estos puntos son concíclicos. Adicionalmente, si C es un punto en el lado opuesto de la recta X'Y' en el que está el punto X tal que $\angle Y'CX' = 3\gamma$ entonces el punto C también está en la circunferencia w.

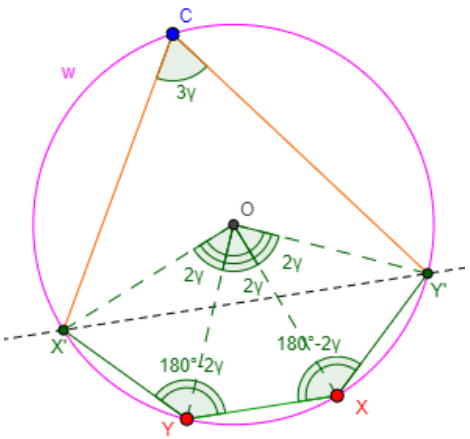


Figura 4 Ilustración para el Lema. Fuente propia.

Demostración: consideremos las bisectrices de los ángulos YXY' y X'YX que se cortan en O, entonces por A-L-A los triángulos OY'X, OXY, OYX' son congruentes e isósceles y ángulos de base de 90° - γ. Por tanto, OY'=OZ=OY = OX', es decir, Y', X, Y, X' están sobre una circunferencia de centro O y radio OY' y sus ángulos en el vértice O son todos iguales a 2γ.

En otras palabras, cada uno de los arcos iguales Y'X, XY, YX' subtiende un ángulo central 2γ y, en consecuencia, subtiende un ángulo 3γ en cualquier punto del arco Y'X' que no contenga X. Este arco es el lugar geométrico de los puntos (en el lado de la recta Y'X' donde no está X) desde los cuales la cuerda Y'X' subtiende un ángulo 3γ. Uno de esos puntos es C; por lo tanto, C se encuentra en la circunferencia.

1.2.2 EL TEOREMA DE MORLEY

Los puntos de corte de las trisectrices adyacentes de un triángulo cualquiera determinan un triángulo equilátero.

Demostración: a partir de un triángulo ABC la demostración de Naraniengar está conformada por tres grandes pasos así:

1. Construir las trisectrices de dos de sus ángulos, digamos A y B las cuales se cortan en los puntos U y Z.
2. Construir, de manera apropiada un triángulo XYZ y demostrar que este triángulo es equilátero.
3. Si se triseca el ángulo en C, demostrar que CX y CY son las trisectrices y que por tanto el triángulo XYZ es el triángulo de Morley. En esta parte debemos construir las condiciones para la aplicación del Lema tratado anteriormente-

Desarrollamos a continuación cada uno de estos pasos.

1. Considere un triángulo ABC y las trisectrices de los ángulos A = 3β y B = 3α las cuales se cortan en los puntos U y Z como en la gráfica.
2. Consideremos los puntos X en BU y, Y en AU de tal manera que

$$\measuredangle YZU = \measuredangle XZU = 30°$$

Cómo en el triángulo ABU, los ángulos A y B están divididos en dos ángulos congruentes por AZ y BZ entonces Z es el incentro

del triángulo y UZ es la bisectriz del ángulo AUB por lo que aplicando el teorema de congruencia ALA se tiene que los triángulos YZU y XZU son congruentes, por lo que ZY = ZX y el triángulo XYZ es equilátero.

Además, el triángulo UYX es isósceles y puesto que en el triángulo AUB, U= 180°- 2β-2α y en el triángulo UYX, $180° - 2\beta - 2\alpha = 180°$ entonces ángulos de base en Y y X iguales a $\alpha + \beta$.

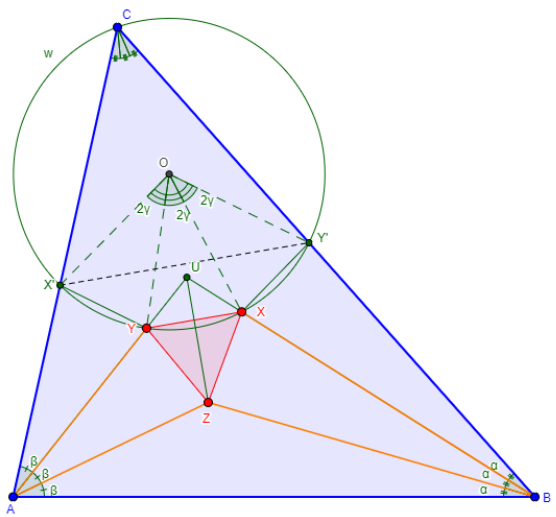


Figura 5 Ilustración para la demostración. Fuente propia.

3. Consideremos el ángulo en C, $C = 3\gamma$. Como $A + B + C = 180°$ entonces,
   $\alpha + \beta + \gamma = 60°$ y por tanto $\measuredangle UYX = \measuredangle UXY = \alpha + \beta = 60° - \gamma$ y $\measuredangle UYZ = 120° - \gamma$.

Creamos las condiciones para aplicar el lema sobre esta configuración. Para lo cual seleccionamos puntos X' y Y' en AC y BC tales que AX'=AZ y BY'=BZ entonces las parejas de triángulos AZY y AYX' y ZBX y XBY' son congruentes por lo que

$$X'Y = YZ = YX = ZX = XY'$$

y además $\sphericalangle UYZ = \sphericalangle UYX' = \sphericalangle UXZ = \sphericalangle UXY' = 60° - \gamma + 60° = 120° - \gamma$.

En estas condiciones comprobamos la segunda condición:

$$\begin{aligned}\sphericalangle XYX' &= \sphericalangle UYX' + \sphericalangle UYX' = (60° - \gamma) + (180° - \sphericalangle AYX) \\ &= (60° - \gamma) + (180° - \sphericalangle AYZ) = (60° - \gamma) + \sphericalangle UYX \\ &= (60° - \gamma) + (120° - \gamma) = 180° - 2\gamma > 60°\end{aligned}$$

puesto que $\gamma = \frac{A}{3} < 60°$.

De la misma manera, $\sphericalangle YXY' = 180° - 2\gamma$ y por aplicación del lema existe una circunferencia w a la cual pertenecen los puntos X, X', Y, Y', C,

Si O es el centro de esta circunferencia entonces al ser los triángulos OX'Y, OYX, OXY' congruentes entonces se tiene

$$\sphericalangle OX'Y = \sphericalangle OYX' = \sphericalangle OYX = \sphericalangle OXY = \sphericalangle OY'X = 90° - \gamma$$

por lo que $\sphericalangle X'OY = \sphericalangle YOX = \sphericalangle XOY' = 2\gamma$.

Dado que las cuerdas iguales Z'Y, YZ, ZY' subtienden ángulos iguales a γ en C, los segmentos CZ y CY trisecan el ángulo C del triángulo ABC. Es decir, los puntos X, Y, Z que fueron construidos artificialmente para formar un triángulo equilátero, son los puntos de intersección de las trisectrices del triángulo ABC.

1.3 LA DEMOSTRACIÓN TRIGONOMÉTRICA DE A. LETAC.

En esta sección consideramos una demostración del teorema de Morley que posee carácter trigonométrico y que de acuerdo con Bogomolny (s.f.) fue desarrollada por A. Letac en 1939. La prueba que presentamos sigue el espíritu de la incluida en Bogomolny (s.f.), sin embargo, se han complementado los detalles para permitir una lectura más ágil de la misma.

1.3.1 Teorema: Los puntos de corte de las trisectrices adyacentes de un triángulo cualquiera determinan un triángulo equilátero.

Demostración: consideramos un triángulo cualquiera ABC en el cual los ángulos en A, B y C sean tales que $A = 3\alpha, B = 3\beta, C = 3\gamma$ con $\alpha + \beta + \gamma = 60°$ como ilustra la Figura 6.

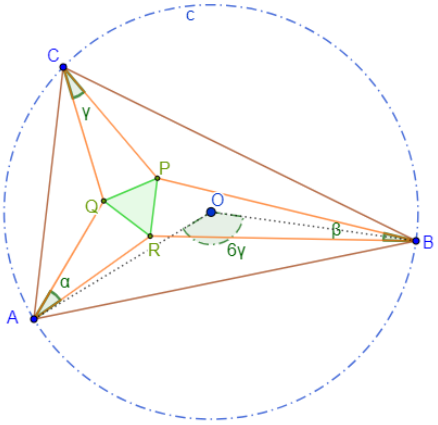


Figura 6 Ilustración para la demostración de A. Letac. Fuente propia.

La demostración consta esencialmente de los siguientes pasos:

1. Expresar las longitudes de los lados AB, BC, AC en términos de los ángulos α, β, γ.
2. Expresar las longitudes de los segmentos BP, CP, BR, AR, AQ, CQ en términos de estos mismos ángulos.
3. Encontrar expresiones para las longitudes de los lados PQ, QR y PR del triángulo PQR y demostrar que son las mismas.

Desarrollemos en detalle cada uno de estos pasos.

1. *Longitudes de los lados AB, BC, AC*. Sea c la circunferencia determinada por los puntos A, B y C y O su centro, puesto que $C = 3\gamma$ entonces un resultado de la geometría euclidiana, ver de Jesús L. F. (1987), nos asegura que para el ángulo central $\angle AOB$ se tiene $\angle AOB = 2(\angle ACB) = 6\gamma$.

Por el teorema del coseno

$$(AB)^2 = (AO)^2 + (OB)^2 - 2(AO)(OB)cos(\angle BOA).$$

Puesto que $OA = OB = r$, el radio de la circunferencia c, entonces

$$(AB)^2 = 2r^2(1 - cos(6\gamma))$$

y utilizando la fórmula del ángulo medio se tiene que

$$2sen^2(\gamma) = 1 - cos(6\gamma)$$

con lo que resulta que

$$AB = 2rsen(3\gamma)$$

y de forma análoga $BC = 2rsen(3\alpha)$ y $AC = 2rsen(3\beta)$.

2. *Longitudes de los segmentos BP, CP, BR, AR, AQ, CQ*. Por el teorema del seno en el triángulo BPC se tiene que

$$\frac{BP}{sen\,(\gamma)} = \frac{BC}{sen\,(180° - \gamma - \beta)} = \frac{2rsen\,(3\alpha)}{sen(\gamma + \beta)} = \frac{2rsen\,(3\alpha)}{sen\,(60° - \alpha)}$$

así que

$$BP = \frac{2rsen\,(3\alpha)\,sen\,(\gamma)}{sen\,(60° - \alpha)}.$$

Simplificamos esta expresión para lo cual utilizaremos la fórmula del ángulo triple y la identidad trigonométrica pitagórica. Veamos esto.

$$\begin{aligned}\operatorname{sen}(3\alpha) &= 3\operatorname{sen}(\alpha) - 4sen^3(\alpha)\\ &= 4\operatorname{sen}(\alpha)\left(\tfrac{3}{4} - sen^2(\alpha)\right)\\ &= 4\operatorname{sen}(\alpha)\left(\tfrac{3}{4}(cos^2(\alpha) + sen^2(\alpha)) - sen^2(\alpha)\right)\\ &= 4\operatorname{sen}(\alpha)\left(\tfrac{\sqrt{3}}{2}cos(\alpha))^2 - (\tfrac{1}{2}sen(\alpha))^2\right)\end{aligned}$$

$$= 4\operatorname{sen}(\alpha)\left(\frac{\sqrt{3}}{2}cos(\alpha) + \frac{1}{2}sen(\alpha)\right)\left(\frac{\sqrt{3}}{2}cos(\alpha) - \frac{1}{2}sen(\alpha)\right)$$

$$=4\operatorname{sen}(\alpha)\big(sen(60°)cos(\alpha)\cos(60°)\,sen(\alpha)\big)en\ (60°)$$
$$cos(\alpha)\ cos\ (60°)\ sen(\alpha))$$

$$= 4\operatorname{sen}(\alpha)\ sen(60° + \alpha)sen(60° - \ \alpha)$$

Por tanto, al reemplazar esta expresión en la fórmula para BP se obtiene que

$$BP = 8r\ \operatorname{sen}(\gamma)\ \operatorname{sen}(\alpha)\ \operatorname{sen}(60° + \alpha). \qquad (1)$$

Ahora, considerando el triángulo BRA, se obtiene de forma semejante que

$$BR = 8r\ \operatorname{sen}(\alpha)\ \operatorname{sen}(\gamma)\ \operatorname{sen}(60° + \gamma). \qquad (2)$$

y procediendo de manera similar, podemos determinar ecuaciones para las longitudes de los lados CP, CQ, AQ y AR.

3. *Longitudes de los lados PR, PQ y QR*. Consideremos el triángulo BPR en el cual conocemos los lados BP, BR y que $\angle PBR = \beta$. Por el teorema del coseno,

$$(PR)^2 = (BP)^2 + (BR)^2 - 2(BP)(BR)cos\,(\beta)\,.$$

Si en esta expresión reemplazamos los valores de BP y BR dados por los términos (1) y (2) tenemos que

$$(PR)^2 = 64\,sen^2(\gamma)\,sen^2(\alpha)[sen^2(60° + \alpha)sen^2(60° + \gamma) - 2(sen\,(60°\,\alpha)sen\,(60° + \gamma)cos\,(\beta)] \qquad (3)$$

Ahora simplificamos la expresión entre corchetes que depende de los ángulos $\beta, 60° + \gamma$ y $60° + \alpha$, para lo cual construimos un triángulo auxiliar MNL que esté inscrito en una circunferencia de radio r.
Nótese en primer lugar que este triángulo existe ya que,

$$\beta + (60° + \alpha) + (60° + \gamma) = 120° + (\alpha + \beta + \gamma) = 180°$$

y por tanto podemos representar esquemáticamente este triángulo como en la Figura 7, para la cual se tiene, replicando el argumento utilizado en la parte 1, que:

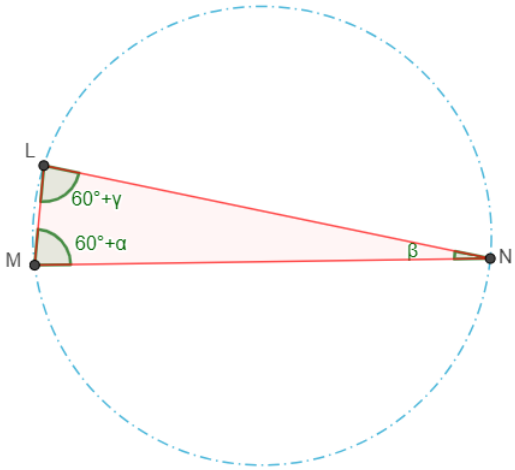


Figura 7 Ilustración para la demostración. Fuente propia.

$$LM = 2rsen(\beta), LN = 2rsen(60° + \alpha), MN = 2rsen(60° + \gamma). \quad (4)$$

Por el teorema del coseno

$$(LM)^2 = (LN)^2 + (MN)^2 - 2(LN)(MN)cos(\beta)$$

Si en esta expresión reemplazamos las expresiones dadas por (4) y simplificamos, obtenemos que $sen^2(\beta)$ se puede expresar como:

$$sen^2(\beta) = sen^2(60° + \alpha) + sen^2(60° + \gamma) - 2sen(60° + \alpha)sen(60° + \alpha)cos(\beta)$$

y al reemplazar esta expresión en (3) resulta que

$$PR = 8r \operatorname{sen}(\alpha)\, sen(\gamma)\, sen(\beta)$$

puesto que $PR > 0$ y α, β, γ están entre 0 y 60°.

Procediendo exactamente de la misma manera con los triángulos CPQ y AQR se encuentra que

$$PQ = 8r \operatorname{sen}(\alpha)\, sen(\gamma)\, sen(\beta)$$

y

$$QR = 8r \operatorname{sen}(\alpha)\, sen(\gamma)\, sen(\beta)$$

de lo que se concluye que el triángulo PQR es equilátero.

Nótese que esta demostración utiliza únicamente argumentos de matemáticas elementales y aunque esta es transparente, las ideas y pormenores algebraicos en la misma no son sencillos y requieren ser bastante creatividad.

1.4 LA DEMOSTRACIÓN TRIGONOMÉTRICA DE A. CHAVES.

En esta sección se trata la demostración del teorema de Morley de naturaleza trigonométrica que fue desarrollada por el ingeniero civil Nariñense A. Chaves quien fue profesor de la Universidad Nacional (sede Manizales). Esta prueba se encuentra en Chaves (1993).

1.4.1 Teorema: los puntos de corte de las trisectrices adyacentes de un triángulo cualquiera determinan un triángulo equilátero.

Demostración: consideramos un triángulo cualquiera ABC con ángulos en A, B y C tales que $A = A = \alpha, B = \beta, C = \gamma$ y cuyas trisectrices adyacentes se intersecan en los puntos P, Q y R como ilustra la Figura 8. La demostración consiste en demostrar que los ángulos en P, Q y R, del triángulo PQR, tienen como medida 60°. Veamos esto.

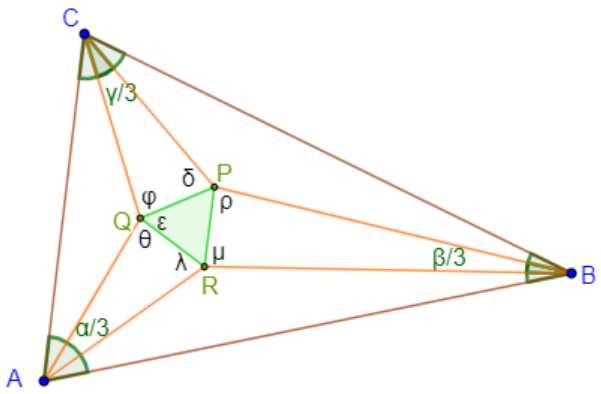


Figura 8 Ilustración para la demostración de A, Chaves. Fuente propia.

Si r es el radio de la circunferencia circunscrita al triángulo $\measuredangle ABC$ entonces por el teorema del seno y el teorema del coseno, se obtiene que

$$\frac{BC}{sen\ (\alpha)} = \frac{AC}{sen\ (\beta)} = \frac{AB}{sen\ (\gamma)} = 2r$$

Por lo que

$$BC = 2r\ sen\ (\alpha), \qquad AC = 2r\ sen(\beta)$$

De la misma manera en el triángulo AQC

$$CQ = \frac{AC\ sen(\frac{\alpha}{3})}{sen\ (180° - \frac{\alpha}{3} - \frac{\gamma}{3})} = \frac{2r\ sen(\beta)\ sen(\frac{\alpha}{3})}{sen\ (\frac{\alpha}{3} + \frac{\gamma}{3})}$$

Puesto que $\frac{\alpha}{3} + \frac{\gamma}{3} + \frac{\beta}{3} = 60°$ entonces $\frac{\alpha}{3} + \frac{\beta}{3} = 60° - \frac{\gamma}{3}$ por lo que

$$CQ = \frac{2r\ sen(\beta)\ sen(\frac{\alpha}{3})}{sen\ (60° - \frac{\beta}{3})} \qquad (1)$$

Argumentos sobre identidades trigonométricas nos muestran que

$$\begin{aligned} sen\,(3z) = sen\,(2z+z) &= sen\,(2z)\,cos\,(z) + sen\,(z)cos\,(2z) \\ &= sen\,(z)(2\,cos^2(z) + cos(2z)) \\ &= sen\,(z)(3\,cos^2(z) - sen^2(z)) \\ &= 4sen\,(z)(\frac{3}{4}\,cos^2(z) - \frac{1}{4}sen^2(z)) \end{aligned}$$

entonces

$$\frac{sen(3z)}{4sen(z)} = sen^2(60°)\,cos^2(z) - sen^2(60°)sen^2(z)$$

$$= (sen\,(60°)\,cos(z) + cos(60°)\,sen(z))(sen\,(60°)\,cos(z) - cos(60°)\,sen(z))$$

y, por tanto

$$\frac{sen(3z)}{4sen(z)} = sen\,(60° + z)\,sen(60° - z) \qquad (2)$$

Si en (2) se hace $z = \frac{\beta}{3}$ entonces

$$\frac{sen(\beta)}{sen\left(60° - \frac{\beta}{3}\right)} = 4\,sen\,(\frac{\beta}{3})\,sen(60° + \frac{\beta}{3})$$

Y reemplazando esta expresión en (1) resulta que

$$CQ = 8r\ sen\ (\frac{\alpha}{3})\ sen\ (\frac{\beta}{3})\ sen(60° + \frac{\beta}{3}) \quad (3)$$

Análogamente

$$CP = 8r\ sen\ (\frac{\beta}{3})\ sen\ (\frac{\alpha}{3})\ \ sen(60° + \frac{\alpha}{3}) \quad (4)$$

Si ahora consideramos el triángulo CQP, se tiene, por el teorema del seno, que

$$\frac{CP}{sen\ (\varphi)} = \frac{CQ}{sen\ (\delta)}$$

Y utilizando las expresiones (3) y (4) obtenemos que

$$\frac{sen\ (\delta)}{sen\ (\varphi)} = \frac{CQ}{CP} = \frac{sen(60° + \frac{\beta}{3})}{sen(60° + \frac{\alpha}{3})}$$

Dado que

$$\varphi + \delta = 180° - \frac{\gamma}{3} = 120° + (60° - \frac{\gamma}{3}) = 120° + \frac{\alpha}{3} + \frac{\beta}{3}$$

entonces podemos considerar el sistema no lineal de ecuaciones en las variables φ, δ

$$\varphi + \delta = 120° + \frac{\alpha}{3} + \frac{\beta}{3}$$

$$\frac{sen\ (\delta)}{sen\ (\varphi)} = \frac{sen(60° + \frac{\beta}{3})}{sen(60° + \frac{\alpha}{3})}$$

que tiene como solución, en términos de α y β

$$\varphi = 60° + \frac{\alpha}{3} \quad \text{y} \quad \delta = 60° + \frac{\beta}{3}$$

De forma análoga, considerando los triángulos BPR y AQR se obtiene que

$$\rho = 60° + \frac{\alpha}{3}, \mu = 60° + \frac{\gamma}{3}, \lambda = 60° + \frac{\beta}{3} \text{ y } \theta = 60° + \frac{\gamma}{3}.$$

Si ahora consideramos los ángulos alrededor del punto Q tenemos que

$$\angle AQC + \varphi + \epsilon + \theta = 360°$$

es decir

$$180° - \frac{\alpha}{3} - \frac{\gamma}{3} + 60° + \frac{\alpha}{3} + \epsilon + 60° + \frac{\gamma}{3} = 360°$$

y por tanto $\varepsilon = 60°$.

Similarmente se muestra que, los ángulos en P y R del triángulo PQR tienen como medida 60° y consecuentemente el triángulo PQR es equilátero.

2. ORIGAMI DEL TRIÁNGULO DE MORLEY

En el fascinante mundo del origami, donde el papel cobra vida en formas geométricas y artísticas, existe una construcción matemática que ha capturado la imaginación tanto de los matemáticos como de los origamistas: el **Triángulo de Morley**. Si bien el teorema de Morley es sorprendente por sí mismo, lo es aún más cuando se combina con el arte del origami para construir dicho triángulo utilizando solo pliegues de papel.

Algunos trabajos publicados que abordan el **Origami del Triángulo de Morley** incluyen contribuciones de Tetsuo Ida y colaboradores. En particular, el artículo "Computational Origami of a Morley's Triangle" presenta la construcción del triángulo de Morley utilizando el método de Abe, un enfoque basado en origami que permite trisectar los ángulos del triángulo original y generar un triángulo equilátero. Este trabajo destaca el uso de métodos algorítmicos para resolver las restricciones geométricas mediante bases de Gröbner, lo que hace posible la automatización de la construcción y la demostración matemática del teorema generalizado de Morley.

Otra publicación relevante es "Morley's Theorem Revisited: Origami Construction and Automated Proof," donde se detalla cómo se construyen las trisectrices de los ángulos y se comprueban los resultados geométricos mediante herramientas computacionales como el software Eos.

Estos estudios se centran en la intersección entre el origami y las matemáticas computacionales, demostrando que la construcción de un triángulo de Morley, que no es posible con regla y compás, puede lograrse utilizando pliegues de papel.

2.1 ORIGAMI COMPUTACIONAL

Además de **Eos**, que es un sistema de origami computacional desarrollado por Tetsuo Ida y colaboradores, existen varios programas diseñados para simular y asistir en la creación de origami. Algunos ejemplos destacados incluyen:

- **TreeMaker:** Desarrollado por Robert Lang, este software permite a los usuarios crear estructuras complejas de origami mediante el uso de algoritmos matemáticos. TreeMaker es especialmente útil para aquellos interesados en el plegado técnico y avanzado.
- **Origamizer:** Creado por Tomohiro Tachi, Origamizer permite diseñar patrones de origami con superficies poligonales. Este software es útil para construir formas tridimensionales a partir de patrones planos.
- **ORIPA:** También desarrollado por Tomohiro Tachi, ORIPA ayuda a crear patrones de pliegue y verifica si los patrones se pueden plegar físicamente en un modelo tridimensional. Es ideal para quienes diseñan modelos de origami y quieren probar la viabilidad de sus patrones.

- **Freeform Origami:** Este es otro software de Tomohiro Tachi, que permite manipular formas tridimensionales de origami libremente, ideal para la creación de formas no convencionales.

Estos programas utilizan matemáticas avanzadas y algoritmos geométricos para simplificar y optimizar el diseño de pliegues de origami, y se han convertido en herramientas valiosas tanto para artistas como para matemáticos que trabajan con el origami.

En el Sitio Web del Institute for Educational Origami se encuentra el siguiente texto:

> El origami, el arte de plegar papel, ha capturado la mente de las personas por su belleza y utilidad en la vida cotidiana. El origami también ha sido objeto de estudios matemáticos. Las operaciones de plegado son simples y fáciles de realizar a mano, pero lo suficientemente poderosas como para resolver ecuaciones cúbicas. El origami se considera una herramienta geométrica poderosa que permite construcciones que las herramientas clásicas euclidianas no pueden realizar. (EOS project, s.f.)

En este sitio, que corresponde al Proyecto Eos y presenta múltiples ejemplos de su uso, se puede descargar versiones de este software. La Figura 9 muestra algunos ejemplos de instrucciones de este software y el resultado gráfico de los pliegues origamis producidos. Específicamente, esta figura corresponde a las tres instrucciones iniciales para crear un

heptágono regular. Lo que hacen estas instrucciones es crear la hoja inicial, hacer un pliegue por la mitad y desdoblar la hoja.

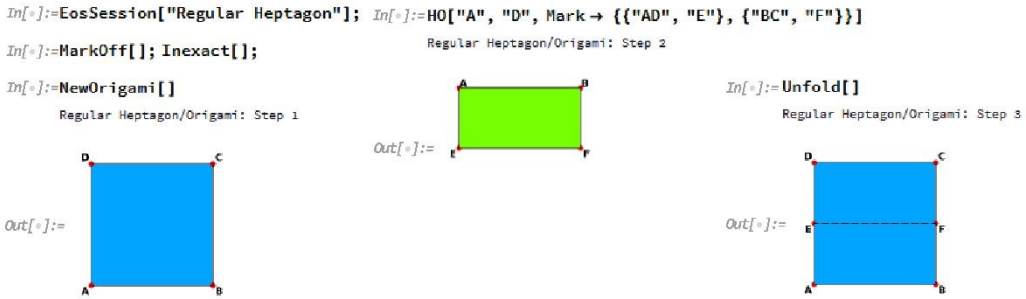


Figura 9 Ejemplos de instrucciones de Eos. Fuente EOS project.

El trazo de un heptágono regular es uno de los problemas clásicos de la geometría y al igual que la trisección de un ángulo, no se puede hacer utilizando regla y compás. En cambio, el Origami computacional puede resolver estos problemas. La Figura 10 muestra las dos últimas instrucciones de la construcción del Heptágono regular.

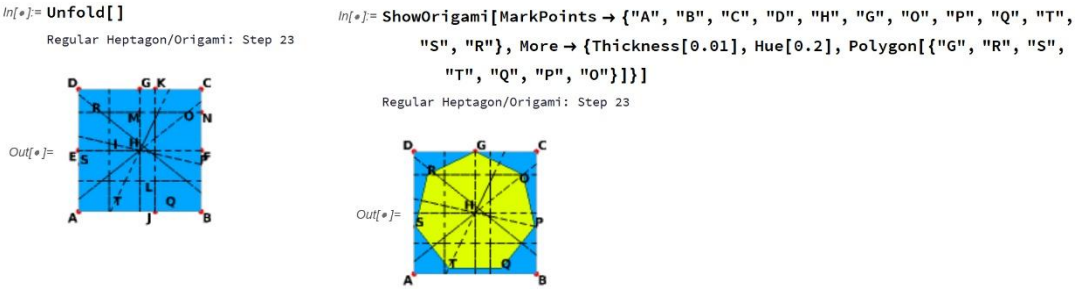


Figura 10 Dos últimas instrucciones del heptágono regular. Fuente EOS project.

El Origami computacional se define como “una disciplina científica que estudia los aspectos matemáticos y computacionales del origami. Incluye el estudio matemático de los pliegues de papel, la modelización del origami mediante métodos algebraicos y simbólicos, la simulación por computadora de los pliegues de papel, y la demostración de la corrección de las propiedades geométricas del origami construido” (Tetsuo, Asem y otros, 2011)

Los **axiomas de Huzita** (1989) son fundamentales para el desarrollo de la **teoría del Origami Computacional**. Estos axiomas, propuestos por el físico japonés Humiaki Huzita, describen formalmente las reglas que permiten realizar construcciones geométricas utilizando solo el plegado de papel. Además, son un conjunto de operaciones fundamentales que definen cómo se pueden crear y manipular líneas de pliegue en el origami, que se expresan como sigue:

Sean Π y Γ el conjunto de puntos construibles y de líneas construibles, respectivamente.

- **Axioma 1**: Dado dos puntos en Π, podemos hacer un pliegue a lo largo de la línea de plegado que pasa por ellos.
- **Axioma 2**: Dado dos puntos en Π, podemos hacer un pliegue para llevar uno de los puntos sobre el otro.
- **Axioma 3**: Dado dos líneas en Γ, podemos hacer un pliegue para superponer las dos líneas.

- **Axioma 4**: Dado un punto P en Π y una línea m en Γ, podemos hacer un pliegue a lo largo de la línea de plegado que es perpendicular a m y que pasa por P.
- **Axioma 5**: Dado dos puntos P y Q en Π y una línea m en Γ, podemos o construir la línea de plegado que pasa por Q y hacer un pliegue a lo largo de esta línea para superponer P y m, o podemos decidir que la construcción de tal línea de plegado es imposible.
- **Axioma 6**: Dado dos puntos P y Q en Π y dos líneas m y n en Γ, podemos o construir una línea de plegado y hacer un pliegue a lo largo de esta línea para superponer simultáneamente P con m y Q con n, o podemos decidir que la construcción de tal línea de plegado es imposible.

Más tarde, **Koshiro Hatori** (2005), propuso un séptimo axioma que se expresa así:

- **Axioma 7**: Dado un punto P en Π y dos líneas m y n en Γ, podemos o construir la línea de plegado que es perpendicular a n y hacer un pliegue a lo largo de esta línea para superponer P con m, o podemos decidir que la construcción de tal línea de plegado es imposible.

Estos axiomas son análogos a las construcciones clásicas de la geometría euclidiana, pero tienen una potencia adicional: el origami permite realizar construcciones que no son posibles con regla y compás, como la **trisección de un ángulo**, una tarea imposible con las herramientas euclidianas tradicionales.

2.2 EL ORIGAMI DEL TRIÁNGULO DE MORLEY

Este origami combina la belleza matemática con la precisión geométrica del plegado, permitiendo crear un triángulo equilátero a partir de cualquier triángulo inicial. En esta sección, exploraremos cómo los principios del origami pueden aplicarse para construir el Triángulo de Morley, abordando desde los conceptos básicos hasta las técnicas de pliegue avanzadas que permiten realizar esta construcción de manera precisa.

2.2.1 EL INICIO DEL PROCESO: PREPARACIÓN

Todo comienza con un cuadrado de papel, la base tradicional del origami. El cuadrado es marcado con los puntos A, B, C y D en sus vértices, estableciendo así un marco de referencia en el que se realizará la construcción. A continuación, se selecciona un punto arbitrario E cercano a uno de los lados del cuadrado, creando el triángulo inicial $\triangle ABE$, sobre el cual se generará el Triángulo de Morley. Este paso es fundamental, ya que el punto E definirá la base de los pliegues necesarios para trisectar los ángulos.

El primer pliegue, sencillo pero crucial, se realiza a lo largo de la línea que conecta los puntos A y E. Este pliegue divide el triángulo en dos partes desiguales, y se utiliza como referencia para los siguientes pliegues más complejos. El objetivo es trisectar cada uno de los ángulos del triángulo $\triangle ABE$ utilizando técnicas avanzadas de origami.

2.2.2 EL MÉTODO DE ABE: TRISECANDO LOS ÁNGULOS

La trisección de los ángulos es una de las tareas más complicadas en la geometría clásica, pero en el origami se logra mediante una serie de pliegues precisos. El método utilizado para trisectar los ángulos en el origami del Triángulo de Morley se basa en el **Método de Abe**, una técnica desarrollada por el matemático japonés Tetsuo Abe. Este método utiliza las propiedades del papel y las líneas de pliegue para lograr una precisión geométrica que no se puede alcanzar con regla y compás. El método en sí, se pone en práctica al final de este capítulo con la obtención real del origami del Triángulo de Morley.

Abe aplica los axiomas de Huzita, fundamentalmente el Axioma 6 en un algoritmo que obtiene la trisección de los ángulos de un triángulo. Con los ángulos ∢EAB, ∢ABE y ∢BEA correctamente trisecados, se forman puntos de intersección entre los trisectores adyacentes de cada ángulo que, al unirlos forman el **Triángulo de Morley**, es una construcción sorprendente, ya que su existencia no es intuitiva al observar el triángulo original. Sin embargo, los pliegues del origami revelan su presencia de manera tangible.

El Triángulo de Morley es notable no solo por su simetría perfecta, sino también porque es una construcción que desafía los límites de la geometría clásica. En el mundo de la regla y el compás, la trisección de un ángulo es imposible de lograr con precisión, pero el origami, con sus sencillos pliegues, permite realizar esta tarea de manera exacta y

elegante. Es en esta combinación de arte y matemáticas donde reside la verdadera magia del origami del Triángulo de Morley.

2.2.3 LA ELEGANCIA DEL ORIGAMI MATEMÁTICO

El origami del Triángulo de Morley no solo es una construcción geométrica precisa, sino que también es un testimonio de la belleza inherente de las matemáticas y su capacidad para fusionarse con el arte. Cada pliegue, cada línea de intersección y cada punto de contacto entre las líneas de trisectores cuenta una historia de simetría y equilibrio.

La capacidad de crear un triángulo equilátero perfecto dentro de cualquier triángulo mediante simples pliegues es un recordatorio de que las matemáticas y el arte no están tan alejados como se perciben. El papel, en manos de un origamista, se convierte en una herramienta para explorar los misterios geométricos que han intrigado a los matemáticos durante siglos.

2.3 CONSTRUCCIÓN DEL ORIGAMI DEL TRIÁNGULO DE MORLEY

2.3.1 PREPARACIÓN DEL PAPEL

Este origami se construye utilizando el método de Abe, descrito por Tetsuo, Asem y otros (2011). El proceso comienza con un cuadrado ABCD ubicado en el centro de una hoja de papel, ya sea impreso o dibujado como muestra la Figura 11. Para formar un triángulo, se selecciona un punto arbitrario E, el cual define el triángulo $\triangle ABE$, sobre el que se construirá el triángulo de Morley. Inicialmente, se realiza un pliegue que

pasa por los puntos A y E, seguido de otro doblez uniendo los puntos A con D y B con C, dividiendo así el cuadrado en dos partes. A continuación, se marcan los puntos F y G, que corresponden a los puntos medios de los lados respectivos.

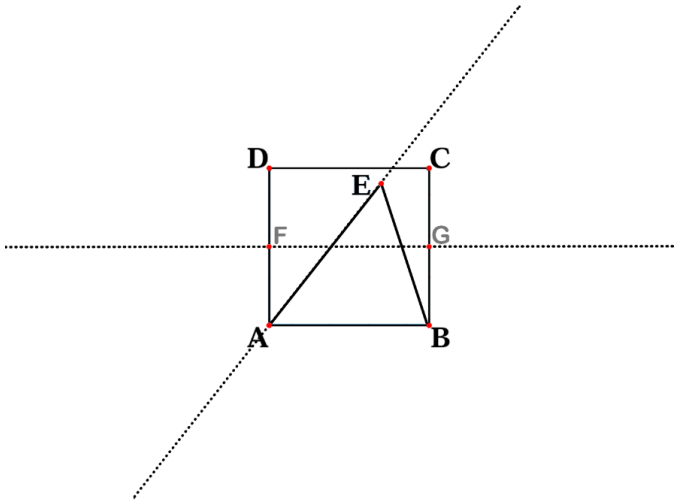


Figura 11 Hoja de papel impresa con un cuadrado. Fuente propia.

2.3.2 APLICACIÓN DEL AXIOMA 6 DE HUZITA

Se pueden realizar tres pliegues bajo la siguiente condición: que el punto D quede sobre la recta AE y que el punto A se posicione simultáneamente sobre la recta FG. En esta sección, se explicará cómo encontrar dos de estos tres pliegues. No obstante, si desea proceder directamente con la trisección del ángulo ∡EAB para construir el Triángulo de Morley, se recomienda ir a la sección 2.3.3.

Para realizar el primer pliegue que cumple con la condición establecida, resalta la recta FG en el reverso de la hoja como lo muestra la Figura 12. Muchas de las figuras fueron obtenidas trazando a contraluz sobre un vidrio de una ventana, por lo que se recomienda emplear la misma técnica para obtener mejores resultados. También es conveniente para mejor comprensión del procedimiento, observar la figura del pliegue terminado y regresar a la anterior que muestra el procedimiento.

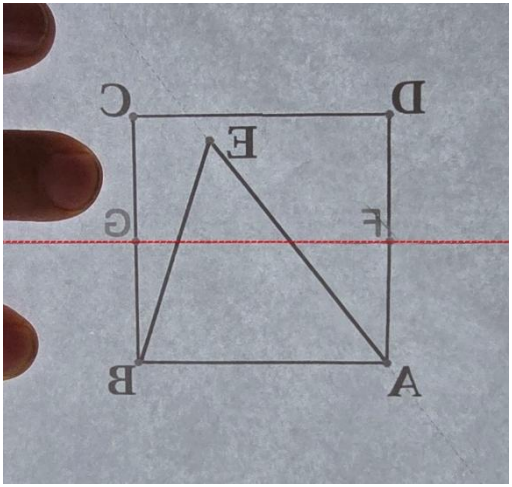


Figura 12 Resaltado de la recta FG por el reverso de la hoja. Fuente propia.

A continuación, desde el frente de la hoja, realiza el pliegue de modo que el punto D quede alineado con la recta AE y el punto A se posicione sobre la recta FG simultáneamente. Observa en la Figura 13 cómo el punto A se sitúa correctamente en la recta FG, gracias a la línea resaltada previamente en el reverso de la hoja.

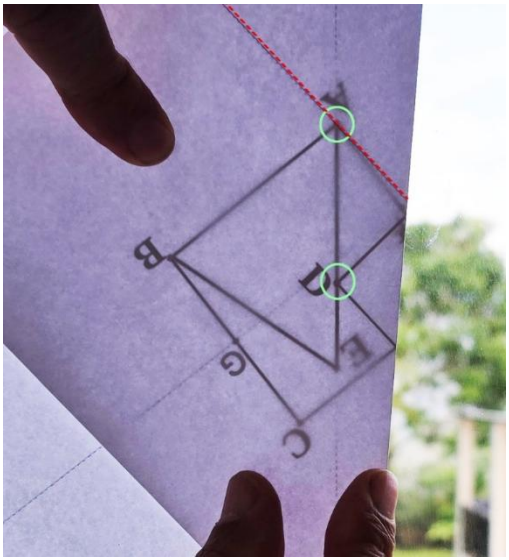


Figura 13 Pliegue a contraluz. Fuente propia.

El doblez obtenido se muestra en la siguiente Figura 14.

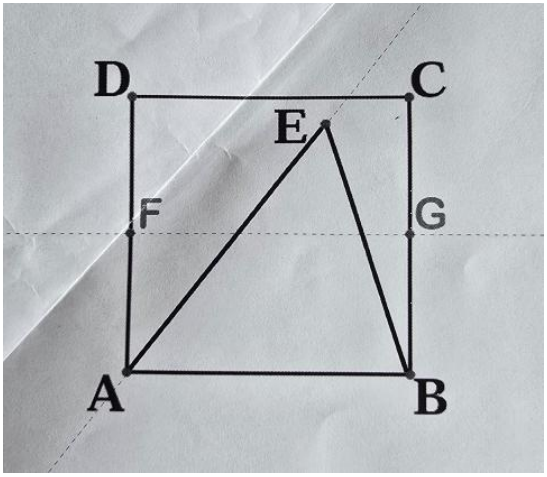


Figura 14 Pliegue obtenido. Fuente propia.

Para el segundo pliegue que cumple con la siguiente condición: que el punto D quede alineado con la recta AE, y el punto A se posicione sobre la recta FG simultáneamente, es conveniente repintar por el reverso de la hoja la recta AE como se ilustra en la Figura 15.

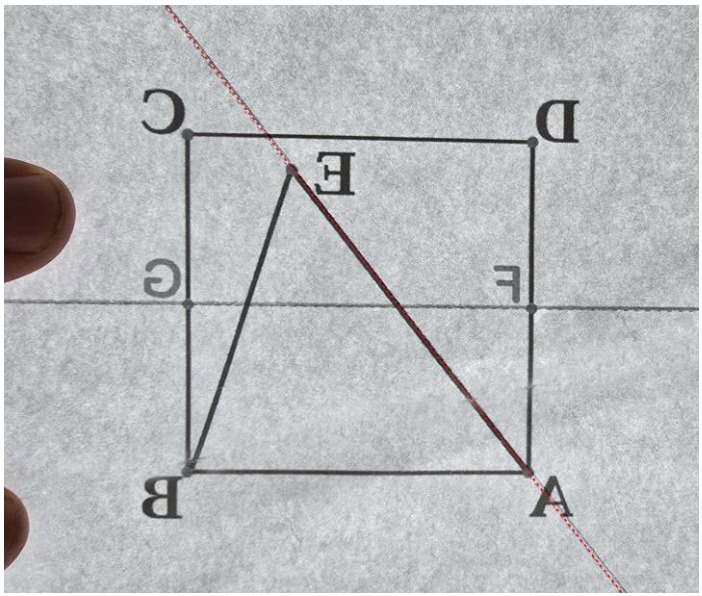


Figura 15 Resaltado de la recta AE por el reverso de la hoja. Fuente propia.

Ahora por el frente de la hoja dóblela hasta que el punto D quede sobre la recta AE y que el punto A quede sobre la recta FG. Note en la Figura 16 que el punto D queda en la recta AE con la ayuda de la línea que se resaltó por el reverso de la hoja, y que el punto A está sobre la recta FG punteada en un tramo con color verde.

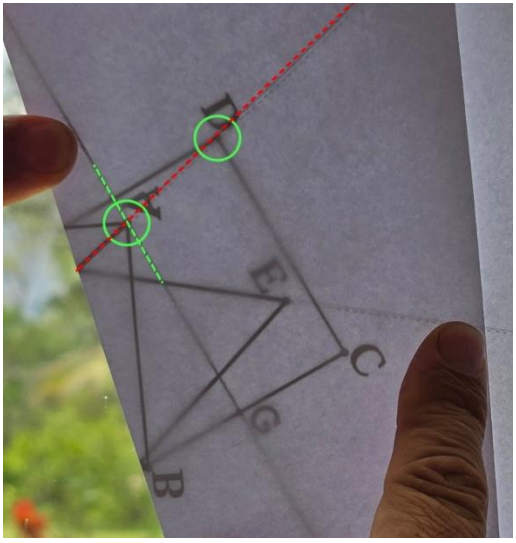


Figura 16 Pliegue a contraluz. Fuente propia.

La Figura 17 ilustra la posición del segundo doblez que cumple la condición citada.

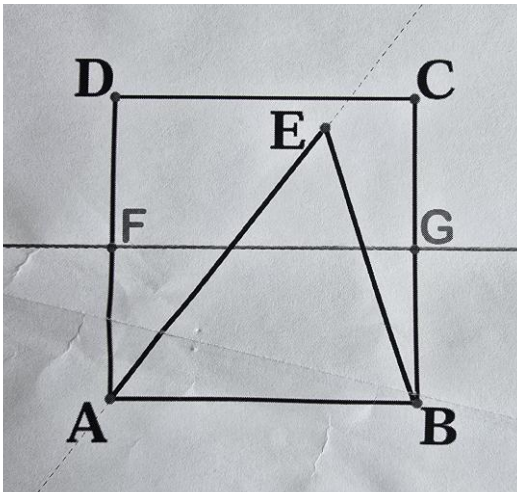


Figura 17 Pliegue obtenido. Fuente propia.

2.3.3 TRISECCIÓN DEL $\measuredangle EAB$

Las siguientes figuras fueron elaboradas en varias hojas de papel para evitar una sobrecarga de pliegues en una sola hoja, con el propósito de presentar el procedimiento de la manera más clara y didáctica posible. Sin embargo, en la práctica, todos los pliegues deben realizarse en una única hoja.

A continuación, se construirá el tercer pliegue que es el indicado para trisecar el ángulo ∡EAB, al cumplir la condición de alinear simultáneamente el punto D con la recta AE y el punto A sobre la recta FG. En la Figura 18, se puede observar cómo el punto D se encuentra sobre la recta AE (marcada con una línea verde punteada) y el punto A se superpone a la recta FG (marcada con una línea roja punteada).

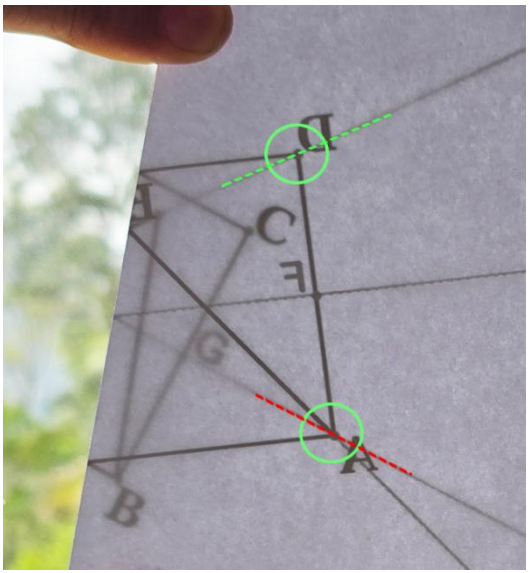


Figura 18 Pliegue a contraluz. Fuente propia.

Se puede ver el pliegue obtenido en la Figura 19.

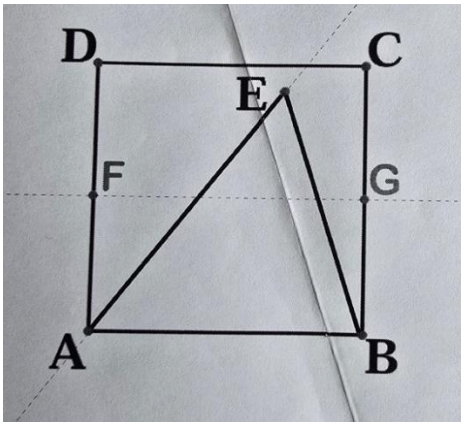


Figura 19 Pliegue obtenido. Fuente propia.

Doble nuevamente la hoja por el reciente pliegue y mediante alfileres o una aguja, determine la proyección de los puntos A y D como se muestra en la Figura 20.

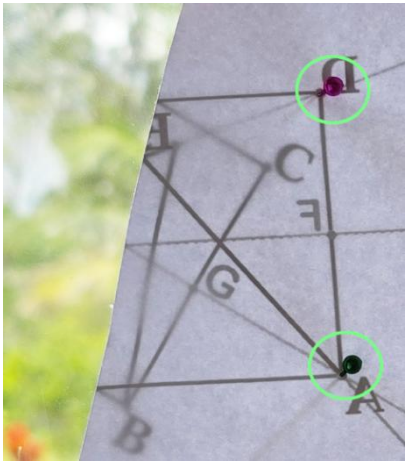


Figura 20 Alfileres marcando proyecciones de puntos. Fuente propia.

Los puntos son H e I resultado de las proyecciones de A y D, tienen la propiedad que los dobleces hechos con el punto A trisecan el ∡EAB. Observe la Figura 21.

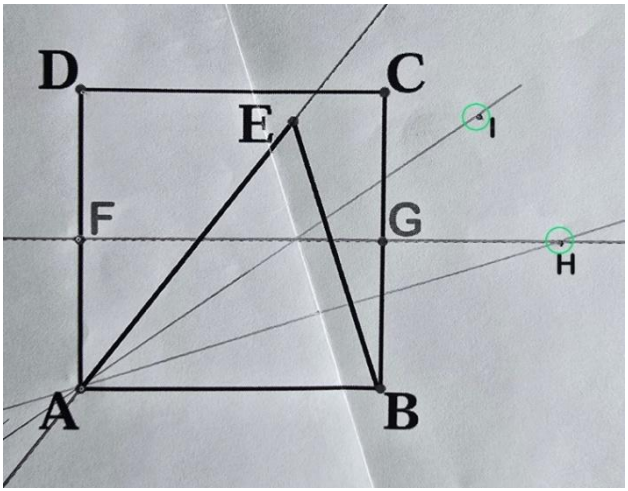


Figura 21 Trisección del ángulo ∡EAB. Fuente propia.

2.3.4 TRISECCIÓN DEL ∡*ABE*

Se procederá a realizar un pliegue que cumpla con la siguiente condición: que el punto C se superponga a la recta BE y el punto B a la recta FG de manera simultánea. La Figura 22 ilustra el procedimiento. En ella se ha repintado la línea FG con un trazo verde punteado y la línea BE con un trazo rojo punteado.

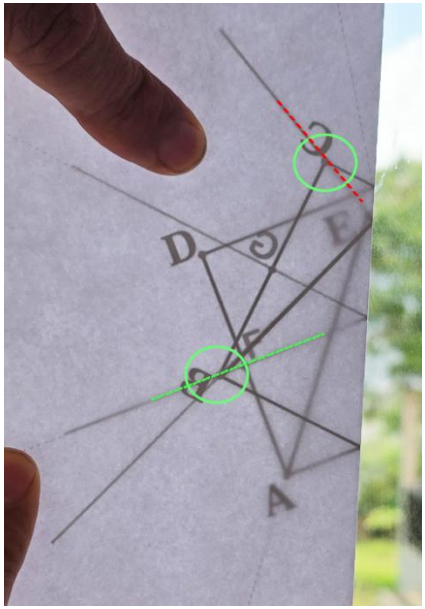


Figura 22 Pliegue a contraluz. Fuente propia.

El doblez debe quedar como se muestra a continuación.

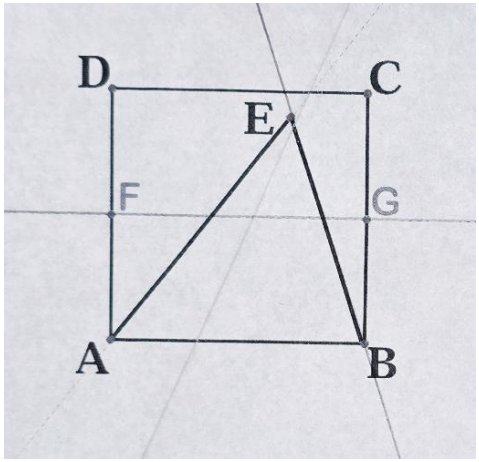


Figura 23 Pliegue obtenido. Fuente propia.

Doblamos nuevamente la hoja siguiendo el último pliegue realizado y, con la ayuda de alfileres o una aguja, perforamos los puntos G y B para marcar sus proyecciones. Ver Figura 24.

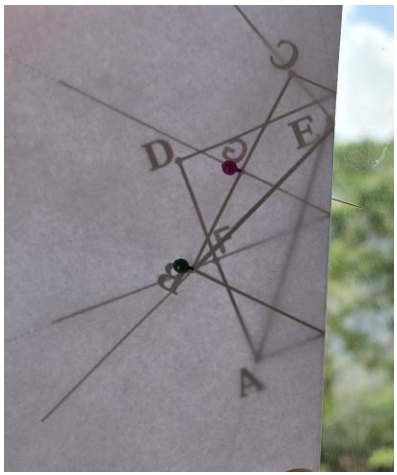


Figura 24 Alfileres marcando proyecciones de puntos. Fuente propia.

Se generan los puntos J y K, cuyos pliegues con el punto B trisecan el ángulo $\measuredangle ABE$ como se puede ver en la Figura 25.

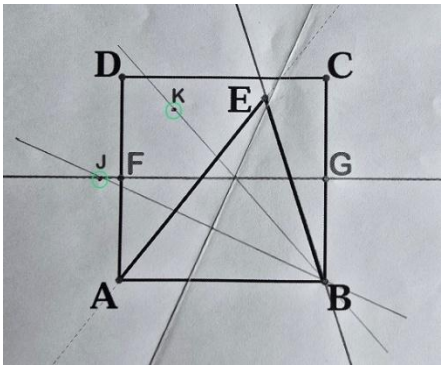


Figura 25 Trisección del ángulo $\measuredangle ABE$. Fuente propia.

2.3.5 TRISECCIÓN DEL $\measuredangle BEA$

Trisectar el ángulo ∡BEA es más complejo que los otros ángulos. Razón por la cual, para lograrlo, primero debemos construir una perpendicular a la recta BE que pase por el punto E. Para facilitar el proceso, plegamos la hoja por recta BE y la resaltamos por el reverso. Observe las Figuras 26 y 27.

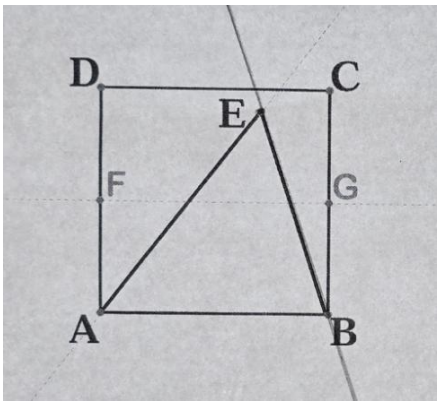


Figura 26 Pliegue por la recta BE. Fuente propia.

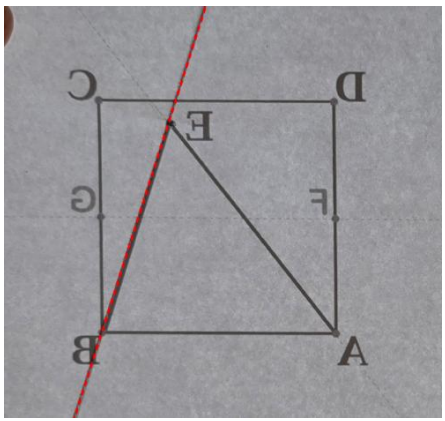


Figura 27 Resaltado de la recta BE por el reverso de la hoja. Fuente propia.

Para trazar una perpendicular a la recta BE que pase por el punto E, simplemente hay que doblar la hoja por el punto E, asegurándose de alinear los trazos de la recta BE hechos en ambos lados de la hoja, tal como se muestra en la Figura 28.

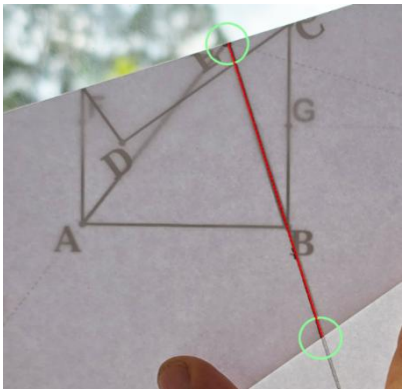


Figura 28 Pliegue a contraluz. Fuente propia.

El pliegue de la recta perpendicular puede observarse en la figura siguiente. El punto M representa la intersección entre la línea de pliegue y la recta AD.

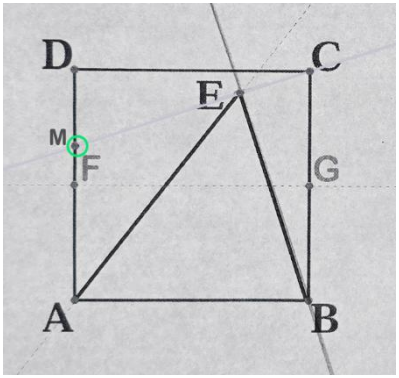


Figura 29 Pliegue obtenido. Fuente propia.

Recordando la trisección del ∡EAB, donde se trabajó con los puntos D, situado sobre la perpendicular a la recta AB, y F, ubicado en la paralela a AB y en la mitad entre A y D, ahora tenemos el punto M sobre la perpendicular a BE, pero faltaría aun determinar el punto medio entre E y M sobre una paralela a BE, el cual llamaremos N. Para encontrar este punto, realizamos un pliegue uniendo los puntos E y M, como se muestra en la Figura 30.

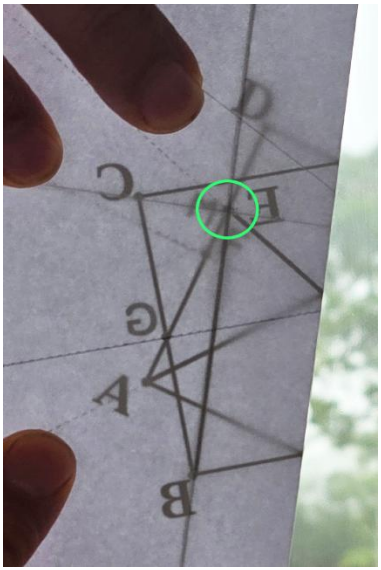


Figura 30 Pliegue a contraluz. Fuente propia.

Se ha identificado el punto N, así como el punto O, que corresponde a la intersección del pliegue con el lado AB. Observe la Figura 31.

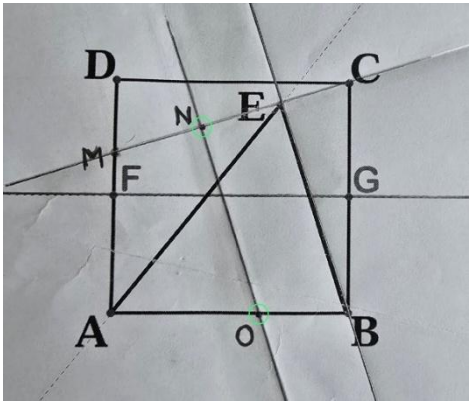


Figura 31 Puntos obtenidos. Fuente propia.

Ahora hacemos un doblez logrando que E se superponga a la recta NO y M se superponga a la recta AE. Sabemos que hay tres pliegues posibles, pero el adecuado es el mostrado en la Figura 32, donde la recta NO se ha punteado con rojo y la recta AE con verde.

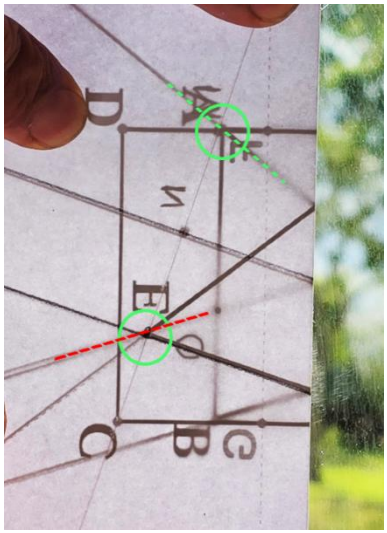


Figura 32 Pliegue a contraluz. Fuente propia.

En este último pliegue, utilizando alfileres o una aguja, marcamos las proyecciones de los puntos N y E como se ilustra en la Figura 33.

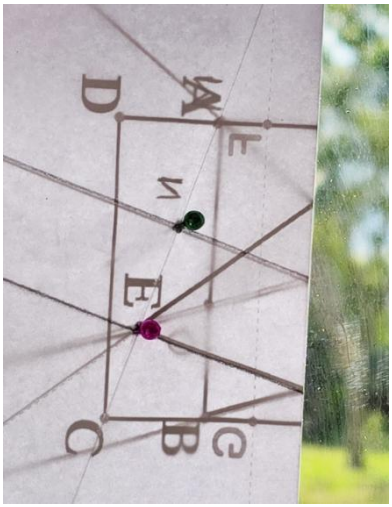


Figura 33 Alfileres marcando proyecciones de puntos. Fuente propia.

Obtenemos los puntos P y Q como proyecciones de los puntos E y N respectivamente. Ver Figura 34.

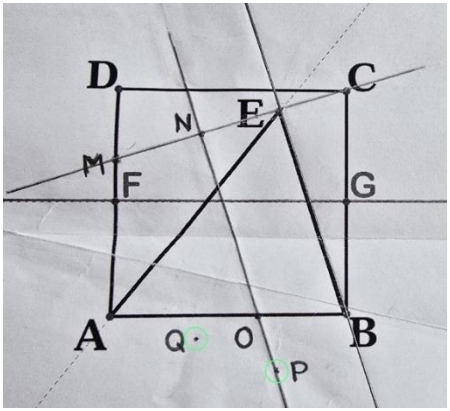


Figura 34 Proyecciones obtenidas. Fuente propia.

Los pliegues que conectan estos puntos con el punto E trisecan el ángulo ∡BEA tal como se ilustra en la Figura 35.

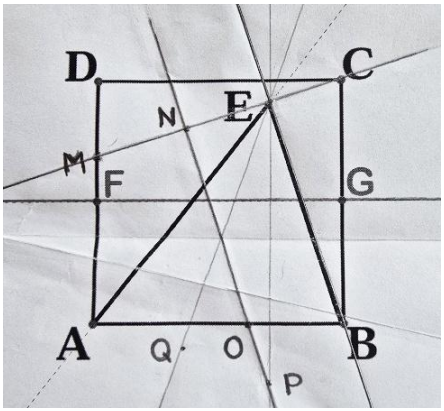


Figura 35 Trisección del ángulo ∡BEA. Fuente propia.

La Figura 36 ilustra los pliegues realizados para la trisección de los tres ángulos del triángulo △ABE, así como los puntos de intersección de los pliegues trisectores adyacentes entre ángulos. Estos puntos forman el Triángulo de Morley.

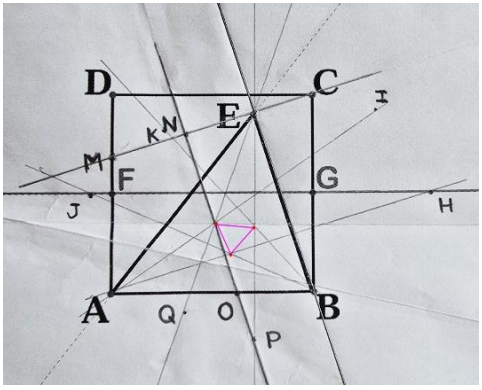


Figura 36 Triángulo de Morley. Fuente propia.

3. Creación del Rompecabezas del triángulo de Morley

En este capítulo abordaremos el desarrollo de un rompecabezas interactivo que tiene como fundamento la demostración del **teorema de Morley**, que se encuentra en Conway (2014). Este teorema es un resultado fascinante de la geometría clásica que, por su naturaleza, invita a la exploración tanto teórica como práctica. El proceso de creación de un rompecabezas geométrico no solo permite profundizar en la comprensión del teorema, sino también aplicar conceptos fundamentales en el ámbito del diseño gráfico y el desarrollo de software.

A lo largo de este capítulo, exploraremos las diferencias, ventajas y desventajas de los **mapas de bits y gráficos vectoriales**, tecnologías esenciales en la representación visual del triángulo de Morley, que permiten una manipulación eficiente de la imagen en diferentes plataformas. También analizaremos el **software de desarrollo** adecuado para la creación de este tipo de aplicaciones, seleccionando herramientas que no solo faciliten la creación de gráficos, sino que también brinden control sobre las interacciones dinámicas del usuario.

En la sección dedicada a la **metodología de desarrollo**, definiremos el enfoque paso a paso para construir el rompecabezas. Consideraremos tanto los aspectos técnicos como pedagógicos, con el objetivo de crear una herramienta intuitiva que invite al usuario a descubrir las propiedades del triángulo de Morley de manera lúdica y educativa.

Finalmente, profundizaremos en la **construcción del rompecabezas** en sí mismo, explicando cómo cada pieza y movimiento contribuyen a una experiencia interactiva que refuerza la comprensión del teorema. Este proceso será acompañado de ejemplos gráficos y una guía detallada sobre su implementación, integrando los conocimientos previos con los nuevos conceptos aprendidos.

Este capítulo busca combinar la belleza de la geometría con las posibilidades de la tecnología digital, proporcionando una herramienta tanto educativa como recreativa para explorar uno de los teoremas más interesantes de la geometría clásica.

3.1 MAPAS DE BITS Y GRÁFICOS VECTORIALES

En el mundo del diseño digital y la creación de imágenes, los **mapas de bits** y los **gráficos vectoriales** juegan un papel fundamental al ofrecer dos enfoques distintos para representar visualmente objetos. Ambos formatos tienen características, ventajas y desventajas que los hacen más adecuados para diferentes tipos de aplicaciones, especialmente cuando se trata de desarrollar un rompecabezas como el del **triángulo de Morley**.

3.1.1 DEFINICIONES

De manera sencilla, podemos definir estos dos objetos computacionales de la siguiente forma:

- **Mapas de bits**: También conocidos como **imágenes rasterizadas**, son representaciones de imágenes mediante una cuadrícula o malla de píxeles, donde cada píxel tiene un valor de color asignado. Los mapas de bits son adecuados para imágenes complejas que contienen muchos detalles y transiciones de color suaves, como fotografías.
- **Gráficos vectoriales**: En contraste, los gráficos vectoriales se basan en **ecuaciones matemáticas** para describir líneas, curvas y formas. Esto los hace ideales para representaciones geométricas, como diagramas o figuras geométricas, ya que son escalables sin perder calidad.

3.1.2 DIFERENCIAS CLAVE

Los mapas de bits y los gráficos vectoriales tienen diferencias fundamentales en varios aspectos. Estas diferencias hacen que ambos tipos de gráficos sean útiles en distintos contextos, dependiendo de las necesidades del proyecto visual.

1. **Escalabilidad**:
    - Los **mapas de bits** no son escalables sin perder calidad; al ampliarlos, los píxeles se distorsionan porque se recalculan, lo que da lugar a una imagen borrosa o pixelada.
    - Los **gráficos vectoriales** son totalmente escalables, lo que significa que pueden ampliarse o reducirse indefinidamente

sin perder calidad, ya que las formas se redibujan matemáticamente.

2. **Tamaño del archivo**:
    - Los **mapas de bits** suelen generar archivos de mayor tamaño, especialmente para imágenes grandes y detalladas, ya que se requiere almacenar información de cada píxel individual.
    - Los **gráficos vectoriales**, al depender de ecuaciones para describir las formas, suelen ocupar menos espacio de almacenamiento para imágenes simples.
3. **Aplicaciones**:
    - Los **mapas de bits** son ideales para representaciones ricas en detalles y texturas, como las fotos y las ilustraciones complejas.
    - Los **gráficos vectoriales** se utilizan principalmente en logotipos, diagramas, tipografías y, en nuestro caso, la representación precisa de figuras geométricas como el **triángulo de Morley**.

3.1.3 VENTAJAS Y DESVENTAJAS

En esta sección exploraremos las ventajas y desventajas de los **mapas de bits** y los **gráficos vectoriales**, dos formatos ampliamente utilizados en

el desarrollo gráfico digital. Ambos presentan características que los hacen más adecuados para diferentes aplicaciones. Mientras que los **mapas de bits** destacan por su capacidad para manejar detalles complejos y transiciones de color suaves, los **gráficos vectoriales** sobresalen por su escalabilidad infinita y eficiencia en el almacenamiento. Sin embargo, cada uno tiene limitaciones específicas que deben considerarse al elegir el formato adecuado para un proyecto, especialmente en contextos como la creación de rompecabezas o representaciones geométricas. A continuación, evaluaremos estos aspectos en función de su rendimiento en diversas aplicaciones.

- **Mapas de bits**:
    - ***Ventajas*:** Son más adecuados para imágenes complejas, como fotos o ilustraciones que necesitan manejar texturas y sombras realistas.
    - ***Desventajas*:** Pierden calidad al escalarse y generan archivos grandes cuando se trata de imágenes detalladas.
- **Gráficos vectoriales**:
    - ***Ventajas*:** Perfectos para ilustraciones geométricas y logotipos, permiten un escalado infinito sin pérdida de calidad y suelen ocupar menos espacio para imágenes simples.
    - ***Desventajas*:** No son aptos para imágenes complejas con muchos detalles o transiciones de color, ya que las

ecuaciones matemáticas que los describen no pueden capturar estas sutilezas.

En el rompecabezas del triángulo de Morley, se eligió utilizar gráficos vectoriales debido a varias razones clave que se alinean tanto con las necesidades geométricas del proyecto como con la experiencia de usuario.

Los gráficos vectoriales son ideales para representar figuras geométricas como el triángulo de Morley, ya que permiten una precisión matemática en la representación de líneas, ángulos y formas. El teorema de Morley implica construcciones geométricas que deben conservar su exactitud independientemente del tamaño o la resolución del dispositivo. Estos gráficos se basan en ecuaciones matemáticas, lo que permite que el triángulo de Morley y otras formas se escalen sin pérdida de calidad o distorsión, una característica crucial para mantener la integridad geométrica del rompecabezas.

Otro aspecto importante es la interacción del usuario. En el rompecabezas, el usuario necesita mover, rotar o ajustar ángulos las piezas del triángulo. Los gráficos vectoriales permiten una manipulación suave y precisa de estas piezas, ya que las formas se pueden redibujar en tiempo real sin problemas de pixelación o pérdida de detalle, algo que no sería posible con mapas de bits.

En cuanto al rendimiento, los gráficos vectoriales suelen ocupar menos espacio en memoria que los mapas de bits cuando se trata de ilustraciones simples, como es el caso del rompecabezas geométrico. Esto garantiza una

mejor eficiencia en el procesamiento y carga de las piezas del rompecabezas, especialmente en dispositivos con recursos limitados.

Por estas razones, los gráficos vectoriales resultan ser la mejor opción para el desarrollo del rompecabezas del Triángulo de Morley, proporcionando una representación precisa, escalable y una interacción fluida para el usuario.

3.2 SOFTWARE DE DESARROLLO

El desarrollo de aplicaciones para diferentes plataformas requiere herramientas especializadas que se adapten a las necesidades específicas del entorno en el que se trabajará. En este apartado, compararemos tres plataformas de desarrollo que se utilizan comúnmente para crear aplicaciones de escritorio e interactivas: **Visual Studio** con lenguaje **C#**, **Android Studio** con lenguaje **Java**, y **Adobe Animate** con lenguaje **ActionScript 3**. Cada una de estas herramientas presenta ventajas y desventajas en términos de funcionalidad, versatilidad y el tipo de aplicaciones que pueden desarrollar.

3.2.1 VISUAL STUDIO CON C#

Visual Studio es una de las plataformas de desarrollo más completas y robustas para crear aplicaciones de escritorio y web utilizando el lenguaje **C#** y otras tecnologías del entorno **.NET**. C# es un lenguaje orientado a objetos que ofrece una gran eficiencia en términos de rendimiento y escalabilidad.

- **Ventajas**:
    - **IDE poderoso**: Visual Studio ofrece herramientas avanzadas de depuración, diseño de interfaces gráficas y administración de proyectos.
    - **Compatibilidad con .NET**: permite aprovechar todo el ecosistema de .NET para desarrollar aplicaciones complejas y robustas.
    - **Soporte multiplataforma**: Con el framework **.NET Core**, las aplicaciones pueden ser ejecutadas en Windows, Linux y macOS.
    - **Herramientas de UI**: Facilita la creación de interfaces de usuario avanzadas utilizando tecnologías como **Windows Forms** y **WPF**.
- **Desventajas**:
    - **Curva de aprendizaje**: Para proyectos simples, puede resultar excesivamente complejo.
    - **Requerimientos de hardware**: Visual Studio consume bastantes recursos del sistema, lo que puede ser un inconveniente en equipos con hardware limitado.

3.2.2 ANDROID STUDIO CON JAVA

Android Studio es la plataforma oficial para el desarrollo de aplicaciones para el sistema operativo Android, y utiliza principalmente **Java** como lenguaje de programación (aunque también es compatible con **Kotlin**). Está optimizada para crear aplicaciones de escritorio para móviles.

- **Ventajas**:
    - **Enfoque móvil**: Android Studio está diseñado específicamente para aplicaciones móviles, lo que lo convierte en una excelente opción para desarrollar aplicaciones destinadas a smartphones o tablets android.
    - **Herramientas de emulación**: Ofrece un **emulador Android** para probar las aplicaciones directamente sin necesidad de un dispositivo físico.
    - **Flexibilidad del lenguaje Java**: Java es un lenguaje muy utilizado y robusto, con una enorme comunidad de soporte.
    - **Integración con Google Services**: Facilita la integración de servicios como Google Maps, Firebase, y otros.
- **Desventajas**:
    - **Alto consumo de recursos**: Android Studio puede ser muy exigente en términos de uso de CPU y RAM.

- **Limitado para escritorio**: Aunque es posible desarrollar aplicaciones de escritorio para computadores en general, Android Studio no está diseñado para ello, por lo que no es la opción ideal para aplicaciones no móviles.

3.2.3 ADOBE ANIMATE CON ACTIONSCRIPT 3

Adobe Animate es una herramienta multiplataforma centrada en la creación de contenido interactivo, animaciones y juegos, utilizando **ActionScript 3** como lenguaje de programación para aplicaciones de escritorio y HTML5 y JavaScript para aplicaciones Web. Aunque su uso ha disminuido en favor de otras plataformas, sigue siendo relevante para el desarrollo de aplicaciones interactivas en entornos controlados.

- **Ventajas**:
  - **Enfoque visual**: Adobe Animate está diseñado para facilitar la creación de animaciones y gráficos mediante herramientas visuales, lo que lo convierte en una opción ideal para proyectos que requieren un alto grado de interactividad y animación.
  - **ActionScript 3**: Aunque no es tan popular como otros lenguajes, **ActionScript 3** es potente y ofrece un control detallado sobre las animaciones y las interacciones.
  - **Exportación a múltiples formatos**: Adobe Animate permite exportar contenido a HTML5, SWF, y otros

formatos, lo que facilita la implementación en diferentes sistemas operativos.

- **Desventajas**:
    - **Enfoque limitado**: Aunque es excelente para proyectos interactivos y animados, **Adobe Animate** no es la herramienta más adecuada para el desarrollo de aplicaciones complejas ya que sus aplicaciones producidas no son nativas para los diferentes sistemas operativos.

La elección del software de desarrollo depende en gran medida de las características del proyecto que se desea implementar. Para aplicaciones de escritorio robustas y con una amplia funcionalidad, **Visual Studio** con **C#** ofrece una solución altamente eficaz y poderosa, pero es más complicado a la hora de tratar gráficos vectoriales, aunque puede producir apps multiplataforma. Si el enfoque es el desarrollo móvil, **Android Studio** con **Java** se convierte en la opción más conveniente para dispositivos Android. Por otro lado, para aplicaciones interactivas y animadas, **Adobe Animate** con **ActionScript 3** proporciona las herramientas necesarias para desarrollar proyectos centrados en la experiencia visual, el diseño gráfico y la interacción con el usuario, por tanto, esta es la plataforma elegida para este proyecto.

3.3 METODOLOGÍA DE DESARROLLO

El desarrollo del **rompecabezas del triángulo de Morley** se basó, desde el punto de vista matemático, en la demostración de Jhon Conway de

este resultado y desde el punto de vista técnico, en la metodología **ADDIE**, un enfoque estructurado y ampliamente utilizado en proyectos educativos. El modelo ADDIE se compone de cinco fases: **Análisis**, **Diseño**, **Desarrollo**, **Implementación** y **Evaluación**, cada una con un rol específico en la creación de contenidos efectivos y con un enfoque continuo en la mejora.

Fase 1: Análisis

La fase de **Análisis** es el punto de partida, donde se identifican las necesidades del proyecto y se determinan los objetivos principales. En este caso, el objetivo fue crear una herramienta interactiva que permita a los usuarios explorar el **teorema de Morley** de manera visual e intuitiva. Según Branch (2009), la fase de análisis implica definir los requisitos del proyecto y las necesidades de los usuarios, permitiendo establecer una base sólida para las fases posteriores (p. 34).

Fase 2: Diseño

En la fase de **Diseño**, se trazaron las bases de la estructura del rompecabezas y se decidieron las herramientas visuales a utilizar, como los gráficos vectoriales, que ofrecen precisión y escalabilidad, algo necesario para una representación geométrica como el triángulo de Morley. Molenda (2003) explica que el diseño instruccional debe tener en cuenta no solo los objetivos del contenido, sino también las características técnicas que faciliten la interacción con el usuario.

Fase 3: Desarrollo

Durante la fase de **Desarrollo**, se llevó a cabo la creación del rompecabezas utilizando herramientas como **Adobe Animate** y **ActionScript 3**, integrando los gráficos y las funcionalidades interactivas planificadas en la fase de diseño. Branch (2009) señala que esta fase consiste en implementar las estrategias planificadas, asegurando que el producto final esté alineado con los objetivos educativos establecidos.

Fase 4: Implementación

En la fase de **Implementación**, el rompecabezas se puso a prueba en un entorno de prueba controlado. Esto permitió obtener feedback valioso de los usuarios antes de su despliegue final. Reiser y Dempsey (2017) enfatizan que, en esta fase, además de implementar el producto, es esencial preparar a los usuarios para interactuar de manera efectiva con la herramienta.

Fase 5: Evaluación

Finalmente, la fase de **Evaluación** consistió en analizar el feedback recopilado durante la implementación y realizar los ajustes necesarios para optimizar la experiencia del usuario. Branch (2009) sugiere que la evaluación es un proceso continuo que se lleva a cabo tanto durante el desarrollo como al final del proyecto para asegurar que se cumplan los objetivos del aprendizaje (p. 63).

El modelo **ADDIE** proporcionó una estructura clara y organizada para el desarrollo del rompecabezas del triángulo de Morley, garantizando que

cada fase contribuyera a un producto final eficiente y educativo. La naturaleza iterativa de ADDIE permitió mejorar continuamente el rompecabezas a medida que se avanzaba, asegurando una experiencia óptima para los usuarios.

3.4 ROMPECABEZAS DEL TRIÁNGULO DE MORLEY

Adobe Animate es una herramienta poderosa para la creación de animaciones interactivas y aplicaciones multimedia, y **ActionScript 3.0 (AS3)** es el lenguaje que se utiliza para agregar interactividad a esos proyectos. Empezar un proyecto con AS3 en Adobe Animate implica seguir una serie de pasos para configurar correctamente el entorno y establecer las bases del proyecto.

3.4.1 PASOS PARA INICIAR UN PROYECTO

El proyecto se inició con las siguientes configuraciones:

1. **Nombre de archivo**: “TrianguloMorley”.
2. **Destino de la producción**: AIR 51 for Desktop
3. **Configurar el escenario**: 1280 x 720 pixeles a 15 cuadros por segundo. Se habilitan para la aplicación cuatro fotogramas así:

- **Fotograma 1:** Corresponde a los elementos de la portada de presentación de la aplicación. La Figura 37 ilustra dicha portada.

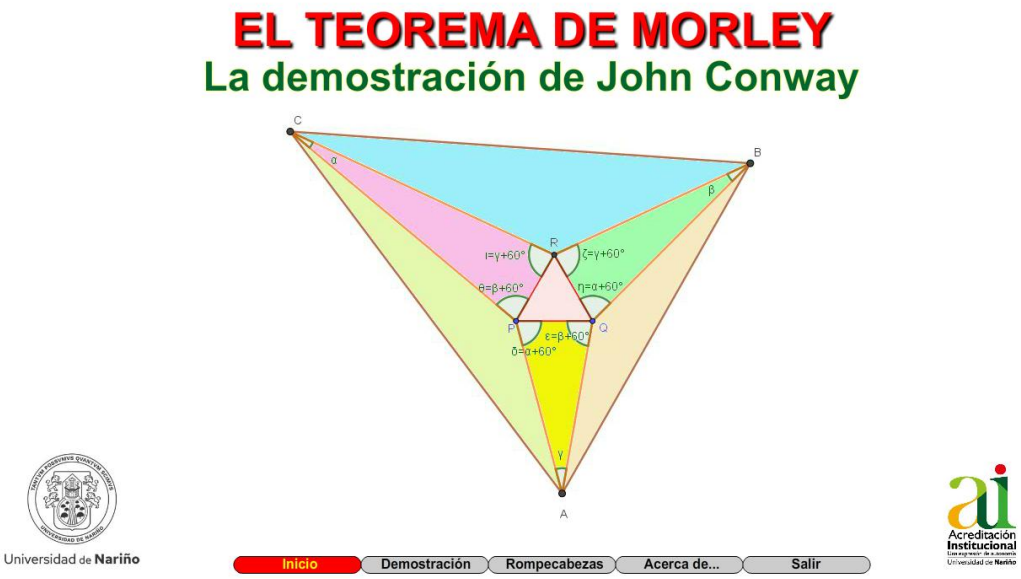


Figura 37 Pantalla de presentación de la aplicación. Fuente propia.

- **Fotograma 2:** Se utiliza para mostrar el documento que contiene la demostración del Teorema de Morley, realizada por John Conway. El visor permite navegar por el documento mediante botones de control. La Figura 38 muestra el contenido de este fotograma.

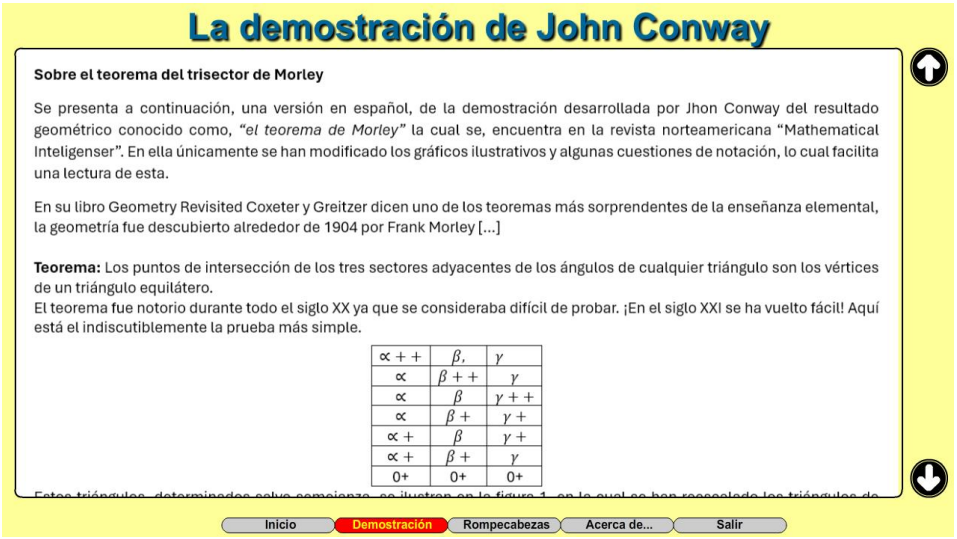


Figura 38 Pantalla de visualización de la Demostración de Conway. Fuente propia.

Fotograma 3: Es la pantalla principal del rompecabezas. En la Figura 39 se ilustra la ejecución del armado de un triángulo de Morley para la versión que corre en **Windows** y **Linux** bajo **Wine**.

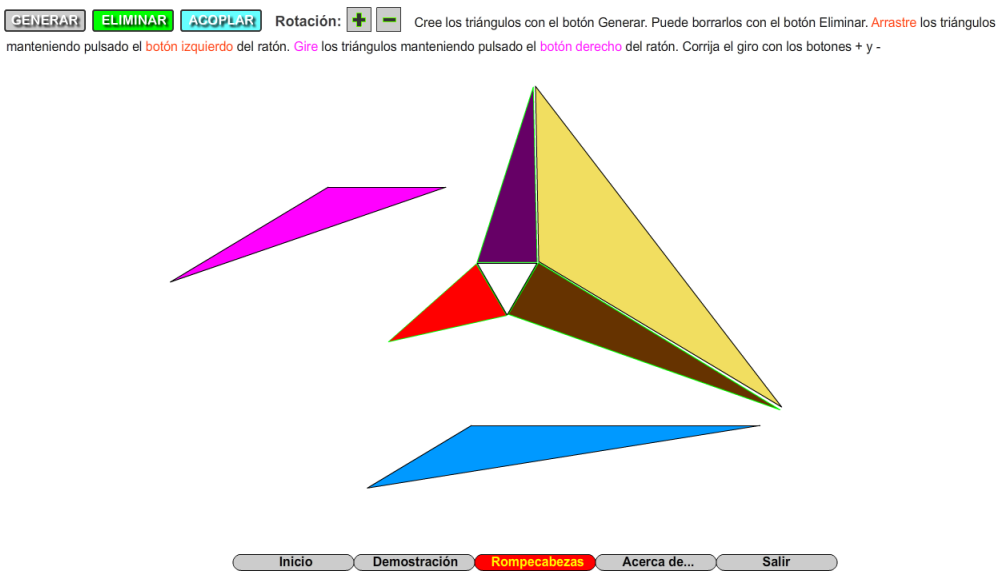


Figura 39 Armado del rompecabezas. Fuente propia.

La versión de la aplicación para **Android**, cambia ligeramente, ya que, en los dispositivos móviles que funcionan con este sistema operativo, no existe el ratón ni menos el evento click con el botón derecho, usado para girar los triángulos en **Windows** y **Linux**. Por esta razón, la interfaz cambia agregando un par de botones para hacer giros con incrementos o decrementos de 10 grados. La Figura 40 muestra la interfaz para **Android**.

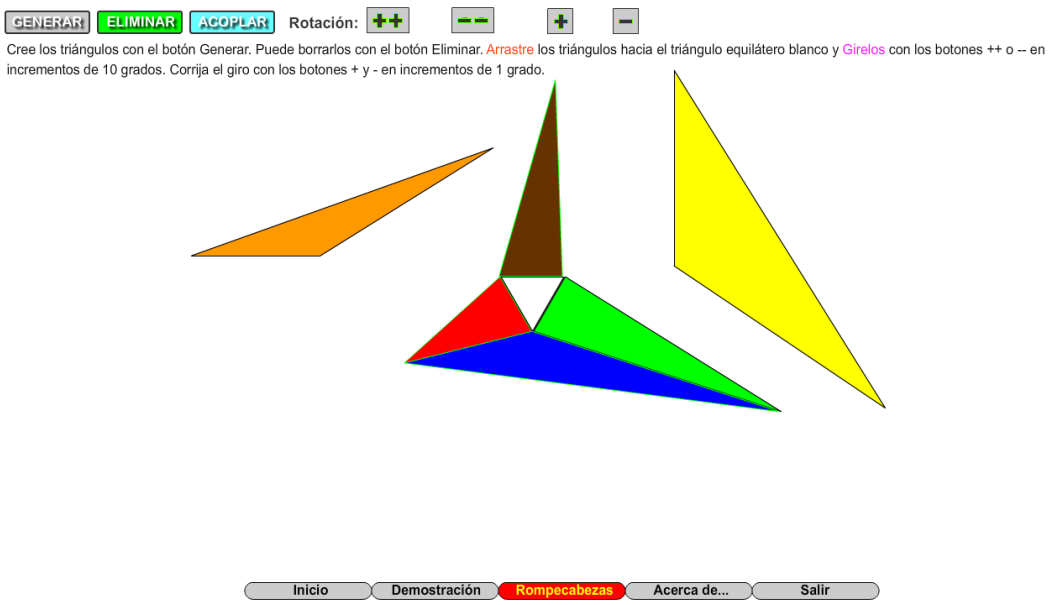


Figura 40 Versión para Android. Fuente propia.

La Figura 41 muestra el resultado final tras haber utilizado el botón **Acoplar**, que permite formar automáticamente el triángulo.

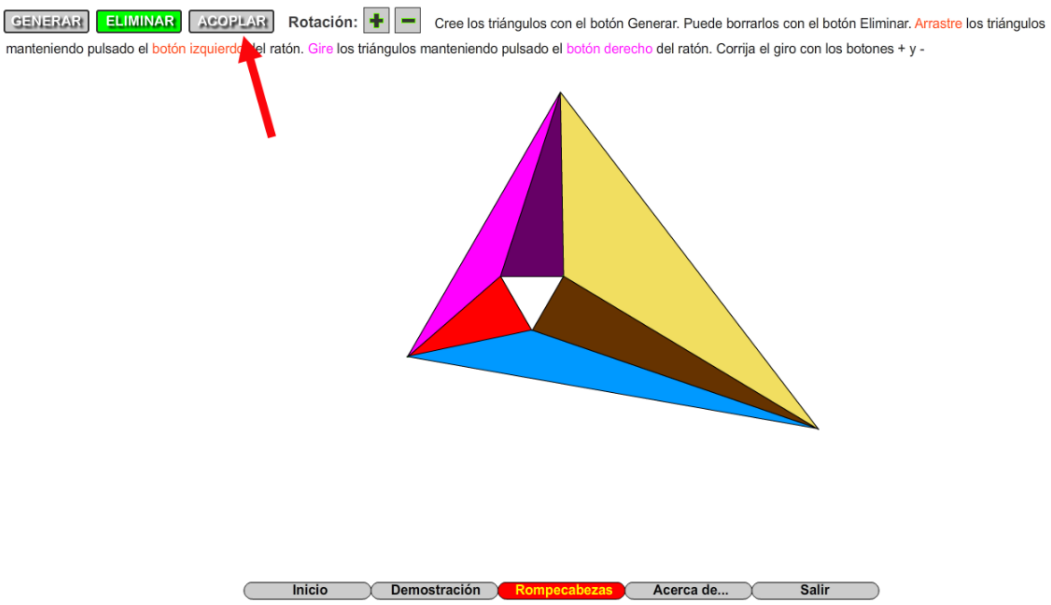


Figura 41 Uso del botón Acoplar. Fuente propia.

- **Fotograma 4:** Pantalla de presentación de los autores, que incluye un botón de enlace a un video demostrativo sobre el uso de la aplicación. Ver Figura 42.

Figura 42 Sección Acerca de. Fuente propia.

4. **Gráficos en mapas de bits**: Logotipos de la Universidad de Nariño, Gráfica PNG de un triángulo de Morley, Gráfica de la demostración del teorema por John Conway, Fotografía de la portada de la Universidad de Nariño.

5. **Elementos gráficos vectoriales**: Le generan en tiempo de ejecución mediante algoritmos que serán descritos más adelante.

6. **Importación de librerías necesarias**: Debido a que los gráficos de los triángulos se realizarán en tiempo de ejecución, y que deben ser manipulados por el usuario y/o máquina, las librerías que se deben importar son las siguientes:

```
import flash.display.MovieClip;
import flash.events.MouseEvent;
import fl.transitions.Tween;
import fl.transitions.easing.*;
import fl.transitions.TweenEvent;
```

3.4.2 DECLARACIÓN DE VARIABLES Y OBJETOS

Las variables utilizadas en la programación del rompecabezas son las siguientes:

1. **Ángulos en grados:** Variables enteras **int.** a, b, c, d, e. s, ii, o, q, r y p que se pueden observar en la Figura 43.

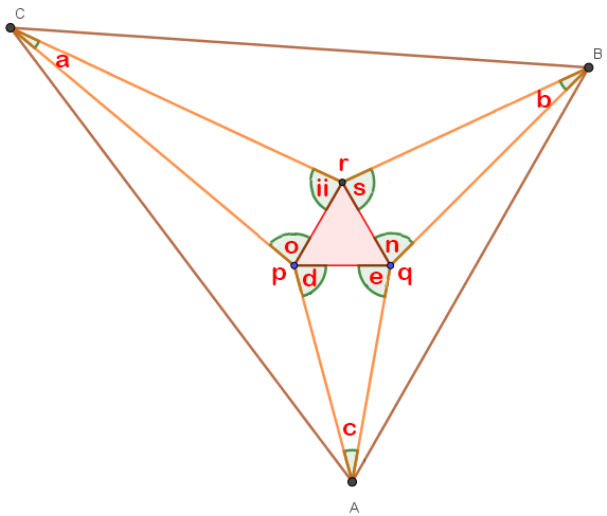


Figura 43 Ángulos para la programación. Fuente propia.

2. **Ángulos en radianes:** Variables reales **Number** radA, radB, radC, radD, radE, radS, radII, radO, radQ, radR y radP que corresponden a los mismos ángulos.

3. **Segmentos:** Variables reales Number PC, RC, RB, QB, PA y QA como se describe en la Figura 44.

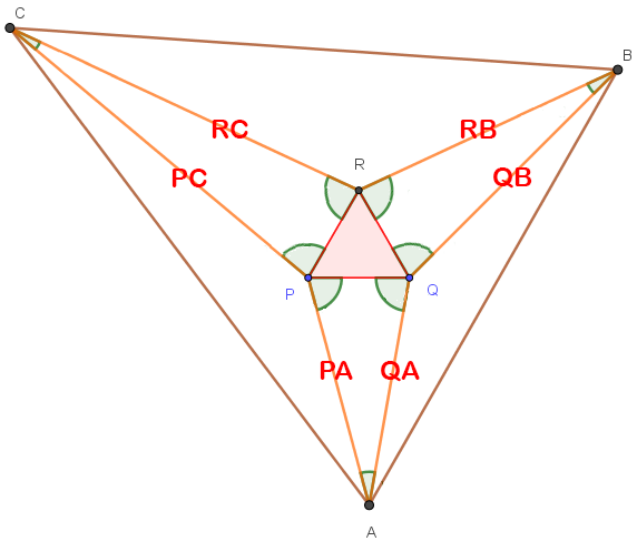


Figura 44 Segmentos para la programación. Fuente propia.

4. **Puntos:** Variables de tipo **Point** P, Q, R mostrados en la figura anterior.

Objetos: variables de tipo **Sprite triángulo1**, **triángulo2**, **triángulo3**, **triángulo4**, **triángulo5**, **triángulo6** y **triánguloEquilatero**. Estos objetos son para calcular y dibujar los siete triángulos del Teorema de Morley y poderlos manipular gráficamente.

Colores: Se declaró una matriz bidimensional llamada **Colores** donde se expresan en datos hexadecimales, 16 parejas de colores contrastantes de relleno y borde para dibujar los triángulos. Las parejas de colores son:

blanco-negro, amarillo-negro, verdeClaro-negro, cyan-negro, fucsia-negro, naranja-negro, rosa-negro, grisClaro-negro, beige-negro, negro-verdeClaro, marron-verdeClaro, grisOscuro-verdeClaro, azul-verdeClaro, rojo-verdeClaro, verdeOscuro-verdeClaro, morado-verdeClaro. El **Array** declarado es:

```
var Colores:Array = [
[0xFFFF00,0x000000],
[0x00FF00,0x000000],
[0x0099FF,0x000000],
[0xFF00FF,0x000000],
[0xFF9900,0x000000],
[0xFFCCCC,0x000000],
[0xCCCCCC,0x000000],
[0xF1DE60,0x000000],
[0x000000,0x00FF00],
[0x663300,0x00FF00],
[0x666666,0x00FF00],
[0x0000FF,0x00FF00],
[0xFF0000,0x00FF00],
[0x006666,0x00FF00],
[0x660066,0x00FF00]
];
```

Posiciones: son datos de puntos donde se ubicarán inicialmente los **Sprites** de los triángulos alrededor del triángulo equilátero central. Estos datos van en una matriz bidimensional llamada **Posiciones**. Las posiciones son fijas pero la distribución de los **Sprites** en ellas es pseudoaleatoria. El **Array** Posiciones se declara como sigue:

```
var Posiciones:Array = [
[400,100],
[600,100],
[900,100],
[400,250],
[600,250],
[900,250]
];
```

Variables adicionales: Como los colores y la posición es pseudoaleatoria para cada triángulo a excepción del equilátero, se deben calcular 6 índices para tomar elementos del array **Colores** y 6 del array **Posiciones**. Por tanto, se deben calcular dos permutaciones de 6 elementos, una para **Colores** llamada **indicesColores** y otra de 6 elementos para las posiciones llamada **indicesPosiciones**, ambas de tipo **Vector.<uint>**.

Para individualizar colores se tienen las variables **uint** llamadas **colorRelleno** y **colorLinea**. De cada triángulo se conocerá un lado y los ángulos adyacentes a ese lado, por tanto, es necesario calcular el tercer vértice, y para recibir ese dato se crea una variable de tipo **Point tercerVertice**.

Se establece la longitud del lado del triángulo equilátero central en la variable **int L**.

Debido a que se deben generar permutaciones de índices, se utilizará la siguiente función que calcula para un grupo de **n** índices del 0 al n-1, una permutación de **K** elementos:

```
function permutacion(numElementos:uint, tamañoPermutacion:uint): Vector.<uint>
{
    var vect:Vector.<uint> = new Vector.<uint>(tamañoPermutacion);
    var nuevo;
    var x:uint = 0;
    var i:uint;
    vect[0] = Math.random() * numElementos;
    for (i = 0; i < tamañoPermutacion; i++)
    {
        nuevo = true;
        while (nuevo)
        {
            nuevo = false;
            x = Math.random() * numElementos;
            for (var j = 0; j <= i - 1; j++)
            {
                if (x == vect[j])
                {
                    nuevo = true;
                    break;
                }
            }
        }
        vect[i] = x;
    }
    return vect;
}
```

5. **Botones del rompecabezas:** Estos objetos se crean como **Clips de Película** en lugar de botones, con el propósito de alternar entre un estado activo e inactivo mediante el cambio de color en dos fotogramas distintos, tal como se muestra en la Figura 45.

Figura 45 Clip de película como botón. Fuente propia.

Los clips de película se denominan **btnGenerar**, **btnEliminar** y **btnAcoplar**. Dado que no son objetos de tipo botón, es necesario

escribir código que les otorgue el comportamiento típico de un botón, incluyendo la aparición del cursor de mano (Hand) cuando el ratón pasa sobre ellos. A continuación, se muestra el código ejemplo que implementa esta funcionalidad para uno de estos clips.

```
btnGenerar.buttonMode = true;
btnGenerar.useHandCursor = true;
btnGenerar.mouseChildren = false;
```

6. **Clip Menú:** Consta de cinco botones que hacen la navegación por la película principal, llevando al usuario al fotograma adecuado. Al igual que en el caso descrito en el numeral anterior, estos no son botones reales, sino clips de película, lo que permite destacar solo uno de ellos para indicar visualmente la sección de la aplicación en la que se encuentra el usuario. La Figura 46 muestra los elementos contenidos en este clip.

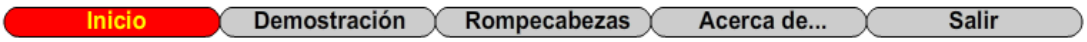


Figura 46 Menú principal. Fuente propia.

3.4.3 ALGUNAS CONSIDERACIONES EN LA PROGRAMACIÓN

La función **generarTriangulos** es un método asociado al evento **MOUSE_CLICK** del botón **btnGenerar**. Comienza configurando los botones **btnGenerar**, **btnEliminar** y **btnAcoplar**. Luego, genera permutaciones de 6 índices de colores y 6 índices de posiciones. El ángulo "**a**" se genera pseudoaleatoriamente como un número entero entre 10 y 18 grados, y el ángulo "**b**" entre 10 y 19 grados. Los ángulos restantes se calculan utilizando fórmulas específicas. A continuación, dentro de la función, los ángulos calculados se convierten a radianes para ser utilizados en funciones trigonométricas. Para los triángulos que comparten lados con el triángulo equilátero, el tercer vértice se determina mediante las fórmulas detalladas en la Figura 47.

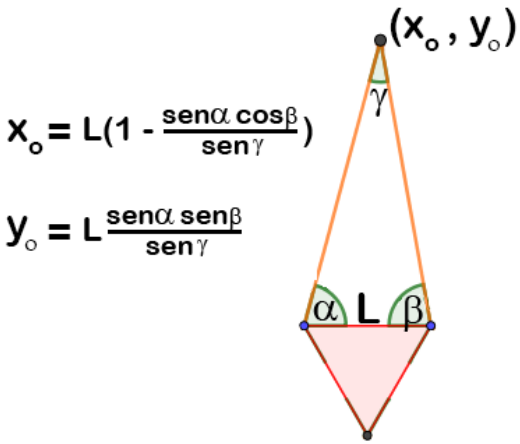


Figura 47 Coordenadas del tercer vértice. Fuente propia.

El método queda programado de la siguiente manera:

```
function calcularTercerVertice(base:Number, alpha:Number, beta:Number):Point {
    // Convertimos los ángulos a radianes
    var radAlpha:Number = alpha * Math.PI / 180;
    var radBeta:Number = beta * Math.PI / 180;

    // Calculamos el ángulo gamma (ángulo opuesto a la base)
    var radGamma:Number = Math.PI - radAlpha - radBeta;

    // Calculamos la longitud de los lados a y b
    var ladoA:Number = base * Math.sin(radAlpha) / Math.sin(radGamma);
    var ladoB:Number = base * Math.sin(radBeta) / Math.sin(radGamma);
    // Calculamos las coordenadas del tercer vértice (C)
    var xC:Number = base * (1 - Math.sin(radAlpha) * Math.cos(radBeta) / Math.sin(radGamma));
    var yC:Number = base * Math.sin(radAlpha) * Math.sin(radBeta)/Math.sin(radGamma);

    // Creamos un objeto Point para representar el tercer vértice
    var tercerVertice:Point = new Point(xC, yC);

    return tercerVertice;
}
```

Para los tres triángulos restantes, el cálculo del tercer vértice se realiza utilizando las fórmulas mostradas en la Figura 48, teniendo en cuenta que uno de sus lados se encuentra sobre el eje "**x**", con un vértice ubicado en el origen del sistema de coordenadas.

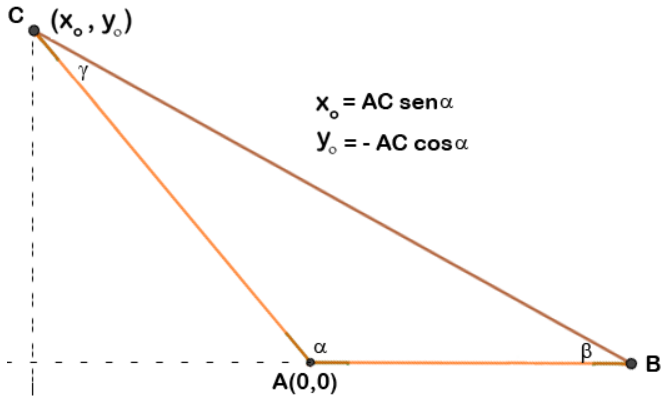


Figura 48 Tercer vértice para los triángulos obtusángulos. Fuente propia.

Finalmente, el método **generarTriangulos** procede a crear cada triángulo, asignándole el color y la posición correspondientes de acuerdo con las permutaciones de los índices generados previamente, y luego los agrega al escenario. Este método está vinculado al evento **MOUSE_CLICK** del botón **btnGenerar**.

```
function generarTriangulos(event:MouseEvent):void {
    btnGenerar.gotoAndStop(2);
    btnGenerar.removeEventListener(MouseEvent.CLICK, generarTriangulos);
    btnEliminar.gotoAndStop(1);
    btnEliminar.addEventListener(MouseEvent.CLICK, eliminarTriangulos);
    btnAcoplar.gotoAndStop(1);
    btnAcoplar.addEventListener(MouseEvent.CLICK, acoplarTriangulos);
    indicesColores = permutacion(Colores.length, 6);//6 índices para seleccionar parejas de colores
    indicesPosiciones = permutacion(6, 6);
    a = 10 + Math.floor(Math.random() * 9);
    b = 10 + Math.floor(Math.random() * 10);
    c = 60 - a - b;
    d = a + 60;
    e = 180 - c - d;
    n = a + 60;
    s = c + 60;
    ii = c + 60;
    o = b + 60;
    q = 300 - e - n;
    r = 300 - ii - s;
    p = 300 - d - o;
    radA = a * Math.PI / 180;
    radB = b * Math.PI / 180;
    radC = c * Math.PI / 180;
    radD = d * Math.PI / 180;
    radE = e * Math.PI / 180;
    radN = n * Math.PI / 180;
    radS = s * Math.PI / 180;
    radII = ii * Math.PI / 180;
    radO = o * Math.PI / 180;
    radQ = q * Math.PI / 180;
    radR = r * Math.PI / 180;
    radP = p * Math.PI / 180;

    //TRIÁNGULO 1
    colorRelleno = Colores[indicesColores[0]][0];
    colorLinea = Colores[indicesColores[0]][1];
```

```
tercerVertice = calcularTercerVertice(L, e, d);
triangulo1 = crearTrianguloArrastrableYrotable(0, 0, L, 0, tercerVertice.x, tercerVertice.y,
colorLinea, colorRelleno);
addChild(triangulo1);  // Agregar el triángulo a la pantalla
triangulo1.x = Posiciones[indicesPosiciones[0]][0];
triangulo1.y = Posiciones[indicesPosiciones[0]][1];

//TRIÁNGULO 2
colorRelleno = Colores[indicesColores[1]][0];
colorLinea = Colores[indicesColores[1]][1];
tercerVertice = calcularTercerVertice(L, s, n);
triangulo2 = crearTrianguloArrastrableYrotable(0, 0, L, 0, tercerVertice.x, tercerVertice.y,
colorLinea, colorRelleno);
addChild(triangulo2);  // Agregar el triángulo a la pantalla
triangulo2.x = Posiciones[indicesPosiciones[1]][0];
triangulo2.y = Posiciones[indicesPosiciones[1]][1];

//TRIÁNGULO 3
colorRelleno = Colores[indicesColores[2]][0];
colorLinea = Colores[indicesColores[2]][1];
tercerVertice = calcularTercerVertice(L, o, ii);
triangulo3 = crearTrianguloArrastrableYrotable(0, 0, L, 0, tercerVertice.x, tercerVertice.y,
colorLinea, colorRelleno);
addChild(triangulo3);  // Agregar el triángulo a la pantalla
triangulo3.x = Posiciones[indicesPosiciones[2]][0];
triangulo3.y = Posiciones[indicesPosiciones[2]][1];

//TRIÁNGULO 4
colorRelleno = Colores[indicesColores[3]][0];
colorLinea = Colores[indicesColores[3]][1];
QB = L * Math.sin(radS) / Math.sin(radB);
QA = L * Math.sin(radD) / Math.sin(radC);
tercerVertice.x = QB * Math.cos(radQ);
tercerVertice.y = QB * Math.sin(radQ);
triangulo4 = crearTrianguloArrastrableYrotable(0, 0, QA, 0, tercerVertice.x, tercerVertice.y,
colorLinea, colorRelleno);
addChild(triangulo4);  // Agregar el triángulo a la pantalla
triangulo4.x = Posiciones[indicesPosiciones[3]][0];
triangulo4.y = Posiciones[indicesPosiciones[3]][1];

//TRIÁNGULO 5
colorRelleno = Colores[indicesColores[4]][0];
colorLinea = Colores[indicesColores[4]][1];
RB = L * Math.sin(radN) / Math.sin(radB);
RC = L * Math.sin(radO) / Math.sin(radA);
tercerVertice.x = RC * Math.cos(radR);
tercerVertice.y = RC * Math.sin(radR);
triangulo5 = crearTrianguloArrastrableYrotable(0, 0, RB, 0, tercerVertice.x, tercerVertice.y,
```

```
    colorLinea, colorRelleno);
    addChild(triangulo5);  // Agregar el triángulo a la pantalla
    triangulo5.x = Posiciones[indicesPosiciones[4]][0];
    triangulo5.y = Posiciones[indicesPosiciones[4]][1];

    //TRIÁNGULO 6
    colorRelleno = Colores[indicesColores[5]][0];
    colorLinea = Colores[indicesColores[5]][1];
    PA = L * Math.sin(radE) / Math.sin(radC);
    PC = L * Math.sin(radII) / Math.sin(radA);
    tercerVertice.x = PA * Math.cos(radP);
    tercerVertice.y = PA * Math.sin(radP);
    triangulo6 = crearTrianguloArrastrableYrotable(0, 0, PC, 0, tercerVertice.x, tercerVertice.y,
    colorLinea, colorRelleno);
    addChild(triangulo6);  // Agregar el triángulo a la pantalla
    triangulo6.x = Posiciones[indicesPosiciones[5]][0];
    triangulo6.y = Posiciones[indicesPosiciones[5]][1];

    //TRIÁNGULO EQUILÁTERO
    colorRelleno = 0xFFFFFF;
    colorLinea = 0x0000000;
    tercerVertice.x = L/2;
    tercerVertice.y = Math.sqrt(3)/2 * L;
    P = new Point(0,0);
    Q = new Point(L,0);
    R = new Point(tercerVertice.x,tercerVertice.y);
    trianguloEquilatero = crearTrianguloArrastrableYrotable(0, 0, L, 0, tercerVertice.x,
    tercerVertice.y, colorLinea, colorRelleno);
    addChild(trianguloEquilatero);  // Agregar el triángulo a la pantalla
    trianguloEquilatero.x = stage.stageWidth / 2 - trianguloEquilatero.width / 2;
    trianguloEquilatero.y = stage.stageHeight / 2 - trianguloEquilatero.height / 2;
    //Elimino susu listeners
    trianguloEquilatero.removeEventListener(MouseEvent.MOUSE_DOWN, iniciaArrastrarTriangulo);
    trianguloEquilatero.removeEventListener(MouseEvent.RIGHT_MOUSE_DOWN, iniciaRotarTriangulo);
    trianguloEquilatero.removeEventListener(MouseEvent.MOUSE_UP, paraArrastrarTriangulo);
    trianguloEquilatero.removeEventListener(MouseEvent.RIGHT_MOUSE_UP, paraRotarTriangulo);
}//Fin del método generarTriangulos
```

La función anterior utiliza un método específico llamado **crearTrianguloArrastrableYrotable** para generar los **Sprites** de los triángulos. Este método no solo crea los **Sprites**, sino que también los convierte en elementos interactivos, permitiendo que se arrastren y roten.

```
function crearTrianguloArrastrableYrotable(x1:Number, y1:Number,
                                           x2:Number, y2:Number,
                                           x3:Number, y3:Number,
                                           lineColor:uint, fillColor:uint):Sprite {
    // Crear un nuevo sprite
    var triangulo:Sprite = new Sprite();

    // Dibujar el triángulo
    triangulo.graphics.lineStyle(3, lineColor);  // Color de los bordes
    triangulo.graphics.beginFill(fillColor);     // Color de relleno
    triangulo.graphics.moveTo(x1, y1);           // Primer punto
    triangulo.graphics.lineTo(x2, y2);           // Segundo punto
    triangulo.graphics.lineTo(x3, y3);           // Tercer punto
    triangulo.graphics.lineTo(x1, y1);           // Cierra el triángulo
    triangulo.graphics.endFill();                // Terminar el relleno

    // Mover los puntos del triángulo relativos al nuevo origen
    triangulo.graphics.clear();
    triangulo.graphics.lineStyle(1, lineColor);
    triangulo.graphics.beginFill(fillColor);
    triangulo.graphics.moveTo(x1 - triangulo.x, y1 - triangulo.y);
    triangulo.graphics.lineTo(x2 - triangulo.x, y2 - triangulo.y);
    triangulo.graphics.lineTo(x3 - triangulo.x, y3 - triangulo.y);
    triangulo.graphics.lineTo(x1 - triangulo.x, y1 - triangulo.y);
    triangulo.graphics.endFill();

    // Hacer que el triángulo sea arrastrable y girable
    triangulo.addEventListener(MouseEvent.MOUSE_DOWN, iniciaArrastrarTriangulo);
    triangulo.addEventListener(MouseEvent.RIGHT_MOUSE_DOWN, iniciaRotarTriangulo);
    triangulo.addEventListener(MouseEvent.MOUSE_UP, paraArrastrarTriangulo);
    triangulo.addEventListener(MouseEvent.RIGHT_MOUSE_UP, paraRotarTriangulo);

    return triangulo;
}
```

Como se observa en la función anterior, se vinculan métodos a los eventos **MOUSE_DOWN**, **MOUSE_UP**, **RIGHT_MOUSE_DOWN** y **RIGHT_MOUSE_UP** de los **Sprites**, para ofrecer al usuario interactividad con estos objetos. A continuación, se describen estos métodos.

```
// Variables para manejar la interacción de arrastrar y rotar
var isDragging:Boolean = false;
var isRotating:Boolean = false;
var lastMouseX:Number;
var lastMouseY:Number;
var lastRotation:Number;
var trianguloActivo:Sprite; //Variable global para mantener el sprite que está rotando o arrastrando
// Función para iniciar el arrastre
function iniciaArrastrarTriangulo(event:MouseEvent):void {
    trianguloActivo = event.currentTarget as Sprite;
    isDragging = true;
    trianguloActivo.startDrag();
    stage.addEventListener(MouseEvent.MOUSE_UP, paraArrastrarTriangulo);
}
// Función para iniciar la rotación
function iniciaRotarTriangulo(event:MouseEvent):void {
    trianguloActivo = event.currentTarget as Sprite;
    isRotating = true;
    lastMouseX = trianguloActivo.stage.mouseX;
    lastMouseY = trianguloActivo.stage.mouseY;
    lastRotation = trianguloActivo.rotation;
    stage.addEventListener(MouseEvent.MOUSE_MOVE, rotateTriangulo);
    stage.addEventListener(MouseEvent.RIGHT_MOUSE_UP, paraRotarTriangulo);
}
// Función para detener el arrastre
function paraArrastrarTriangulo(event:MouseEvent):void {
    if (isDragging) {
        trianguloActivo.stopDrag();
        isDragging = false;
        stage.removeEventListener(MouseEvent.MOUSE_UP, paraArrastrarTriangulo);
    }
}

// Función para detener la rotación
function paraRotarTriangulo(event:MouseEvent):void {
    if (isRotating) {
        stage.removeEventListener(MouseEvent.MOUSE_MOVE, rotateTriangulo);
        stage.removeEventListener(MouseEvent.RIGHT_MOUSE_UP, paraRotarTriangulo);
        isRotating = false;
    }
}
```

```
// Función para rotar el triángulo
function rotateTriangulo(event:MouseEvent):void {
    if (trianguloActivo != null) {
        var dx:Number = stage.mouseX - trianguloActivo.x;
        var dy:Number = stage.mouseY - trianguloActivo.y;
        var angle:Number = Math.atan2(dy, dx) * 180 / Math.PI;

        trianguloActivo.rotation = lastRotation + (angle - Math.atan2(lastMouseY -
        trianguloActivo.y, lastMouseX - trianguloActivo.x) * 180 / Math.PI);
    }
}
```

La función **eliminarTriangulos** es un método asociado al evento **MOUSE_CLICK** del botón **btnEliminar**. Su propósito es eliminar de la memoria los **Sprites** generados, si es que los hubiese, y desvincular los métodos que tenían asociados a sus eventos para restaurar el estado inicial de la aplicación.

```
function eliminarTriangulos(event:MouseEvent):void {
    btnGenerar.gotoAndStop(1);
    btnGenerar.addEventListener(MouseEvent.CLICK, generarTriangulos);
    btnEliminar.gotoAndStop(2);
    btnEliminar.removeEventListener(MouseEvent.CLICK, eliminarTriangulos);
    btnAcoplar.gotoAndStop(2);
    btnAcoplar.removeEventListener(MouseEvent.CLICK, acoplarTriangulos);
    // Lista de los nombres de los sprites
    var triangulos:Array = [triangulo1, triangulo2, triangulo3, triangulo4, triangulo5,
    triangulo6, trianguloEquilatero];

   for each (var triangulo:Sprite in triangulos) {
       if (triangulo != null){
           // Remover los event listeners de cada triángulo
           triangulo.removeEventListener(MouseEvent.MOUSE_DOWN, iniciaArrastrarTriangulo);
           triangulo.removeEventListener(MouseEvent.RIGHT_MOUSE_DOWN, iniciaRotarTriangulo);
           triangulo.removeEventListener(MouseEvent.MOUSE_UP, paraArrastrarTriangulo);
           triangulo.removeEventListener(MouseEvent.RIGHT_MOUSE_UP, paraRotarTriangulo);
```

```
            // Eliminar el sprite del escenario si está presente
            if (triangulo.parent != null) {
                triangulo.parent.removeChild(triangulo);
            }
        }
    }//Fin del for
}//Fin del método eliminarTriangulos
```

La interactividad del usuario está asegurada mediante los métodos previamente descritos. Sin embargo, es necesario añadir la opción que permite acoplar automáticamente los triángulos generados. Este proceso se lleva a cabo mediante el método **acoplarTriangulos**, el cual está asociado al evento **MOUSE_CLICK** del botón **btnAcoplar**.

```
function acoplarTriangulos(event:MouseEvent):void {
    btnAcoplar.gotoAndStop(2);
    btnAcoplar.removeEventListener(MouseEvent.CLICK, acoplarTriangulos);
    acoplarTriangulo(triangulo1, P.x + trianguloEquilatero.x, P.y + trianguloEquilatero.y, 60);
    acoplarTriangulo(triangulo2, Q.x + trianguloEquilatero.x, Q.y + trianguloEquilatero.y, 180);
    acoplarTriangulo(triangulo3, R.x + trianguloEquilatero.x, R.y + trianguloEquilatero.y, -60);
    acoplarTriangulo(triangulo4, P.x + trianguloEquilatero.x, P.y + trianguloEquilatero.y, 60 + e);
    acoplarTriangulo(triangulo5, Q.x + trianguloEquilatero.x, Q.y + trianguloEquilatero.y, 180 + s);
    acoplarTriangulo(triangulo6, R.x + trianguloEquilatero.x, R.y + trianguloEquilatero.y, b);
}//Fin del método acoplarTriangulos
```

El método anterior se encarga de acoplar los seis triángulos alrededor del triángulo equilátero de manera automática. Sin embargo, el trabajo individual de acoplar cada triángulo es realizado por el método **acoplarTriangulo**, que aplica cálculos de traslación y rotación en una sola interpolación. A continuación, se describe el funcionamiento de este método.

```
function acoplarTriangulo(triangulo:Sprite, coorXfin:int, coorYfin:int, angFinal:Number):void {
    // Crear la interpolación (Tween) para mover el sprite a las coordenadas (xFin, yFin)
    var tweenX:Tween = new Tween(triangulo, "x", None.easeNone, triangulo.x, coorXfin, 2, true);
    var tweenY:Tween = new Tween(triangulo, "y", None.easeNone, triangulo.y, coorYfin, 2, true);

    // Crear la interpolación para rotar el sprite al ángulo 'ang'
    var tweenRotation:Tween = new Tween(triangulo, "rotation", None.easeNone, triangulo.rotation,
    angFinal, 2, true);
}
```

Los botones **btnMas** y **btnMenos**, los cuales controlan giros finos de un grado en el **Sprite** activo, tienen los siguientes métodos asociados a sus eventos **MOUSE_CLICK**.

```
btnMas.addEventListener(MouseEvent.CLICK, sumar1);
function sumar1(event:MouseEvent):void {
    if (trianguloActivo != null) trianguloActivo.rotation++;
}
```

```
btnMenos.addEventListener(MouseEvent.CLICK, menos1);
function menos1(event:MouseEvent):void {
    if (trianguloActivo != null) trianguloActivo.rotation--;
}
```

En el clip **Menú**, al iniciarse, se deben configurar todos los botones, excepto **btnInicio**, para que muestren su estado alterno utilizando el siguiente código:

```
btnPresentacion.gotoAndStop(2);
btnRompecabezas.gotoAndStop(2);
btnAcercaDe.gotoAndStop(2);
btnSalir.gotoAndStop(2);
```

También se deben configurar los clips de película en forma de botones que contiene, para que asuman comportamiento de botones. Por ejemplo, el código para el **btnAcercaDe** es el siguiente:

```
btnAcercaDe.buttonMode = true;
btnAcercaDe.useHandCursor = true;
btnAcercaDe.mouseChildren = false;
btnAcercaDe.addEventListener(MouseEvent.CLICK, irAcercaDe);
```

Es importante tener en cuenta que, cuando el usuario presiona cualquiera de los botones, excepto **btnRompecabezas**, los **Sprites** de los triángulos deben eliminarse. Si estos objetos permanecen, pueden superponerse en cualquier fotograma al que el usuario navegue. Por lo tanto, en los métodos de estos botones, debe ejecutarse el método **eliminarTriangulos** desde la película principal. A continuación, se muestra como ejemplo el código que se dispara con el evento **MOUSE_CLICK** del botón **btnAcercaDe**.

```
function irAcercaDe(event:MouseEvent):void {
    btnInicio.gotoAndStop(2);
    btnPresentacion.gotoAndStop(2);
    btnRompecabezas.gotoAndStop(2);
    btnAcercaDe.gotoAndStop(1);
    btnSalir.gotoAndStop(2);
    if (MovieClip(root).currentFrame == 3) MovieClip(root).eliminarTriangulos(event);
    MovieClip(root).gotoAndStop(4);
}
```

En la sección **Acerca de**, se incluyó un botón llamado **btnVideo**, que enlaza a un video demostrativo sobre el uso de la aplicación. A continuación, se muestra el código correspondiente.

```
var URL : URLRequest;

btnVideo.addEventListener(MouseEvent.CLICK, irAvideo);
function irAvideo(e:MouseEvent):void {
    URL = new URLRequest("https://drive.google.com/file/d/1efZ7kgQ5zFGc_45Eqfef9GKgTNh0nW8D/view?usp=sharing");
    navigateToURL(URL);
}
```

Por último, el código para salir de la aplicación es:

```
function salir(event:MouseEvent):void {
    NativeApplication.nativeApplication.exit();
}
```

3.4.4 ALGUNAS CONSIDERACIONES TÉCNICAS.

El software **Triángulo de Morley** fue desarrollado para ser compatible con los sistemas operativos **Windows**, **Android** y **Linux**, ofreciendo versiones optimizadas para cada plataforma.

En la versión para **Windows**, se proporciona un archivo comprimido (TrianguloDeMorleyWindows.zip) que contiene el instalador ejecutable (TrianguloDeMorley.exe). Este archivo guía al usuario paso a paso durante el proceso de configuración para asegurar una instalación sin complicaciones. Además, es indispensable tener instalada la tecnología **Adobe AIR** en el equipo. Para facilitar este requisito, se incluye el instalador correspondiente (AdobeAIR_v51.1.1.5.exe).

Esta integración garantiza que el software funcione correctamente y proporcione al usuario la experiencia completa sin necesidad de configuraciones adicionales complicadas.

Para dispositivos **Android**, se proporciona el archivo comprimido TrianguloDeMorleyAndroid.zip, que contiene dos paquetes APK:

- TrianguloDeMorley**32.apk**: diseñado para tabletas Android más antiguas con arquitectura de 32 bits.
- TrianguloDeMorley**64.apk**: optimizado para tabletas Android con arquitectura de 64 bits.

Si bien la aplicación puede instalarse en teléfonos celulares, no se recomienda su uso en pantallas pequeñas, ya que la interfaz puede volverse difícil de manejar. En particular, algunos botones de la app podrían ser complicados de presionar sin un lápiz electrónico o stylus.

Estos archivos APK están listos para ser instalados en dispositivos móviles. Sin embargo, la instalación de aplicaciones de fuentes externas requiere la activación de la opción **"Permitir instalación de aplicaciones de orígenes desconocidos"**. El procedimiento para habilitar esta opción varía según la versión de Android:

- **Android 7 y versiones anteriores:** En estas versiones, la activación de la opción "**Permitir instalación de aplicaciones de orígenes desconocidos**" se realiza desde **Ajustes del dispositivo** > **Seguridad**. A continuación, se presentan algunas capturas de pantalla que ilustran el proceso en estos sistemas Android, mostrando los pasos necesarios para habilitar esta configuración.

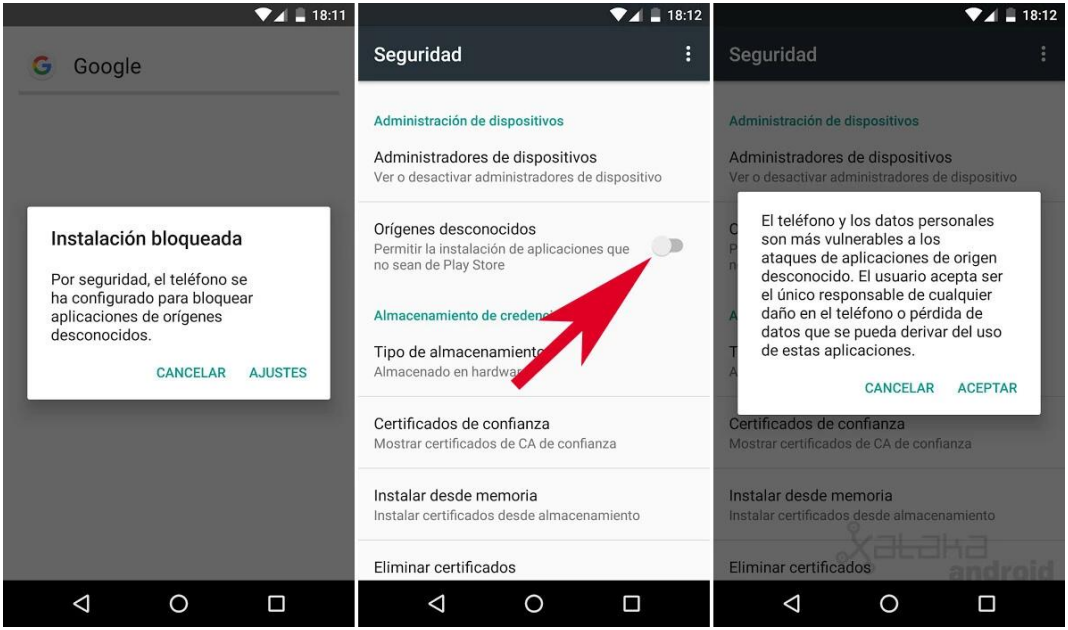


Figura 49 Activación de Orígenes desconocidos. Fuente (Xataka Android, octubre 13 de 2024)

- **Android 8 y versiones posteriores:** En estas versiones, la opción de instalar aplicaciones desde fuentes desconocidas ya no se encuentra en los ajustes generales del sistema. En su lugar, el permiso se concede por aplicación individual (por ejemplo, Chrome o un administrador de archivos). Para habilitar esta opción, el usuario debe acceder a **Ajustes** > **Aplicaciones** > **[Aplicación que descarga el APK]** > **Instalar aplicaciones desconocidas** y activar el permiso para cada aplicación que se utilizará para la instalación.

Por ejemplo, si el archivo APK se descarga desde un navegador como Chrome, el dispositivo mostrará pantallas similares a las ilustradas en la Figura 50. Este enfoque garantiza un control más granular sobre las fuentes externas, mejorando la seguridad del dispositivo al limitar los permisos solo a las aplicaciones necesarias para cada instalación.

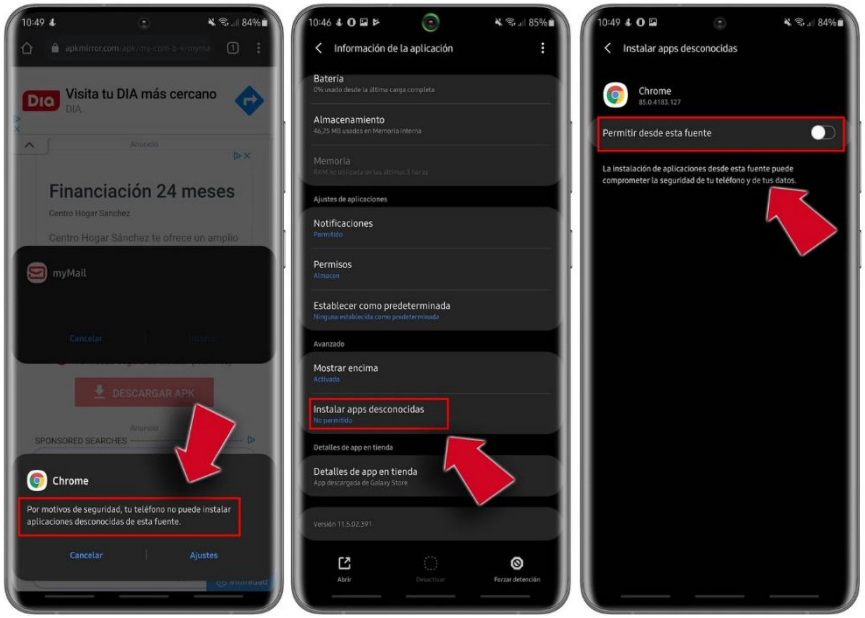


Figura 50 Pantallas en sistemas Android 8 o superiores. Fuente (Xataka Android, octubre 13 de 2024)

Si el archivo APK se transfiere al dispositivo mediante **cable USB, memoria externa** o **Bluetooth**, la instalación se realizará a través de un **administrador o gestor de archivos**, como la aplicación **Mis Archivos**. En este caso, el usuario debe otorgar los permisos

necesarios para permitir la instalación desde esta fuente específica. Para hacerlo, debe acceder a **Ajustes** > **Aplicaciones** > **[Administrador de archivos utilizado]** > **Instalar aplicaciones desconocidas** y habilitar el permiso correspondiente para **Mis Archivos** o cualquier otro gestor utilizado.

La Figura 51 muestra las pantallas típicas que aparecen durante este proceso, ilustrando los pasos necesarios para completar la instalación correctamente. Este método permite instalar aplicaciones desde medios externos de manera controlada, garantizando la seguridad del dispositivo.

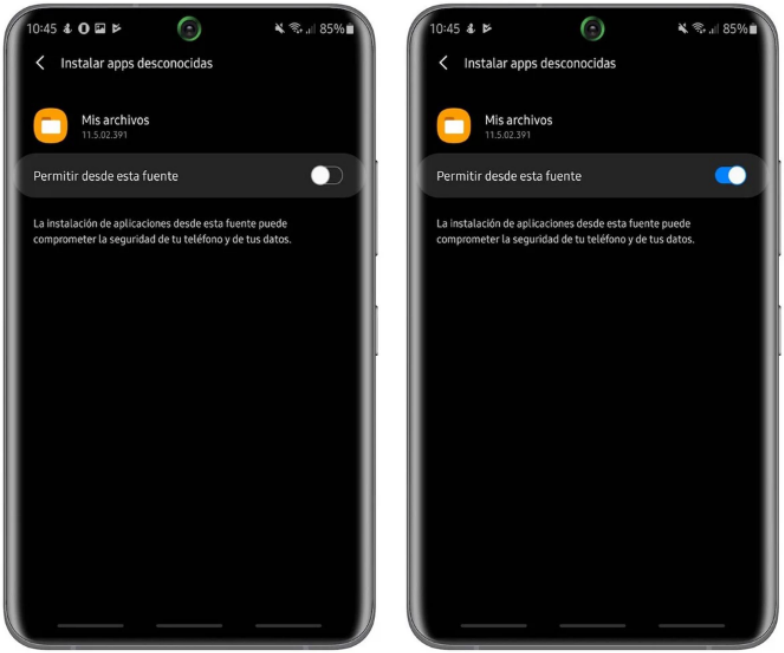


Figura 51 Activar la opción para Mis Archivos. Fuente (Xataka Android, octubre 13 de 2024)

- **Android 14:** Esta versión introduce una capa adicional de seguridad al requerir una verificación mediante **Google Play Protect** antes de permitir la instalación de aplicaciones de fuentes externas. Estas medidas garantizan que el usuario sea consciente de los riesgos potenciales asociados con la instalación de aplicaciones no oficiales y refuerzan la protección del sistema operativo contra software malicioso.

Si la aplicación no supera la verificación, el sistema emite una alerta sobre los riesgos detectados y recomienda precaución al usuario. Además, Android 14 ha incrementado las restricciones, evitando la instalación de aplicaciones que utilicen APIs obsoletas, lo que garantiza que el software instalado cumpla con los estándares actuales de seguridad y rendimiento.

La Figura 52 muestra cómo se presenta la interfaz en un dispositivo con **Android 14** durante el proceso de instalación, incluyendo las opciones de verificación y las alertas correspondientes. **Triángulo de Morley** es una aplicación completamente segura, libre de virus y malware que puedan comprometer su dispositivo. Por lo tanto, no es necesario utilizar la opción **"Analizar app"** que aparece en la pantalla ilustrada.

Para proceder con la instalación sin realizar el análisis, despliegue la opción **"Más detalles"** y seleccione **"Instalar sin analizar"**. Esto permitirá completar la instalación de manera rápida y segura, sin que se active el análisis adicional de Google Play Protect.

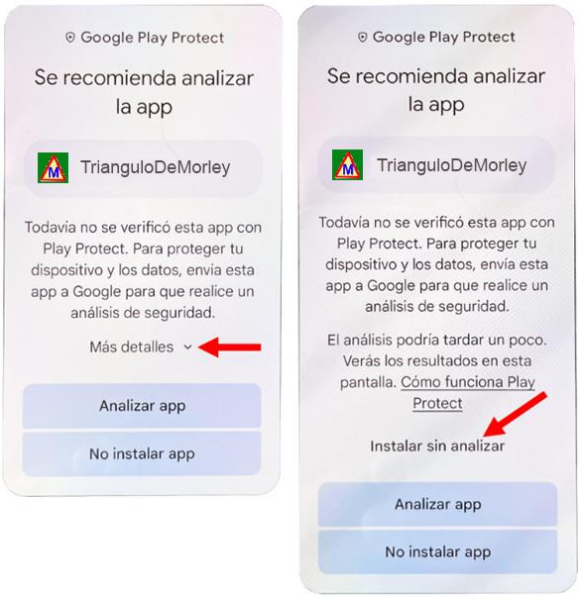


Figura 52 En Android 14. Fuente propia.

Para **Linux**, la aplicación se distribuye en un archivo comprimido, **CajaDePolinomiosLinux.zip**, que contiene la carpeta con todos los recursos necesarios para su correcta ejecución. Es imprescindible instalar previamente **Wine**, una aplicación que proporciona un entorno de compatibilidad en sistemas Linux, permitiendo ejecutar software diseñado para Windows sin complicaciones.

La Figura 53 ilustra este proceso: a la izquierda, se muestra la carpeta de **Triángulo de Morley** en un sistema **Linux Ubuntu**; y a la derecha, se observa el ícono característico de las aplicaciones Windows listo para ser ejecutado mediante **Wine**. La ejecución es sencilla, basta con hacer **doble clic** en el archivo **.exe** incluido para iniciar la aplicación.

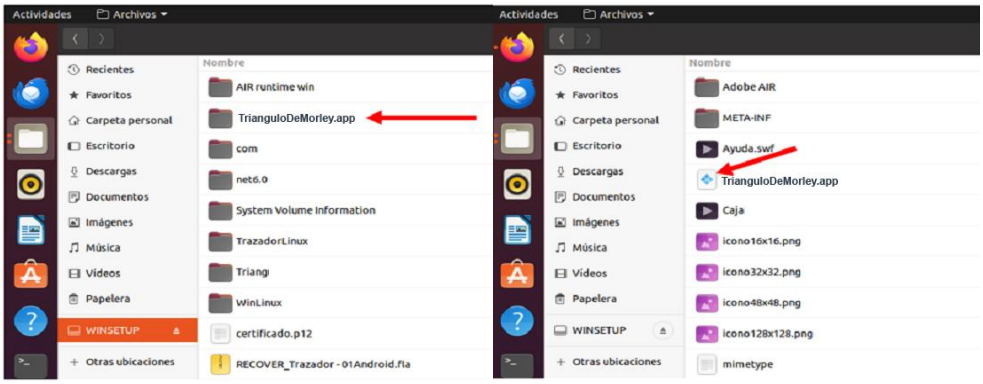


Figura 53 Carpeta y ejecutable en Linux Ubuntu. Fuente propia.

La Figura 54 muestra la ejecución del software **Triángulo de Morley** en **Linux Ubuntu** utilizando **Wine** como entorno de compatibilidad. Esta ilustración destaca cómo la aplicación se integra y funciona sin problemas en un entorno Linux, permitiendo a los usuarios acceder a todas sus funcionalidades como si se tratara de un sistema Windows.

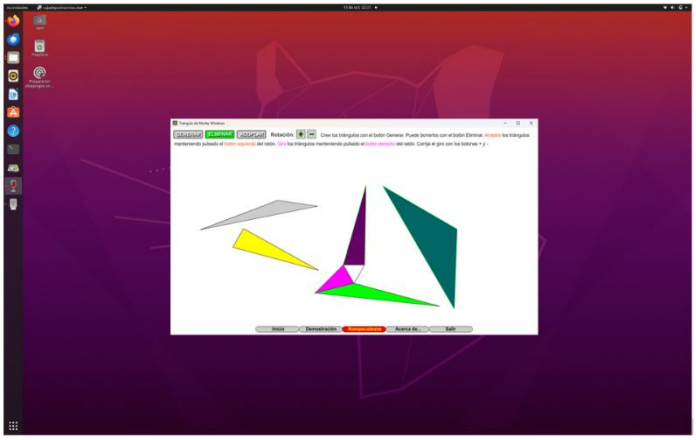


Figura 54 Triángulo de Morley en Linux Ubuntu. Fuente propia.

3.4.5 DESCARGA DE LA APLICACIÓN.

A continuación, se presentan los **enlaces de descarga** para las distintas versiones de la aplicación **Triángulo de Morley**, asegurando compatibilidad con múltiples plataformas:

Instalador para Windows:

https://drive.google.com/file/d/1JvYwNkCdjovpw6IsmcdF662rrcrl_jn0/view?usp=sharing

Carpeta para ejecución en Linux bajo Wine:

https://drive.google.com/file/d/1Htx3EbB_dfTvEsR3mJM3FBueWI2azYWX/view?usp=sharing

APK de 32 y 64 bits para Android:

https://drive.google.com/file/d/1OEAk9rnudWMXX9gNIGZhAQrgu3sqOObe/view?usp=sharing

4. CONCLUSIONES.

En esta última sección, se exponen a modo de consideraciones finales, algunas reflexiones y observaciones que complementan el análisis realizado a lo largo de este documento:

- El **teorema de Morley** destaca como un resultado geométrico de características excepcionales que ha fascinado a matemáticos por más de 120 años. A lo largo de este tiempo, se han formulado más de cien demostraciones, la mayoría empleando técnicas de geometría euclidiana o trigonometría. Es interesante notar cómo estas demostraciones han evolucionado; los enfoques modernos, en general, son más claros en sus argumentos y requieren menos detalles técnicos, lo que las hace más accesibles y comprensibles en comparación con las versiones originales.
- Una cuestión aún por explorar en profundidad es por qué este teorema no fue descubierto por los antiguos griegos. La razón principal parece ser que los griegos no se interesaban por problemas geométricos cuya construcción no pudiera resolverse utilizando instrumentos euclidianos como la regla y el compás. Este es el caso del problema de la trisección del ángulo, que es clave para el Teorema de Morley, ya que no puede resolverse mediante estos instrumentos clásicos.
- El **Origami**, una técnica milenaria originaria de Japón, se ha revelado como una herramienta útil en geometría. Esta técnica,

basada en el plegado de papel sin cortes ni adhesivos, puede no solo potenciar la capacidad de visualizar y razonar espacialmente, sino también desarrollar destrezas manuales y coordinación. En este trabajo, la construcción del **triángulo de Morley** mediante origami se fundamenta en el método de trisección de ángulos desarrollado por el matemático japonés **Tetsuo Abe**, demostrando la conexión entre el arte y la matemática.

- La aplicación interactiva relacionada con el Teorema de Morley presentada en este texto, ha sido diseñada para funcionar en las plataformas **Windows**, **Android** y **Linux**; además, se ha optimizado para ofrecer una experiencia amigable para el usuario final. Cabe recordar que, su descarga se ha descrito en el numeral 3.4.5 de este libro. Asimismo, como complemento, se ha desarrollado un recurso en **GeoGebra**, basado en la demostración del teorema de **John Conway**, que permite recrear el Teorema de Morley de forma dinámica y visual. Este recurso está disponible en la siguiente dirección:
  https://www.geogebra.org/classic/fufajqqe

REFERENCIAS

Bogomolny, A. (s.f.) Morley's Miracle, Interactive Mathematics Miscellany and Puzzles, http://www.cut-the-knot.org/triangle/Morley/

Branch, R. M. (2009). Instructional Design: The ADDIE Approach. Springer Science & Business Media.

Chaves, A. (1993). Trigonometría, Universidad Nacional, Seccional Manizales, ISBN 958-9322-05- 0

Conway, J. (2014). On Morley's Trisector Theorem. The Mathematical Intelligencer, vol. 36, n. 3, 3.

Coxeter, H. S. M. y Greitzer, S. L. (1967). Geometry Revisited, The Mathematical Association of America.

de Jesús Landaverde, F. (1987). Geometría. Editorial Progreso.

EOS project. (n.d.). Institute for Educational Origami. IEO. https://ieo.sakura.ne.jp/wp/

Ghiocas, G. M. D. (1934). Sur un theoreme de la theorie du triangle, Actes Congres Interbalkan Math., Athenes.

Hatori, K. (2005). K's Origami- Fractional Library- Origami Construction. URL http://origami.ousaan.com/library/conste.html

Huzita, H. (1989). Axiomatic Development of Origami Geometry. In: Proceedings of the First International Meeting of Origami Science and Technology.143–158.

Ida, T., Takahashi, H., Marin, M. (2006). Computational Origami of a Morley's Triangle. In: Kohlhase, M. (eds) Mathematical

Knowledge Management. MKM 2005. Lecture Notes in Computer Science(), vol 3863. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11618027_18

Letac, A. (1939). Solution (Morley's triangle), Problem No. 490 [Sphinx: revue mensuelle des questions recreatives, Brussels, 8 (1938) 106], Sphinx, 9 (1939) 46.

Molenda, M. (2003). "In Search of the Elusive ADDIE Model." Performance Improvement, 42(5), 34-36.

Oakley, C. O., & Baker, J. C. (1978). The Morley trisector theorem. The American Mathematical Monthly, 85(9), 737-745.

Quang, H. T. (2024). Short note. A synthetic proof of the Morley trisector theorem using congruent and similar triangles. Elem. Math. Published online first. DOI 10.4171/EM/527

Reiser, R. A., & Dempsey, J. V. (2017). Trends and Issues in Instructional Design and Technology. Pearson.

Taylor F. G. y W. L Marr, W. L. (1913). The six trisectors of each of the angles of a triangle, Proc. Edinburgh Math. Soc., 32.

Tetsuo I., Asem K., Ghourabi F., y Takahashi, H. (2011). Morley's theorem revisited: Origami construction and automated proof. Journal of Symbolic Computation, Volume 46, Issue 5, 571-583

Xataka Android. (2024). Cómo instalar aplicaciones en APK en un móvil Android. Xataka Android.

https://www.xatakandroid.com/tutoriales/como-instalar-aplicaciones-en-apk-en-un-movil-android

ÍNDICE DE FIGURAS

Sobre los autores

Edwin Giovanni Insuasty Portilla

Edwin Insuasty Portilla, Licenciado en Matemáticas y Física, Especialista en Computación para la Docencia, Especialista en Docencia Universitaria, Magister en Modelos de Enseñanza Problémica, DEA en Procesos de Formación en Espacios Virtuales, de la Universidad de Salamanca (España) y Doctor en Procesos de Formación en Espacios Virtuales, de la misma Universidad con pasantía doctoral de 9 meses en la Universidad de Padova (Italia). Ha orientado asignaturas de matemáticas en diversos programas. Actualmente es profesor de Fundamentos de Lógica, Programación I, II y III, Software de Autoría y Programación de Videojuegos. Desarrollador de software para Windows, Linux, Android, IOs y aplicaciones MathLab. Integrante de los grupos de investigación GESCAS de la Licenciatura en Matemáticas, y GALERAS.NET, de Ingeniería de Sistemas.

Saulo Mosquera López

Saulo Mosquera, Licenciado en Matemáticas y Física de la Universidad de Nariño, Especialista en Enseñanza de la Matemática y Magíster en Matemáticas. Fue director del Dpto. de Matemáticas y Estadística de la Universidad de Nariño y Coordinador del grupo de investigación GESCAS. Se pensionó en 2017 en la categoría de profesor Titular. Autor de diferentes textos y artículos relacionados con matemáticas y/o Educación Matemática; actualmente es investigador activo del grupo GESCAS en la línea Comunicación, transformación y objetivación de objetos matemáticos vinculados a registros semióticos bidimensionales. También es su interés la utilización de Software educativo para promover el desarrollo de pensamiento matemático en el aula. Ha ejercido como Par Académico del CNA para la evaluación de programas académicos, de Pregrado y Posgrado, relacionados con Educación Matemática.

êditorial
Universidad de **Nariño**

Año de publicación: 2025
San Juan de Pasto - Nariño - Colombia

Este libro ofrece una travesía fascinante por el **Teorema de Morley**, destacando su relevancia tanto en la matemática teórica como en aplicaciones prácticas innovadoras. A través de un enfoque interdisciplinario, los autores entrelazan las matemáticas, el origami y la programación para presentar una obra que combina rigor académico con creatividad y practicidad.

En el ***Capítulo 1***, se introduce al lector al Teorema de Morley, analizando su belleza matemática y desarrollando tres de sus demostraciones más destacadas: la de M. T. Naraniengar (1909), A. Letac (1939) y el colombiano A. Chaves (1993). Cada demostración se explica paso a paso, priorizando la claridad y accesibilidad para una amplia audiencia, desde matemáticos hasta estudiantes entusiastas. El ***Capítulo 2*** aborda el teorema desde una perspectiva visual y experimental mediante el ***origami matemático***. Se explora cómo los pliegues de papel pueden modelar conceptos geométricos complejos, con un énfasis especial en el ***Método de Abe*** para trisecar ángulos y construir el Triángulo de Morley. Este capítulo incluye una guía detallada de técnicas de origami computacional, combinando arte y algoritmos para una experiencia de aprendizaje única. En el ***Capítulo 3***, el libro traslada el Teorema de Morley al ámbito digital con el desarrollo de un **rompecabezas interactivo**. Los lectores aprenden sobre gráficos vectoriales, mapas de bits y cómo seleccionar herramientas como ***Visual Studio, Android Studio o Adobe Animate*** para crear aplicaciones funcionales y atractivas. Además, se detalla una metodología de desarrollo que abarca desde la planificación hasta la implementación, asegurando que los proyectos sean técnicamente sólidos y accesibles para usuarios de diferentes plataformas.

Más que una obra matemática, este libro es una invitación a explorar las conexiones entre disciplinas. Al combinar teoría abstracta, arte manual y programación, muestra cómo la matemática puede ser creativa, tangible y divertida. Dirigido a matemáticos, entusiastas del origami y programadores, este libro es una muestra brillante de cómo las matemáticas trascienden su rigor para convertirse en una experiencia educativa y recreativa. ***"Aspectos Teóricos y Prácticos del Teorema de Morley"*** inspira a sus lectores a transformar conceptos abstractos en herramientas para la creatividad y el aprendizaje.

ISBN: 978-628-7771-33-8

9 786287 771338

Universidad de Nariño
FUNDADA EN 1904

Universidad de Nariño
ACREDITADA EN ALTA CALIDAD
RESOLUCIÓN MEN 000022 - ENERO 11 DE 2023

120 Años
Universidad de Nariño

êditorial
Universidad de Nariño