Figura 5. Modelo "cremallera" del movimiento de rodadura.

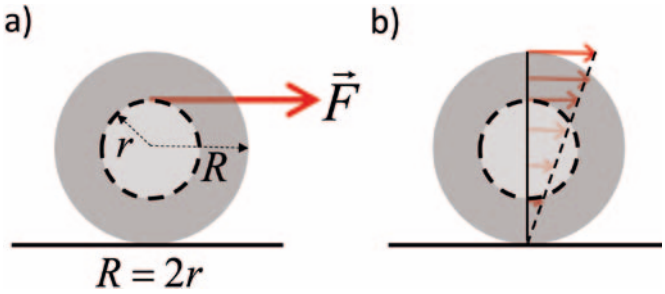

Figura 6. Situación de rodadura de un carrete de radios r y R del que se tira con una fuerza externa F (vector rojo) y donde la fuerza de rozamiento estático es nula. **a)** Fuerza externa horizontal aplicada en la parte superior. **b)** Distribución de velocidades del caso a).

Conclusiones

Los docentes de Física debemos de ser conscientes de la complejidad del aprendizaje de la fuerza de rozamiento estático. Debe presentarse como una manifestación de la adhesión entre cuerpos en contacto (constreñidos mutuamente como consecuencia de su propio movimiento), pero siempre que exista movimiento relativo tentativo a nivel local (entre zonas/puntos en contacto). Este movimiento relativo tentativo se debe razonar cuidadosamente a partir de las fuerzas externas locales o la inercia. La fuerza de rozamiento estático se opone a este intento de movimiento. A partir de ahí, es una incógnita más del problema. En cursos más avanzados, al igual que las fuerzas normales de apoyo se tratan como ligaduras geométricas, conviene tratar la fuerza de rozamiento estático como una ligadura cinemática que justifique el movimiento solidario entre cuerpos, mientras dure.

Referencias

[1] U. Besson, L. Borghi, A. De Ambrosis y P Mascheretti, "How to Teach Friction: Experiments and Models", *Am. J. Phys.* **75**,1106-1113 (2007).

[2] M. F. Arcodía y S. M. Islas, "Las fuerzas de roce en libros de texto y en revistas científicas", *Revista de Enseñanza de la Física* **19**(2), 7-24 (2006).

[3] A. Manzur, "Cuando la fuerza de fricción estática se convierte en fuerza de fricción cinética y viceversa", *Rev. Mex. Fis. E* **54**, 51-54 (2008).

[4] G. T. Pickett, "A Pedagogical Model of Static Friction", http://arxiv.org/abs/1507.04015, 2015.

[5] K. D. Hahn y J. M. Russell, "The Indeterminate Case of Classical Static Friction When Coupled with Tension", *Phys. Teach.* **56**, 83 (2018).

[6] J. Güémez, "Sobre trabajo y fuerza de rozamiento", *REF* **31**(2), 29-33 (2017).

Visualizacion de la ley de Snell a partir del principio de Fermat con el modulo *turtle* de Python

Fernando Blasco
Alumno del Máster de Formacion del Profesorado,
Universidad Politécnica de Madrid

Juan Manuel Pastor
Grupo de Sistemas Complejos
Universidad Politécnica de Madrid

Javier Galeano
Grupo de Sistemas Complejos
Universidad Politécnica de Madrid

Ilustrar la deducción de la ley de Snell a partir del principio de Fermat nos permite mostrar una alternativa más visual e intuitiva, que complementa la deducción trigonométrica a partir del principio de Huygens. Para ello, se presenta un pequeño programa de Python que muestra, mediante una carrera de tortugas, que el camino más rapido es el predicho por la ley de Snell.

> —Está bien —dijo Momo—, voy contigo. ¿Pero no podría llevarte, para ir un poco más deprisa? "No", ponía en el caparazón de Casiopea. —¿Por qué tienes que arrastrarte tú misma? —preguntó Momo—. A esto vino la enigmatica respuesta: "El camino esta en mí". Con esto, la tortuga se puso en marcha y Momo la siguió, poco a poco y pasito a pasito.
>
> *Momo,* Michael Ende

Motivación

En el currículum de Física de 2.º de Bachillerato se debe introducir la ley de Snell, ya que es fundamental para el desarrollo del bloque de la óptica geométrica [1]. Normalmente, esta ley se introduce de manera fenomenológica, pero si se deduce, suele hacerse utilizando el principio de Huygens [2]. En este trabajo presentamos una manera diferente de llegar a la ley de Snell, utilizando el principio de Fermat, basado en la minimización del tiempo. Para una mejor comprensión de la idea, nos apoyamos en un programa que hemos diseñado para poder visualizar este principio y llegar experimentalmente a la ley de Snell.

El potencial didáctico del principio de Fermat reside en el contraste entre lo sencillo de su planteamiento y lo profundo de sus implicaciones en la óptica ondulatoria en la que se basa. Abre la puerta a numerosas preguntas y ampliaciones en el campo de la óptica, de la mecánica, de la geometría, de la filosofía e incluso en la computación, a través del programa de Python que presentamos en este trabajo.

El programa que hemos diseñado permite visualizar la relación de dicho principio con la ley de Snell y presentar ambos de manera más lúdica, proponiendo a los estudiantes que inicialmente adivinen y más tarde calculen que tortuga ganará la carrera bajo cada proporción entre índices de refracción (cuya relación con la velocidad de la luz en cada medio, además, quedará clara durante la carrera). Este programa lo puede utilizar el profesor en el ordenador de clase si lo puede proyectar en pantalla o en los ordenadores de los alumnos si se disponen de ellos en el aula. Ademas, se pueden proponer, como actividades avanzadas a los alumnos (si tienen nociones básicas de programación), algunas modificaciones sencillas del código para profundizar más en el principio de Fermat, por ejemplo, comprobar que los caminos cercanos al tomado por la luz varían poco en tiempo, como sugerimos en este trabajo.

Introducción

El principio de Fermat, en su versión original, se basa en que la luz se propaga de un punto a otro recorriendo el camino más rapido. Es, por tanto, un principio basado en la minimización del tiempo, no como ocurre en el cuarto menguante de la película española dirigida por Luis Piedrahíta y Luis Sopeña, *La habitacion de Fermat,* donde lo minimizado es el espacio [3].

Este principio no pierde vigencia cuando la luz cambia de medio, lo que provoca el fenómeno de la refracción: un rayo cambia su dirección al atravesar regiones en las que se propaga a distinta velocidad, porque de esta manera viaja más rápido de un punto a otro.

Para ilustrar este hecho, hemos desarrollado un programa de Python en el que se proponen varios caminos posibles que podrá recorrer la luz para llegar del punto A al B. Cada camino es recorrido por tortugas de distintos colores, que salen a la vez y "compiten" por ser las primeras en llegar a la meta.

Las tortugas pueden no parecer idóneas para representar a la luz, que se mueve lo más rapido posible [4], pero las fábulas y los documentales de La 2 nos indican que tienen experiencia echando carreras. También ha influido en la elección el hecho de que Python incluya un módulo que permite dibujar guiando a una tortuga, característica original de un lenguaje de programación didáctico desarrollado hace cinco decadas: Logo.

Obtención de la ley de Snell

La ley de Snell, propuesta en 1621, nos indica cómo cambia de dirección la luz al pasar de un medio a otro, relacionando los ángulos de incidencia y refracción (respecto a la normal a la superficie de contacto) con los índices de refracción de cada medio:

$$n_1 \sin(\alpha_1) = n_2 \sin(\alpha_2); \qquad (1)$$

donde n_1 y n_2 son los índices de refracción y α_1 y α_2 los ángulos respecto a la normal, en los medios 1 y 2, respectivamente.

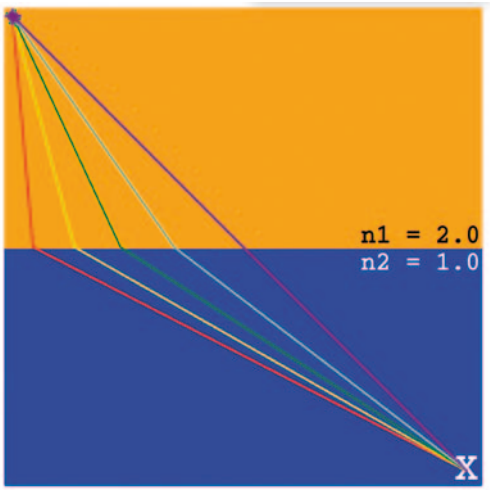

Figura 1. Captura de pantalla del programa, donde se visualiza el camino seguido por cada una de las tortugas, que se encuentran en la esquina superior izquierda preparadas para comenzar la carrera.

Sin embargo, esto era solo una ley fenomenológica, que daba cuenta de lo que ocurría sin relacionarlo con un principio físico subyacente. Fermat propuso su principio de mínimo tiempo en 1662, mientras que Huygens introdujo su principio (basado en que cada punto de un frente de ondas es a su vez emisor de ondas esféricas) en 1678. Fue modicado por Fresnel, quien le dio su forma actual, en 1818 [5-7]

A traves del principio de Huygens

El principio de Huygens explica la propagación de un frente de ondas considerando que cada punto del mismo es un emisor de ondas secundarias, que interfieren entre sí originando el siguiente frente de onda.

Es común aplicarlo a frentes de onda planos para la deducir la ley de Snell, y suele ser la demostración de la ley mostrada a los estudiantes de Bachillerato e incluida en la mayoría de libros de texto (si no la deducción completa, sí un esquema como el de la Figura 2). Dicha deducción, que no vamos a desarrollar aquí para centrarnos en la basada en el principio de Fermat, se puede encontrar en numerosos textos, como [8].

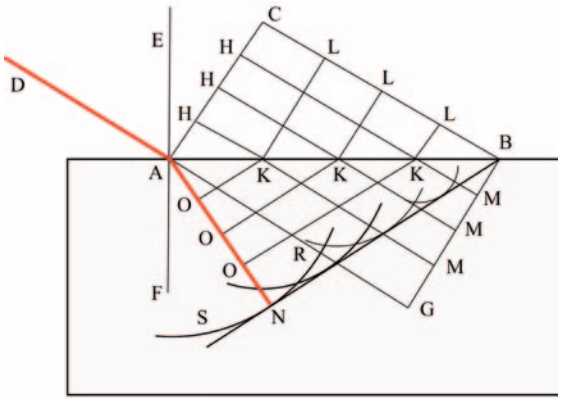

Figura 2. Esquema de la deducción de la ley de Snell de Huygens, incluido en su *Tratado sobre la luz* [9].

A traves del principio de Fermat

La afirmación de que la luz viaja de un punto a otro siguiendo el camino que minimiza el tiempo es el principio original-

mente planteado por Fermat, válido en la mayoría de casos, pero no en todos. El ejemplo más evidente de violación del mismo es la reflexión: claramente, en la Figura 3, la luz tardaría menos en ir del punto A al B en línea recta en lugar de reflejarse en la superficie del espejo.

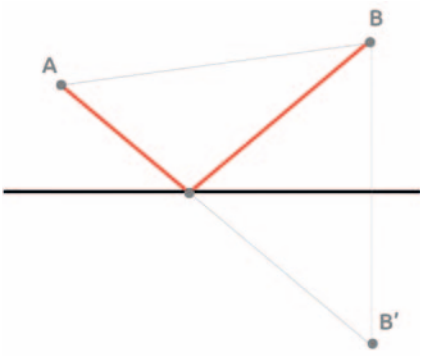

Figura 3. El segmento AB es más corto que la suma de los dos marcados en rojo. Sin embargo, el camino marcado en rojo sí que es la forma más rápida de ir de A a B pasando por un punto de la interfase (la supercie del espejo). Se puede demostrar fácilmente percibiendo que el camino en rojo tiene la misma longitud que el segmento AB', siendo B' el punto simétrico de B respecto a la interfase [10].

En la versión vigente del principio, Fermat figura un cambio sutil: el tiempo en recorrer el camino no debe ser mínimo, sino estacionario frente a variaciones de la trayectoria. Es decir, la luz sigue el camino que verifica que, bajo pequeñas alteraciones, el tiempo en recorrerlo apenas cambia (no varía en primer orden) [11].

Este principio, por su formulación aparentemente teleológica, puede incitar debates filosóficos o incluso místicos. Los últimos son estériles debido a que, al contrario que Casiopea (la tortuga del maestro Hora en *Momo*), la luz no sabe cuál es el camino más corto de antemano. Simplemente es reemitida en todas direcciones, pero su camino de propagación es aquel en el que hay interferencia constructiva. Esta se da cuando la diferencia de fase es mínima, lo cual sólo ocurre en caminos estacionarios (en los que pequeñas diferencias de camino no causan diferencias de fase). En el resto de caminos posibles, al no ser estacionarios, convergen ondas con distintas fases que interfieren destructivamente [12].

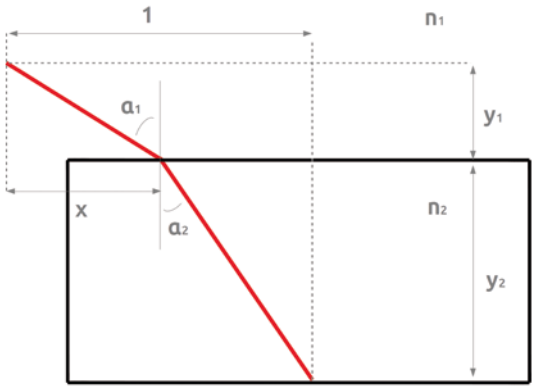

Figura 4. Refracción de un rayo al pasar de un medio con índice n_1 a otro con n_2.

Vamos a deducir la ley de Snell usando el concepto de minimizar el tiempo, por tanto, expresamos el tiempo como el espacio recorrido en cada tramo dividido por la velocidad en el mismo:

$$t = \frac{\sqrt{x^2 + y_1^2}}{c/n_1} + \frac{\sqrt{(1-x)^2 + y_2^2}}{c/n_2} \qquad (2)$$

Derivamos según el tramo recorrido en el primer medio, x, que es el único parámetro libre, e igualamos a cero:

$$\frac{dt}{dx} = + \frac{n_1 x}{c\sqrt{x_2 + y_1^2}} - \frac{n_2 (1-x)}{c\sqrt{(1-x)^2 + y_2^2}} = 0 \qquad (3)$$

Por tanto,

$$\frac{n_1 x}{\sqrt{x_2 + y_1^2}} - \frac{n_2 (1-x)}{c\sqrt{(1-x)^2 + y_2^2}} \qquad (4)$$

Usando la definición de seno y coseno, para los ángulos con respecto a la normal (Figura. 4) obtenemos la ley de Snell (Eq. 1).

Como podemos ver, se obtiene la ley de Snell de una manera sencilla y utilizando conceptos que los alumnos de 2.º de Bachillerato deberían conocer.

El programa

Para que los alumnos puedan trabajar con la ley o simplemente divertirse utilizándola, hemos diseñado un programa en Python que está disponible en https://github.com/ferblasco7/tortugas/blob/ master/SnellCarreraTortugas.py.

Está inspirado en la analogía de Feynman para explicar el cambio de dirección de la luz en la refracción: habla de un socorrista que va más rapido corriendo por la arena que nadando por el mar, y se pregunta cuál es el camino que le permite llegar antes al punto donde se está ahogando una persona [10]. Nuestras tortugas comparan la efectividad de distintos caminos posibles aunque, al contrario que el socorrista, van más rapido por el agua que por la arena.

Módulo turtle *en Python*

En la actualidad, la programación que realizan nuestros alumnos de ESO y Bachillerato se suele realizar en Scratch, el lenguaje de programacion diseñado en el MIT. Antes de esto, nuestros adolescentes comenzaban su camino en la programación usando el lenguaje Logo. El lenguaje fue diseñado en 1967 con fines didácticos por Danny Bobrow, Wally Feurzeig, Seymour Papert y Cynthia Solomon. Logo es un lenguaje de programación de alto nivel de muy fácil aprendizaje, por lo que solía usarse como el lenguaje de programación para enseñar a los niños y jóvenes.

Papert desarrolló un enfoque en el que presentaba a los niños retos intelectuales que puedan ser resueltos mediante el desarrollo de programas en Logo. Y básicamente se trataba de que las tortugas gráficas cumplieran todas las órdenes programadas. La "tortuga" de Logo es un cursor al que se le pueden dar órdenes de movimiento y que puede ir dejando un rastro sobre la pantalla. Moviendo adecuadamente la tortuga se puede conseguir dibujar todo tipo de figuras.

Siguiendo las ideas de Papert, en este trabajo se utilizan las tortugas de Logo para resolver un problema de óptica, pero en

nuestro caso hemos usado el lenguaje de programación Python. En la actualidad, Python es un lenguaje mucho más moderno y extendido. En muchos casos se convierte en el lenguaje de programación que usarán algunos de los estudiantes en la universidad. En Python existe un módulo llamado "turtle" que permite realizar las gráficas de las tortugas de Logo [13]. Una vez instalado Python en cualquiera de sus entornos (p. e., Anaconda), basta con instalar el módulo *PythonTurtle* (bajo licencia *MIT License*) [14], para después importarlo en el código.

Uso del programa

Al iniciar el programa, se nos pide fijar el índice de refracción del primer medio.

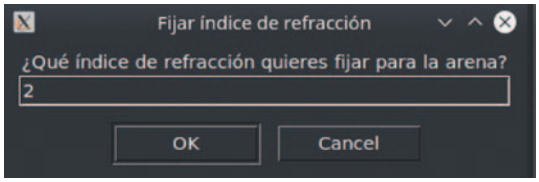

Figura 5. Diálogo para elegir la relación n_1 / n_2 (se fija $n_2 = 1$ por simplicidad).

Posteriormente, se dibuja el sencillo entorno gráfico y se pregunta al usuario si quiere comenzar la carrera.

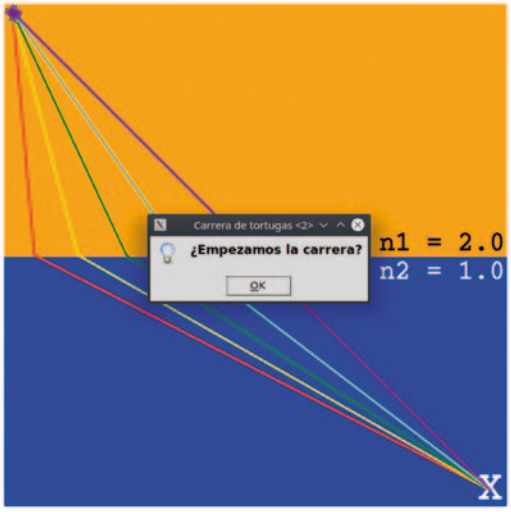

Figura 6. Todo listo para empezar.

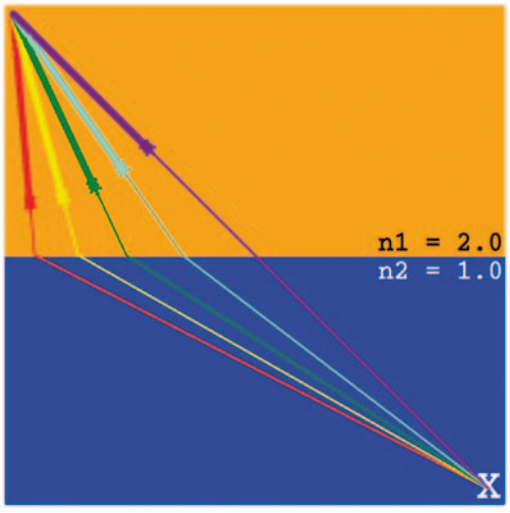

Figura 7. Carrera en curso.

Se va plasmando el orden de llegada en la parte superior derecha, donde aparecen ordenadas segun su clasificación en la carrera.

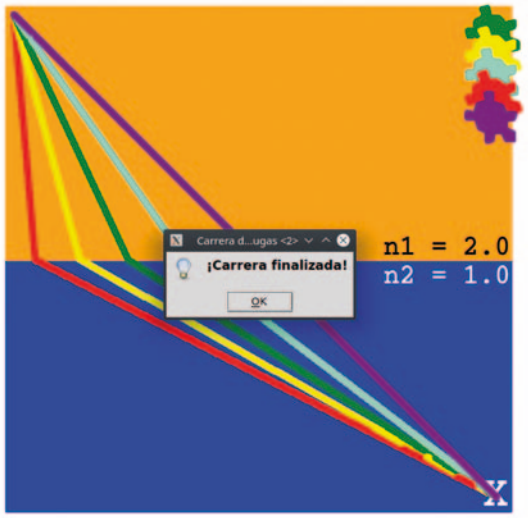

Figura 8. El programa finaliza cuando todas las tortugas han llegado a meta.

Al presentarse el código estructurado y explicado por partes, se facilita al profesor (o a los alumnos con pequeñas indicaciones del profesor) introducir algunas modificaciones, sin mucha dificultad. Por ejemplo, fácilmente pueden cambiar el dibujo de la tortuga por otro que les pueda resultar más divertido; también se puede sofisticar el programa haciendo que al girar la tortuga aparezcan los ángulos de incidencia y refracción, o se podría añadir un pequeño rótulo en el que se indicase el camino óptico ($L = l \cdot n$) recorrido por cada una de las tortugas.

Conclusiones

En este artículo presentamos un programa diseñado en Python, utilizando la librería *turtle*, para ilustrar la ley de Snell mediante el principio de Fermat. El código está estructurado por partes para que los alumnos entiendan su funcionamiento y puedan modificarlo según sus necesidades de uso. Pensamos que aproximarse a la ley de Snell de una manera diferente puede hacer la tarea más sencilla y divertida.

Apéndice: Descripción del código

La descripción del código se encuentra también disponible en formato Jupyter Notebook: https://github.com/ferblasco7/tortugas/blob/ master/Carrera%20de%20tortugas.ipynb

Importamos el modulo *turtle* para poder usar las tortugas, *numpy* para cálculos trigonometricos y *tkinter* para emplear diálogos (y que el usuario pueda fijar el índice de refracción, decidir cuándo empieza la carrera, etc.).

```
import turtle
from numpy import sin, tan, arctan, radians, degrees
import tkinter #crear dialogos
```

Preguntamos, mediante un diálogo, el índice de refracción de la arena (el del agua se toma como 1 por simplicidad), y fijamos las coordenadas de la meta y la salida, junto con los ángulos de partida y colores de cada tortuga.

```
parent = tkinter.Tk()
parent.overrideredirect(1) # para que no desaparezca
parent.withdraw()
simpledialog=tkinter.simpledialog
n = simpledialog.askfloat('Fijar índice de refracción'...
#Coordenadas de meta y salida
meta=[330,-330]
salida=[-340,340]
X=meta[0]-salida[0] #distancia recorrida en X (ambos medios)
Y1,Y2=330,340 #distancia recorrida en Y en cada medio
posicion_y_podio=320 #coordenada y del podio
#angulos (respecto al eje x ...
angulos=[-45,-55,-65,-75,-85] #en orden inverso...
colores=['purple','#7FFFD4','green','#FFFF00','red'] #...
```

Configuramos el título y dimensiones de la ventana:

```
#turtle.setup()
wn=turtle.Screen()
wn.title("Carrera de tortugas")
width, height= 800, 800
wn.screensize(width, height)
```

Pintamos el mar:

```
t=turtle.Turtle() #t dibuja nuestro entorno
t.hideturtle()


t.pencolor("blue")
t.fillcolor('blue')
t.pensize(5)
t.begin_fill()

t.goto(-350, 0)
t.goto(350, 0)
t.goto(350, -350)
t.goto(-350, -350)
t.goto(-350, 0)
t.end_fill()
```

Pintamos la arena:

```
t.pencolor("#FFA500")
t.fillcolor('#FFA500')
t.pensize(5)
t.begin_fill()

t.goto(-350, 0)
t.goto(350, 0)
t.goto(350, 350)
t.goto(-350, 350)
t.goto(-350, 0)
t.end_fill()
```

Dibujamos la meta:

```
t.hideturtle()
t.penup()
t.goto([330,-360])
t.color('white')
style = ('Courier', 40, 'bold')
t.write('X', font=style, align='center')
```

Mostramos los índices de refracción de la tierra y el agua:

```
#indice agua
t.goto([175,-45])
style = ('Courier', 25, 'bold')
t.write('n2 = 1.0',font=style)

#indice tierra
t.goto([175,-5])
t.color('black')
t.write('n1 = '+ str(n),font=style)
```

Creamos las nuevas tortugas y las metemos en una lista para facilitar su control mediante bucles:

```
a,b,c,d,e=turtle.Turtle(), turtle.Turtle(), ...
tortugas=[a,b,c,d,e]
for tortuga in tortugas: tortuga.hideturtle()
```

Llevamos las tortugas a la salida (pasando por meta, para dejar dibujada su trayectoria) y las orientamos en los ángulos fijados en la lista homónima:

```
for tortuga in tortugas:
    tortuga.penup()
    tortuga.shape('turtle')
    tortuga.showturtle()
    tortuga.color(colores.pop())
    tortuga.pensize(3)
    tortuga.turtlesize(1, 1, 2.4)
    tortuga.goto(meta)
    tortuga.pendown()
    angulo=angulos.pop()
    tortuga.goto(-340-340/tan(radians(angulo)),0)
    tortuga.goto(salida)
    tortuga.setheading(angulo)
```

Diálogo para que el usuario decida cuándo empieza la carrera:

```
messagebox=tkinter.messagebox
info = messagebox.showinfo('Carrera de tortugas'...
```

Fijamos la velocidad de la animación y el tamaño del trazo, y creamos la lista para diferenciar qué tortugas están en tierra y cuáles han llegado ya al agua (necesario para modificar su velocidad):

```
for tortuga in tortugas:
  tortuga.pensize(10)
  tortuga.speed(5)

tortugas_tierra=tortugas[:]
tortugas_agua=[]
```

Bucle que controla la carrera:

```
while tortugas_agua!=[] or tortugas_tierra!=[]:
  for tortuga in tortugas_tierra:
    tortuga.forward(1)
  for tortuga in tortugas_agua:
    tortuga.forward(n)

  #cuando las tortugas llegan al agua, giran:
  posiciones_y_tierra=[tortugalenta.ycor() for tortugalenta...
  if tortuga_llega_agua!=[]:
    tortu_que_gira=tortugas_tierra[tortuga_llega_agua[0]]
    tortu_que_gira.setheading(tortu_que_gira.towards(330...
    #quitamos la tortuga de la lista de las que estan en ...
    tortugas_agua.append(tortugas_tierra.pop(tortuga...


  #las tortugas ganadoras dejan de avanzar
  posiciones_y_agua=[tortugarapida.ycor() for  ...
  if tortuga_gana!=[]:
    tortu_que_gana=tortugas_agua[tortuga_gana[0]]

    tortu_que_gana.penup()
    tortu_que_gana.goto(320,posicion_y_podio)
    tortu_que_gana.turtlesize(3, 3, 7.5)
    tortu_que_gana.stamp()
    posicion_y_podio=posicion_y_podio-30

    tortu_que_gana.stamp
    #quitamos la tortuga de la lista de las que estan en mar
    tortugas_agua.pop(tortuga_gana[0])
```

Avisamos al usuario de que ha terminado la carrera y cerramos *turtle:*

```
messagebox=tkinter.messagebox
info = messagebox.showinfo('Carrera de tortugas'...

turtle.bye()
```

Incluimos un pequeño bucle que permite comprobar qué tortuga se acerca más a la ley de Snell:

```
#hacemos una copia de los ángulos de cada tortuga ...
angulos_normal=[90-abs(angulo) for angulo in angulos]
angulos_normal.reverse() #los ordenamos ...
tortugas_nombres=['roja','amarilla','verde','azul','morada']

for angulo_inc in angulos_normal:
    X1=Y1*tan(radians(angulo_inc)) #distancia recorrida ...
    angulo_ref=round(degrees(arctan((X-X1)/Y2)),1)
    nombre=tortugas_nombres.pop(0)
    print('\n\nLa tortuga {} incide en el agua con un ...
    print('\nPara la tortuga ...

quit() #turtle no parece cerrarse correctamente...
```

Referencias

[1] Recogidos en el BOE, Real Decreto 1105/2014, de 26 de diciembre.

[2] Sexto estándar del Bloque IV de Física de 2.º de Bachillerato: *Utilizar el Principio de Huygens para comprender e interpretar la propagación de las ondas y los fenómenos ondulatorios.*

[3] *La habitación de Fermat,* dirigida por Luis Piedrahíta y Luis Sopeña. Enlace a la película en IMBD: https://www.imdb.com/title/tt1016301/.

[4] La velocidad de la luz es insuperable en el vacío, pero realmente aquí estamos tratando con medios materiales, con $n > 1$. Realmente sí es posible desplazarse más rápido que la luz en un medio material (sin alcanzar nunca $v = c$), y cuando ocurre se pueden dar fenómenos muy curiosos, como la radiación Cherenkov.

[5] H. Gaertner, "Huygens' Principle: A case against Optimality", *Behavioral and Brain Sciences* 26(6), 2003, pp. 779- 781.

[6] P. Schoemaker, "Huygens Versus Fermat: No Clear Winner", *Behavioral and Brain Sciences* 26(6), 2003, pp. 781- 782.

[7] J. Kimball y H. Story, Fermat's Principle, Huygens' Principle, Hamilton's Optics and Sailing Strategy", *European Journal of Physics* 19(1), 1998, pp. 15-24.

[8] S. Ling, J. Sanny y W. Moebs, *University Physics: Volume 3.* (Houston, Texas, OpenStax, Rice University, 2016). (Acceso libre en https://bit.ly/2OZ0c5K).

[9] C. Huygens, *Traite de la lumiere* (1690).

[10] R. Feynman, R. Leighton, y M. Sands, *The Feynman Lectures On Physics* (San Francisco, Calif., Addison-Wesley, 1964). (Acceso libre en https://www.feynmanlectures. caltech.edu).

[11] El caso del principio de Fermat es análogo al de Hamilton: es conocido como principio de mínima acción, cuando realmente propone la estacionaridad de la acción. M. Anderson, M. Hadi y U. Deta, "Fermat's Principle and Hamilton's Principle: Does a Least Action Take a Least Time for Happening?", *Journal of Physics: Conference Series,* 1467, 2020, p. 012038.

[12] A. L. Aina, *La llamada de Fermat* (2017). (Acceso libre en: https://webs.ucm.es/info/gioq/docencia/ MaterialesDocentes/ LallegadadeFermat.pdf).

[13] Documentación del módulo *turtle* (Disponible en https://docs.python.org/3.3/library/turtle.html? highlight=turtle).

[14] Una manera sencilla de instalar el módulo es tecleando: `python3 -m pip install {user PythonTurtle PythonTurtle.`