

Y separó la luz de las tinieblas. Génesis de Montecarlo.

María Sebastiá Fortes y Antonio Pariente Granero

Diciembre 2020

# Índice

- 1 Abstract
- 2 Antecedentes
- 3 Marco Teórico
  - Naturaleza de la luz
  - Resplandor
  - BRDF
  - Ecuación del renderizado
- 4 Método de Montecarlo
- 5 Simulación
- 6 Bibliografía
- 7 Materiales y métodos

# Abstract

El trazado de rayos es un método ampliamente utilizado en la industria audiovisual, pues la luz un componente fundamental de cómo vemos el mundo.

Nuestro objetivo es comprender las leyes que gobiernan la iluminación de un entorno para tratar así de modelizar el trazado de los rayos de luz, con la consecuente participación de los métodos de Monte Carlo. Para visualizar mejor este modelo, se hará uso de una simulación por ordenador, así como algún ejemplo conocido.

En lo relativo al marco teórico se han usado diversas fuentes comentadas en la bibliografía. Para la simulación y el trazado de rayos, se ha usado el compilador oficial de Python en su versión 2.9.0, Microsoft Visual Studio CODE y la biblioteca pyRT, cuyo repositorio en GitHub, se encuentra en las referencias..

De la combinación Física y Probabilidad, se desprende un desglosamiento de la función que determina la forma de los rayos de luz al reflejar e iluminar la imagen, y el papel de Monte Carlo, se entiende esencial para esta tarea. Tras la simulación de este método en objetos, obtenemos una iluminación satisfactoria de la escena, que nos acerca a un entorno virtual más realista, como se buscaba desde un principio.

Después de evaluar los resultados, podemos afirmar que la combinación de la Física y la Probabilidad, nos lega una potente herramienta para hacer de los medios visuales, entornos mucho más parecidos al mundo real.

# Antecedentes

Desde que existe el hombre, existe el arte, y desde que hay arte, existe la intención, se podría decir necesidad, de representar el mundo y el entorno que lo rodea. Así, la siguiente pregunta nace naturalmente: ¿cuál es la finalidad de esto?, no estamos aquí para digresar sobre los motivos y finalidades del arte, pero no sería exagerado afirmar que una de las intenciones es preservar nuestro mundo, de la manera más fiel posible. Ya sea el realismo en cuadros o esculturas, el próximo paso parece que viene de la mano de la ciencia: simulación computacional.

Gracias a estos métodos, estamos un paso más cerca de emular el mundo que nos rodea e incluso de crear uno nuevo, con nuestras propias normas. Todo empieza pues con la luz, que da forma y color.



Figura: El Ángel Caído Cabanel(1868)

# Antecedentes



# Naturaleza de la luz

## Luz

Dualidad **onda-corpúsculo** { Onda electromagnética  
Rayo de Fotones

La luz, además viaja en línea recta y a velocidad constante, donde los fotones vibran en diferentes direcciones con distintas frecuencias que determinan la intensidad de la luz, a más frecuencia, más energía transporta. Para hablar de rayos de luz e iluminación, hay que saber qué es lo que queremos medir de los mismos.

## Resplandor

Vamos a hablar de **R**→ **Resplandor** o Brillo (Reflection):

Cantidad de energía emitida, reflejada, transmitida o recibida en una dirección concreta.

# BRDF

Y trabajaremos con la **BRDF**, es decir, "Función de distribución de reflectancia bidireccional". Esta función de distribución mide el **Resplandor Reflejado**, por una superficie opaca, en una dirección concreta, cuando le incide un rayo de luz con respecto de otra dirección.



Figura: Rayo de luz en la oscuridad

Hay que entender esto medido en  $[0, 1] \in \mathbb{R}$  como una proporción de energía reflejada. *La manera cuantitativa de medir la propiedad cualitativa brillo de un objeto.* Esta energía se mide como **flujo radiante**, que entendido como haz de luz, es análogo al flujo eléctrico o de aire.

# Resplandor

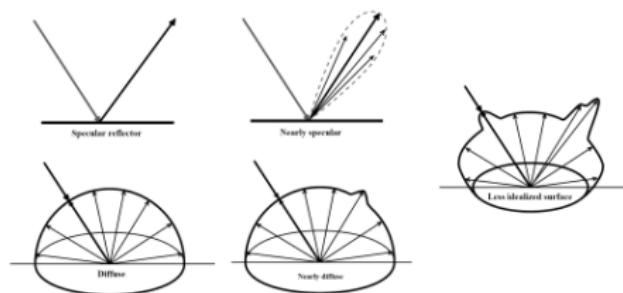


Figura: Izquierda, menos realista, derecha superficie menos idealizada

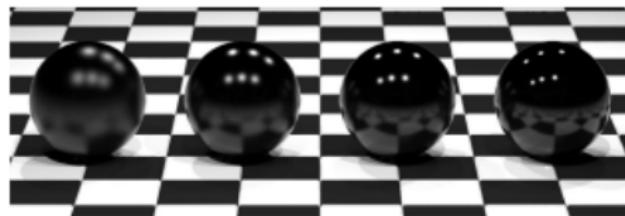


Figura: Índice BDFR decreciente, izquierda derecha

# Fórmula BDFR

Factores que debería influir:

- Geometría de la superficie (Macro-molecular en este modelo)
- El ángulo de incidencia
- La energía que incide

## Fórmula BDFR

$$f_r(\vec{x}, w_i, w_r) = \frac{\partial L_r(w_r)}{\partial E(w_i)}$$

Factores que influyen:

- $\vec{x}$  es el punto de la superficie.
- $w_i, w_r$  son los ángulos de **incidencia** y **reflexión**, en ángulo sólido.
- $L_r(w_r)$  es el **Resplandor reflejado**, la energía que se ver reflejada.
- $E(w_i)$  es la **Irradiación**, flujo radiante incidente.

Se puede reescribir la fórmula anterior para que nos sea más útil:

## Fórmula BDFR

$$f_r(\vec{x}, w_i, w_r) = \frac{\partial L_r(w_r)}{L_i(w_i) * \cos(\theta_i) \partial w_i}$$

Los factores de la fórmula son:

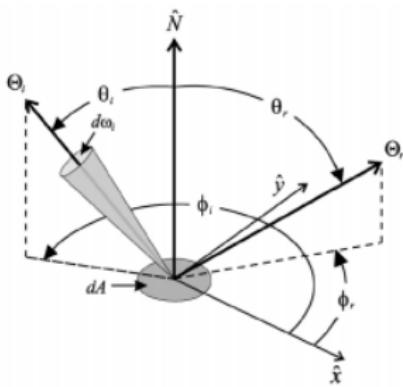


Figura: Esquema de ángulos

- $\vec{w}_i, \vec{w}_r$  son los ángulos de **incidencia y reflexión**, en ángulo sólido.
- $L_r(w_r)$  es el **Resplandor reflejado**, la energía que se ver reflejada.
- $L_i(w_i)$  es el **Resplandor incidente**
- $\cos(\theta_i)$  es el ángulo de incidencia respecto de la normal en  $\vec{x}$
- $\partial w_i$  es el ángulo proyectado en la esfera.

# Fórmula de Renderizado

## Ecuación de Renderizado

$$L_r(\vec{x}, \vec{w}_r) = \int_{\Omega_i} f_r(\vec{x}, \vec{w}_i \rightarrow \vec{w}_r) * L_i(\vec{x}, \vec{w}_i) * \cos(\theta_i) * \partial w_i$$

Como podemos observar, la función sólo tiene dos parámetros, por lo que al integrar sobre  $\Omega_i$ , estamos calculando el Resplandor reflejado en  $w_r$ , teniendo en cuenta todos los rayos incidentes en  $\Omega_i$ , que es el hemisferio superior de la esfera con vector polo  $\vec{n}$ .

Se trata de una fórmula muy intuitiva, pues es la proporción de energía reflejada dada por  $f_r(\vec{x}, \vec{w}_i \rightarrow \vec{w}_r)$  y la cantidad de energía que le llega, es decir  $L_i(\vec{x}, \vec{w}_i) * \cos(\theta_i) * \partial w_i$ , para todas las  $w_i$  del hemisferio superior.

# Ecuación de Renderizado

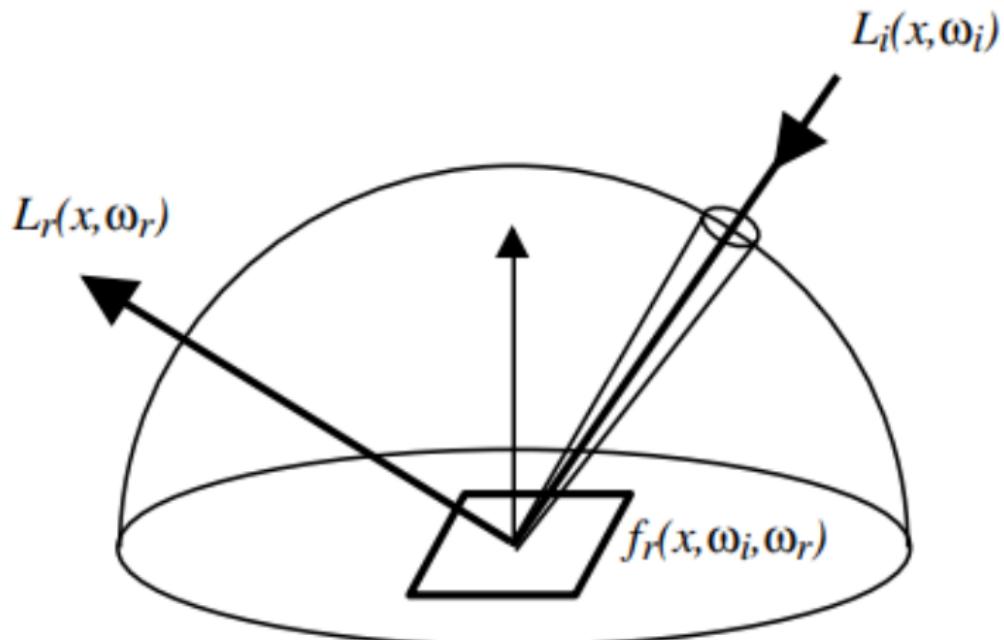


Figura: Ecuación de Renderizado

# Desarrollo de la Fórmula

Si seguimos razonando, más detenidamente en esta dirección, podríamos observar que a parte de la luz emisora (considerada puntual), que **la propia reflexión de otro punto del entorno** puede afectar al resplandor, incidente, así podemos separarlo en dos partes.

## Resplandor Incidente

$$L_i(\vec{x}, \vec{w}_i) = L_e(\vec{x}, \vec{w}_i) + L_r(\vec{x}', \vec{w}_i)$$

Esto es una forma de ecuación diferencial, pero nos faltaría entonces calcular el Resplandor Incidente de cada punto de la superficie en  $\vec{x}'$ . Para ello hay que cambiar los dominios de integración y considerar la integral **para todos los puntos de la superficie**.

# Desarrollo de la Fórmula

En lugar de considerar un ángulo infinitesimal, como variable de integración, haremos el cambio de variable que nos permita escoger un área infinitesimal, de cada punto que afecte a la superficie con el resplandor reglejado, es decir, cada punto que **vea** a  $\vec{x}$ .

## Ecuación de renderizado

$$L_r(\vec{x}, \vec{w}_r) = L_e(\vec{x}, \vec{w}_i) + \int_A fr(\vec{x}, w(\vec{x}, \vec{x}')) * \\ L_r(\vec{x}', w(\vec{x}, \vec{x}'), *G(\vec{x}, \vec{x}') * V(\vec{x}, \vec{x}') * \partial A'$$

Factores nuevos:

- $G(\vec{x}, \vec{x}')$  es el factor geométrico consecuencia del cambio de variable.
- $V(\vec{x}, \vec{x}')$  vale 1 si  $\vec{x}$  ve a  $\vec{x}'$  y 0 si no.
- $\partial A'$  es el área en  $\vec{x}'$

## Desarrollo de la fórmula

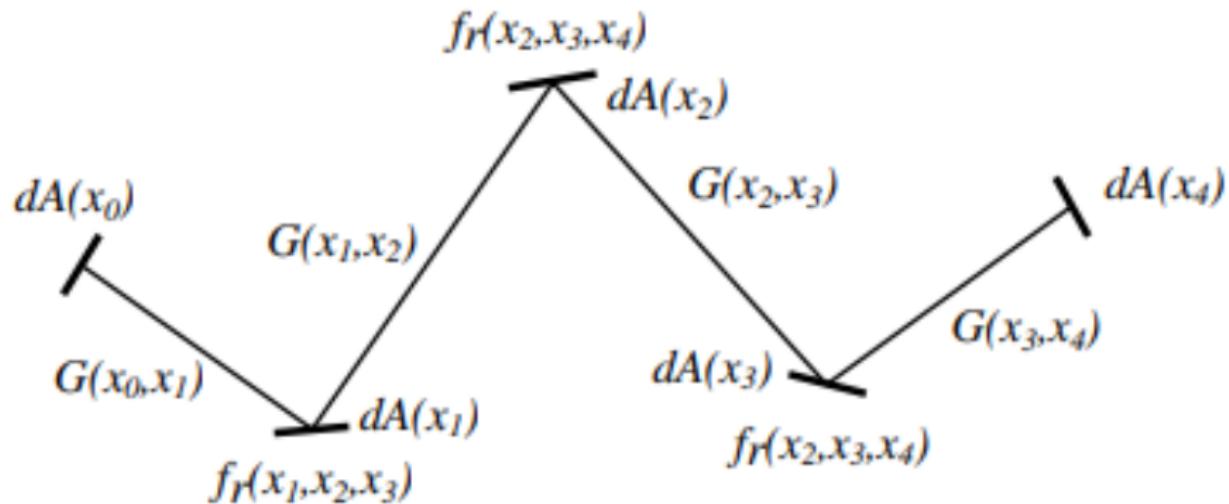


Figura: Ecuación de Renderizado completa

# Cadenas de rayos

$$L^0 = L_e$$

$$L^1 = L_e + K \circ L^0 = L_e + K \circ L_e$$

...

$$L_e + K \circ L^{n-1} = \sum_{i=0}^n K^i \circ L_e$$

Donde  $K$  es el operador de la integral de la ecuación de Renderizado. Esta serie es el planteamiento teórico, pero para calcular estas cadenas de integrales, necesitaremos una manera sencilla de computar una aproximación. Es aquí donde el método que da nombre a este trabajo juega su papel.

# Método de Montecarlo

Fue desarrollado en 1944 en Laboratorio Nacional de Los Álamos, por los matemáticos John Von Neumann y Stanislaw Ulam. La simulación de Montecarlo, le debe el nombre al famoso casino del principado de Mónaco, por ser el ejemplo más sencillo que permite generar números aleatorios.



Figura: Ruleta de un casino

El Método de Montecarlo es un método de integración numérica usado para aproximar integrales definidas,  $A = \int_D g(x)dx$ , sobre un dominio  $D \in \mathbb{R}^m$  complejas o costosas de calcular.

# Método de Montecarlo

Si hablamos de la esperanza de una función sabemos que:

$$E[g(X)] = \int_D g(x) * f_X(x) dx \approx \frac{1}{N} \sum_{i=1}^N g(x_i)$$

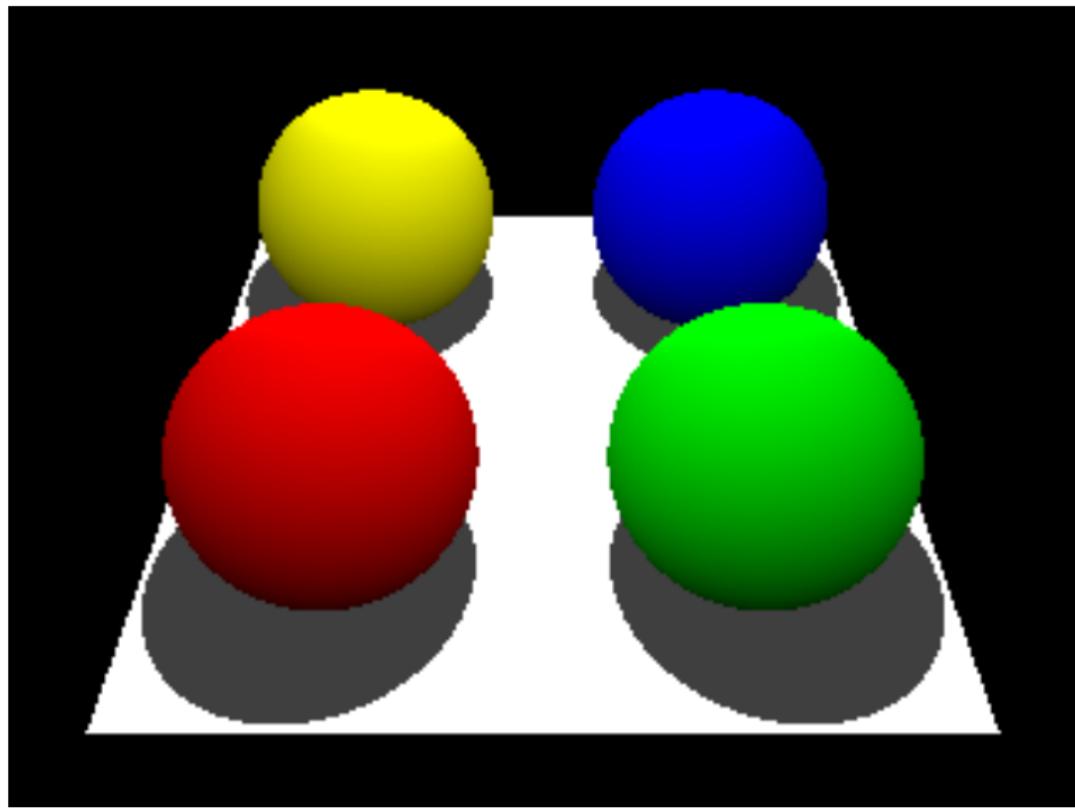
El Método de Montecarlo se basa en este hecho para estimar el valor de la integral definida:

$$A = \int_D g(x) dx \approx \frac{1}{N} \sum_{i=1}^N \frac{g(x_i)}{f_X(x_i)}$$

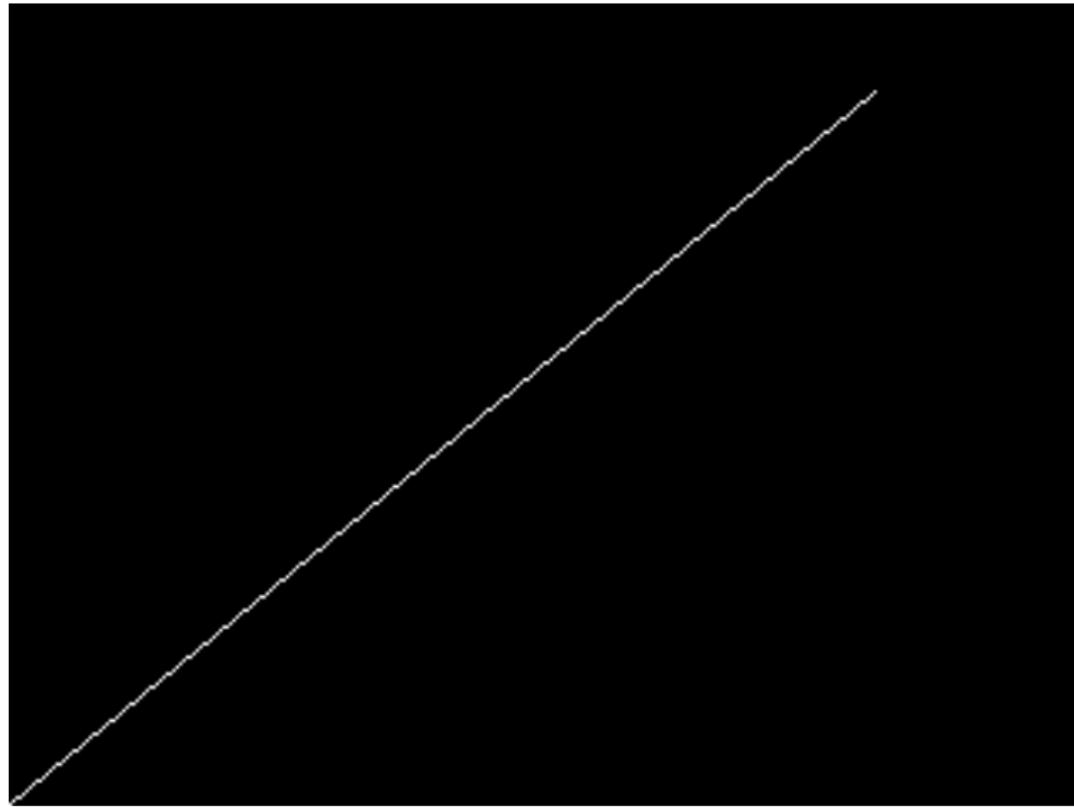
En el caso de tomar las muestras sobre una distribución uniforme tendríamos:

$$f_X(x_i) = \frac{1}{\int_D dx} \implies \frac{1}{N} \sum_{i=1}^N \frac{g(x_i)}{\int_D dx}$$

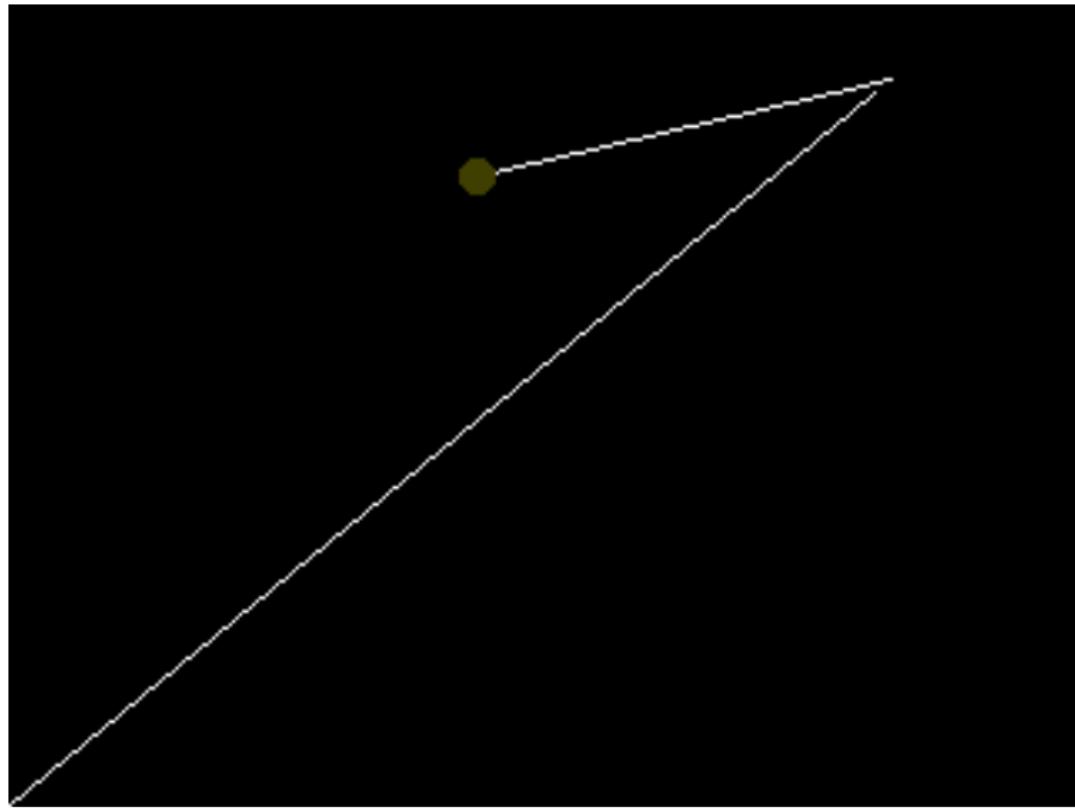
# Simulación



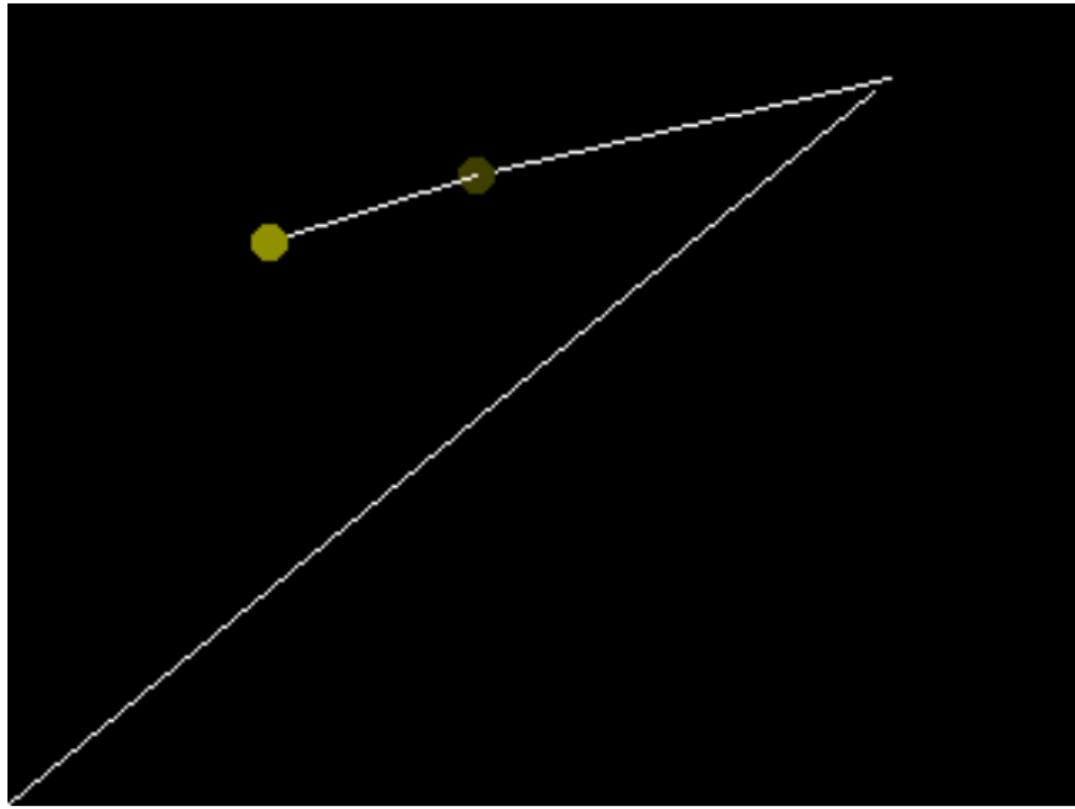
## Ejemplo 1



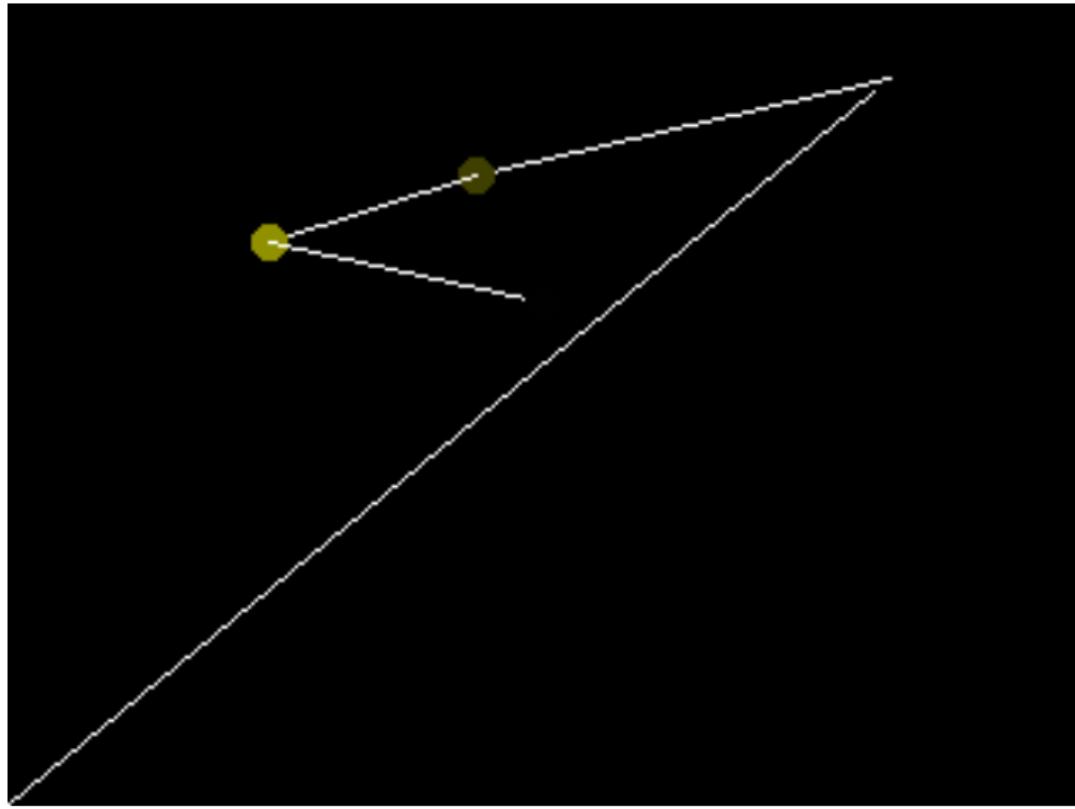
## Ejemplo 1



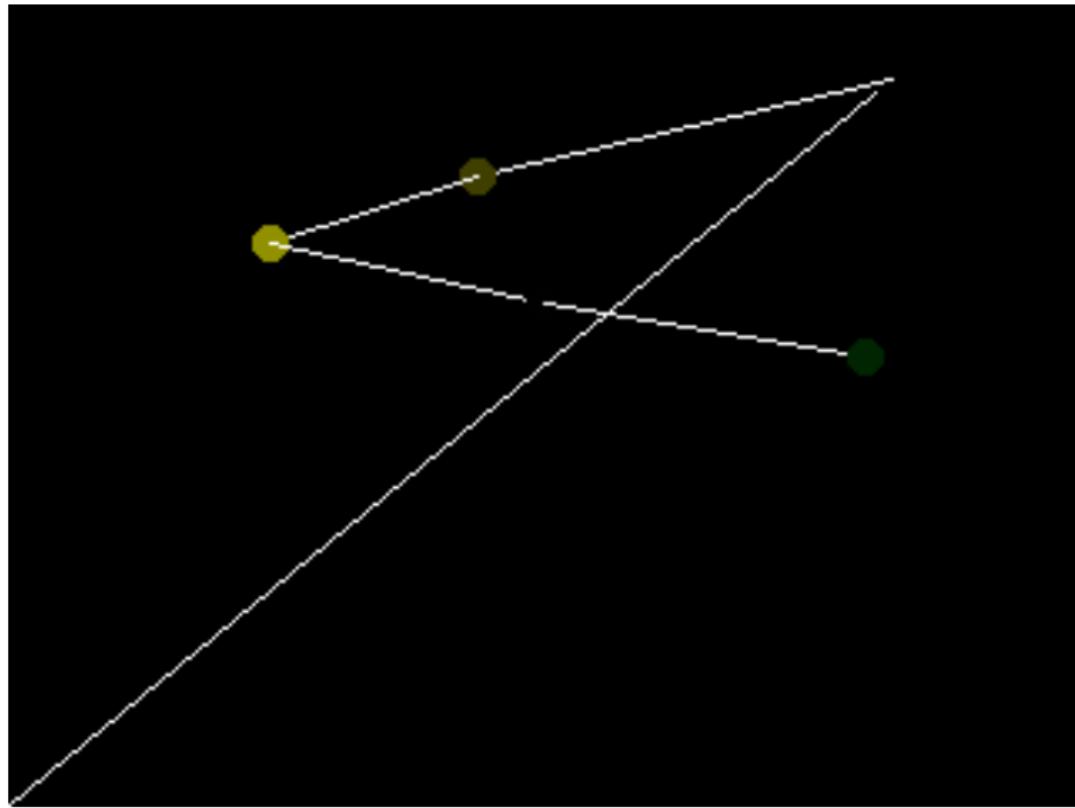
## Ejemplo 1



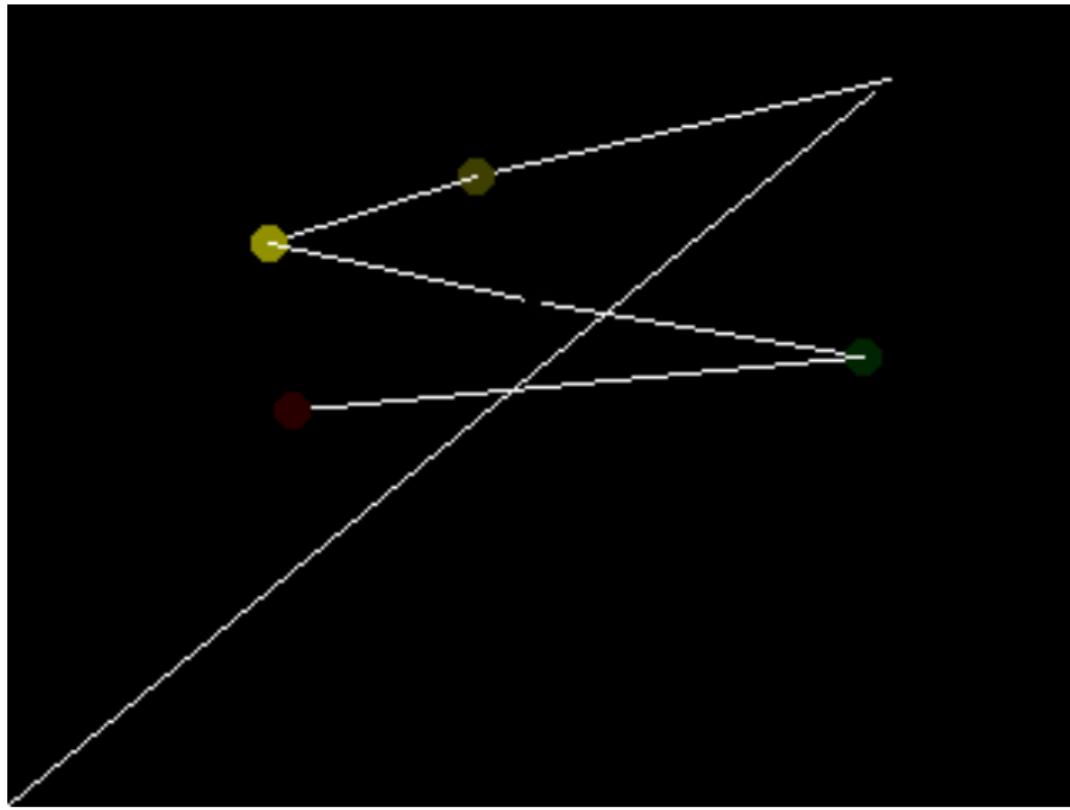
## Ejemplo 1



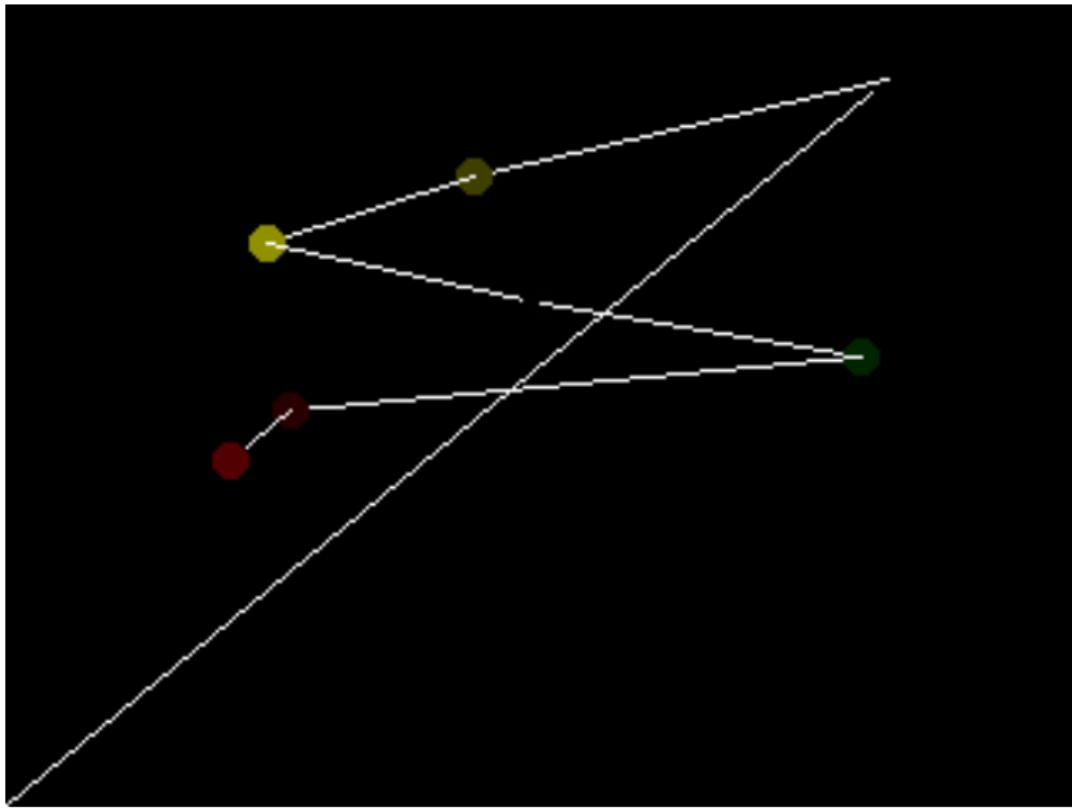
## Ejemplo 1



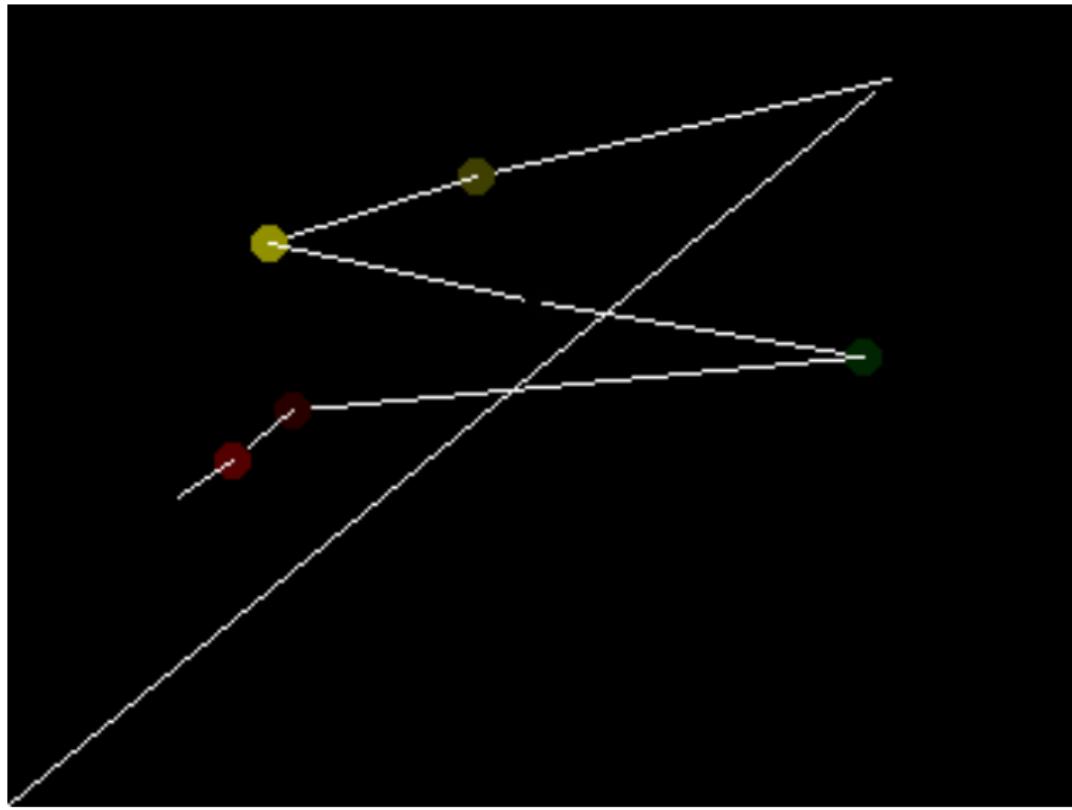
## Ejemplo 1



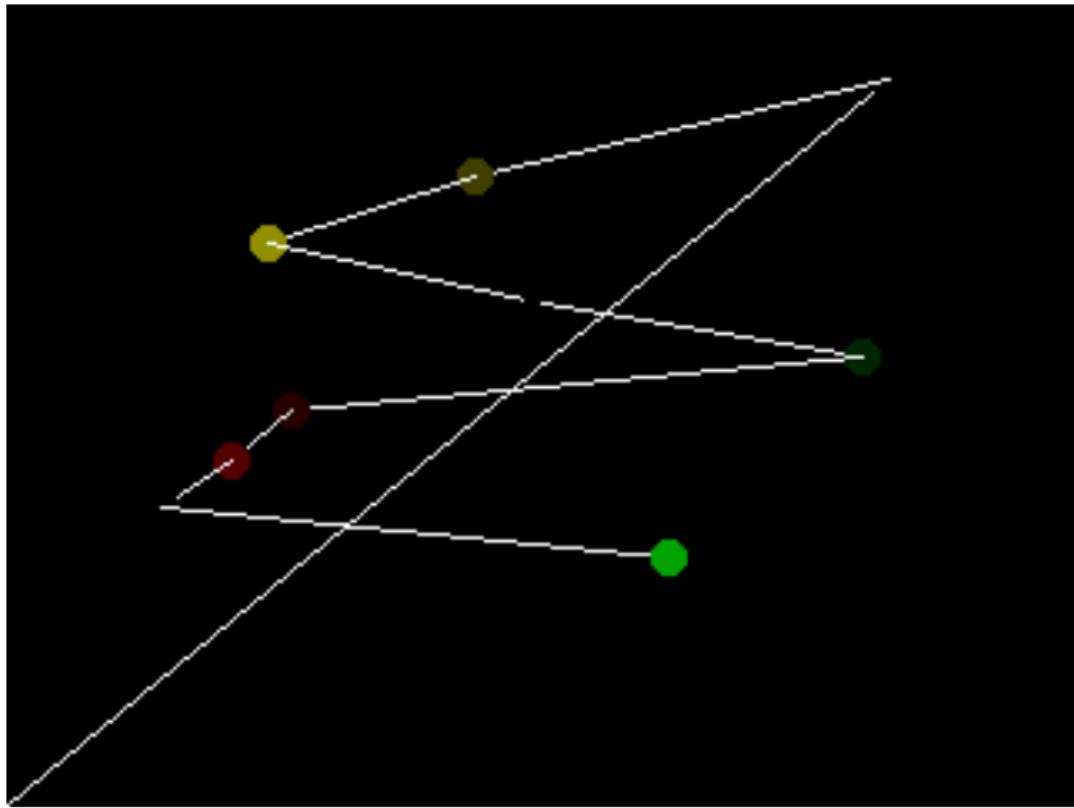
## Ejemplo 1



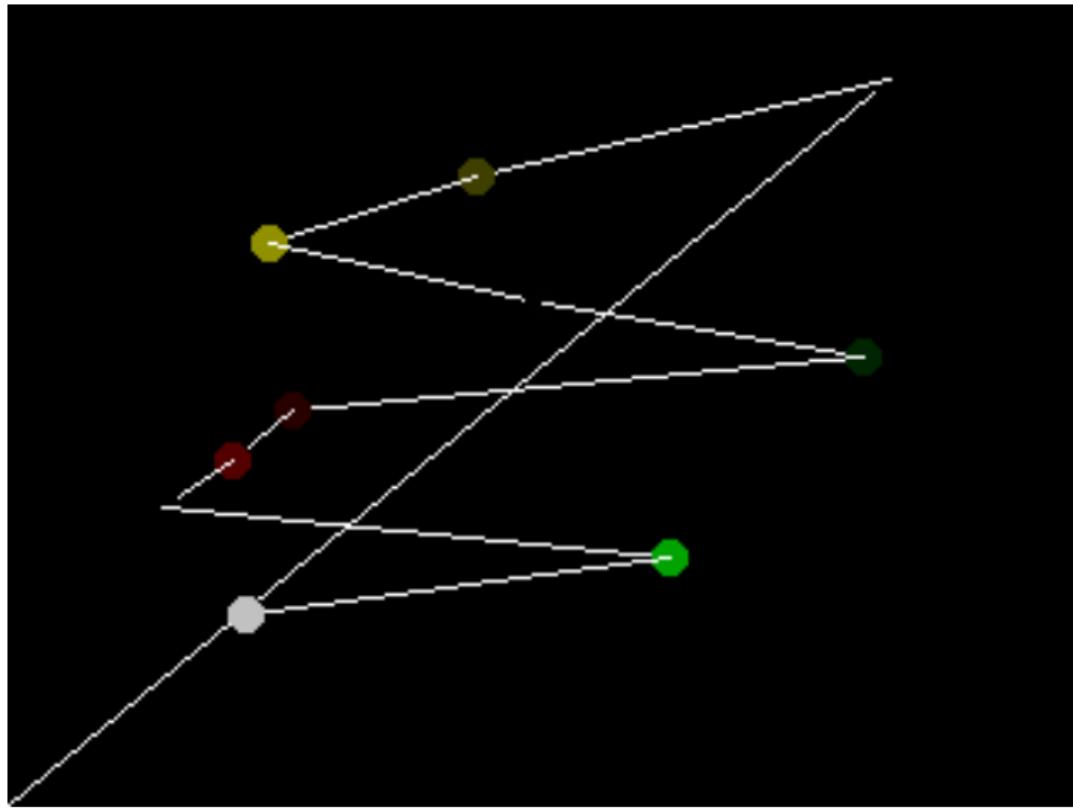
## Ejemplo 1



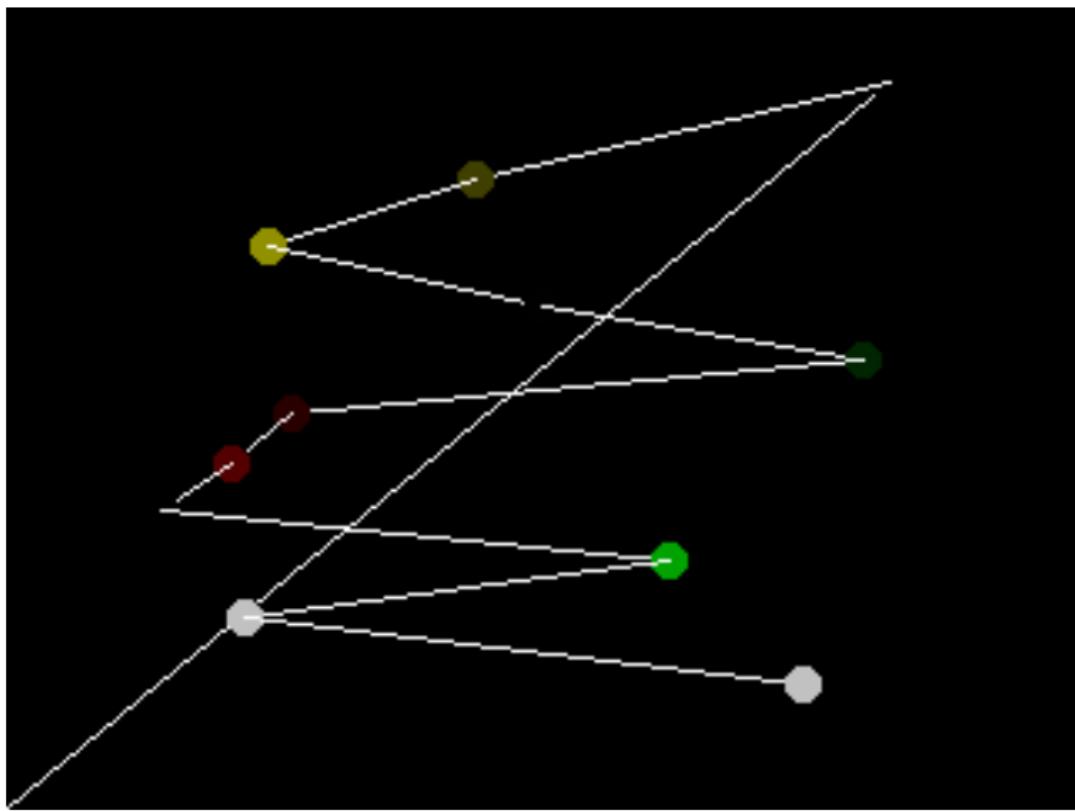
## Ejemplo 1



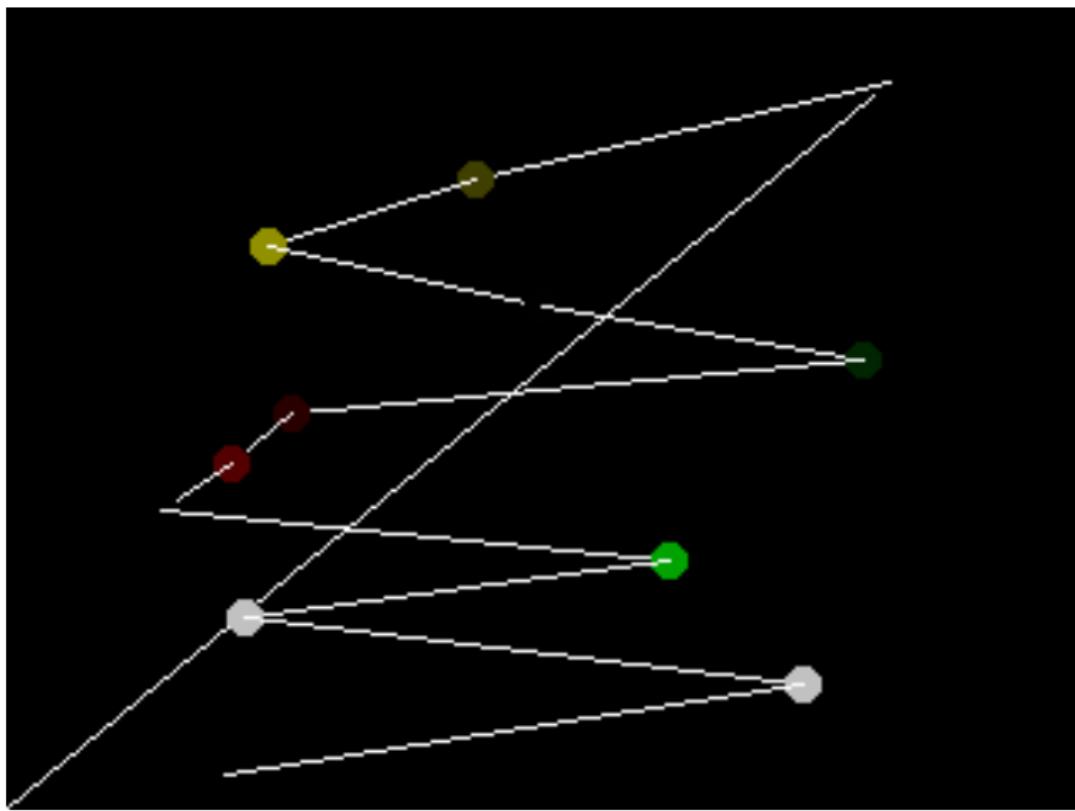
## Ejemplo 1



## Ejemplo 1



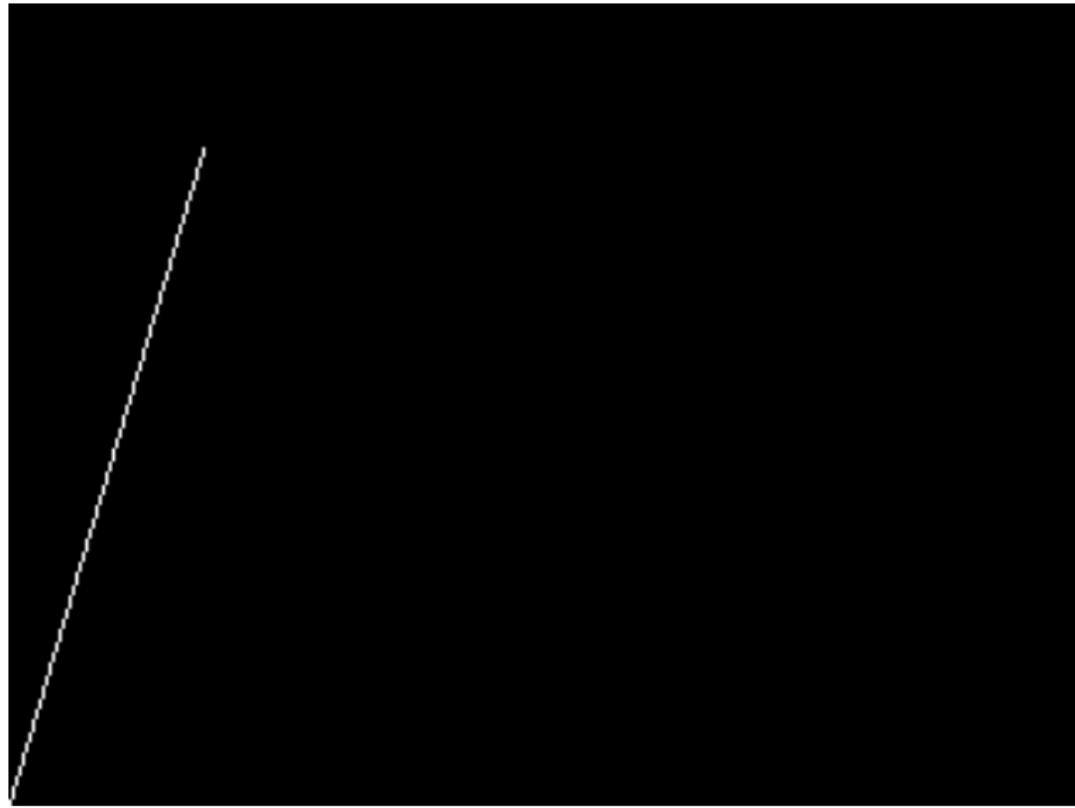
## Ejemplo 1



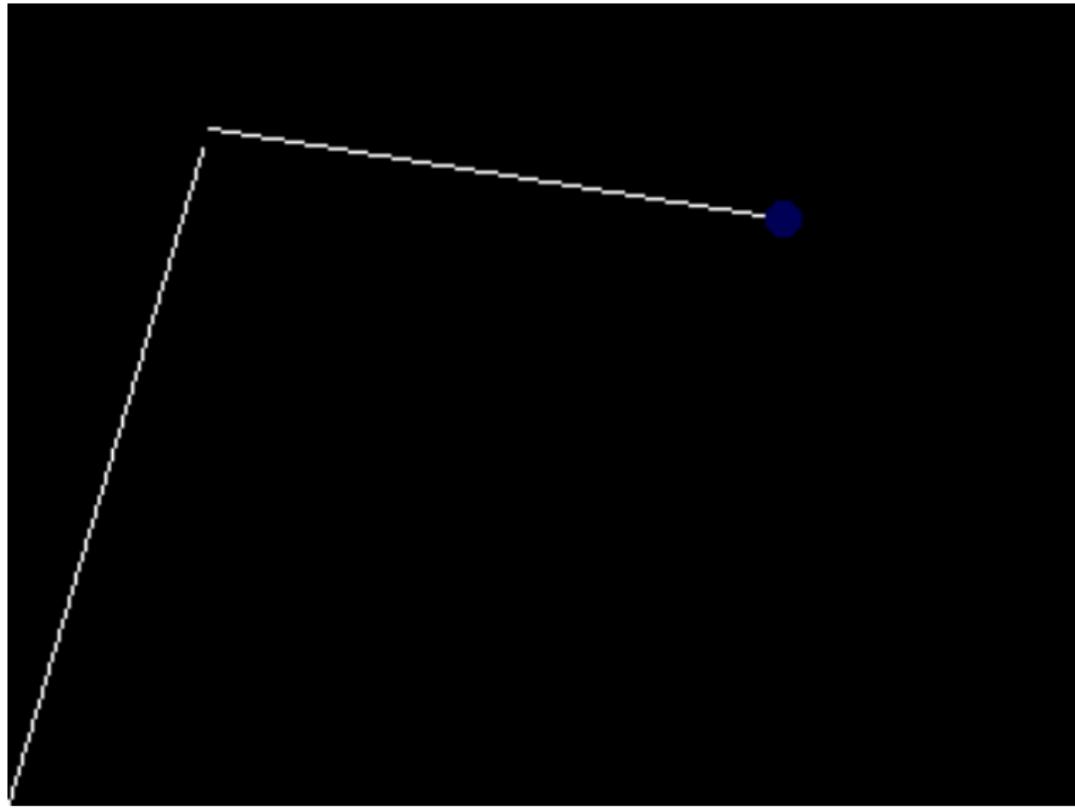
# Ejemplo 1



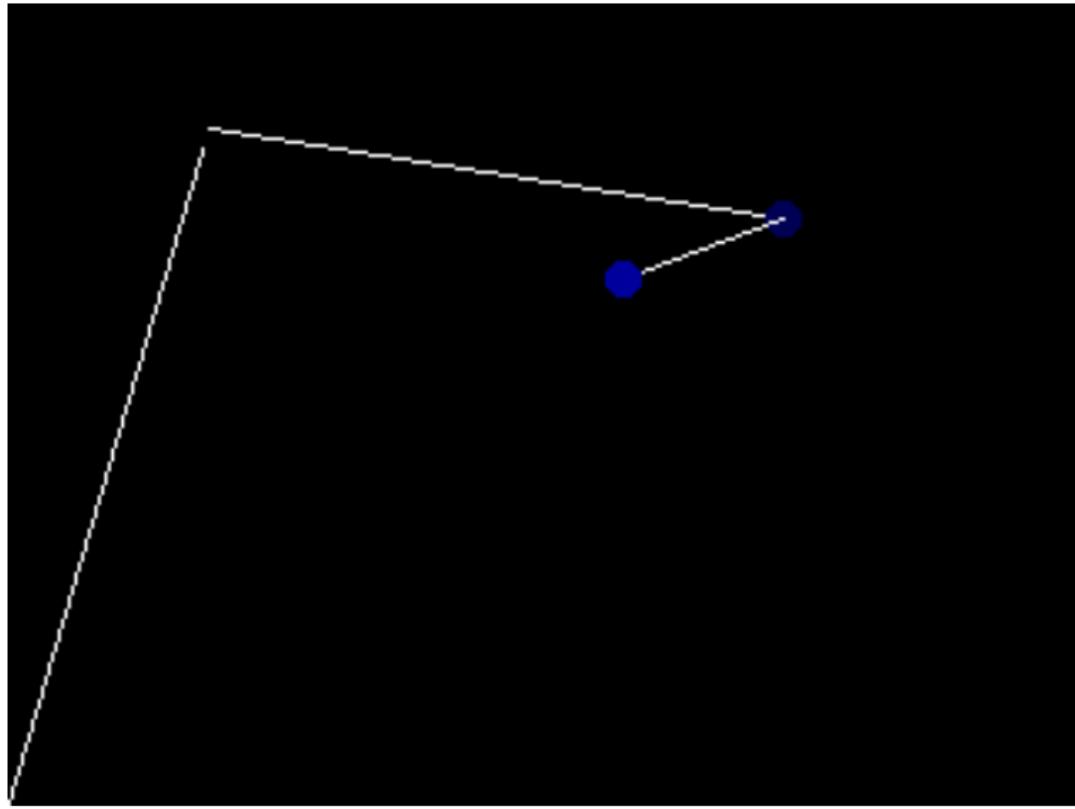
## Ejemplo 2



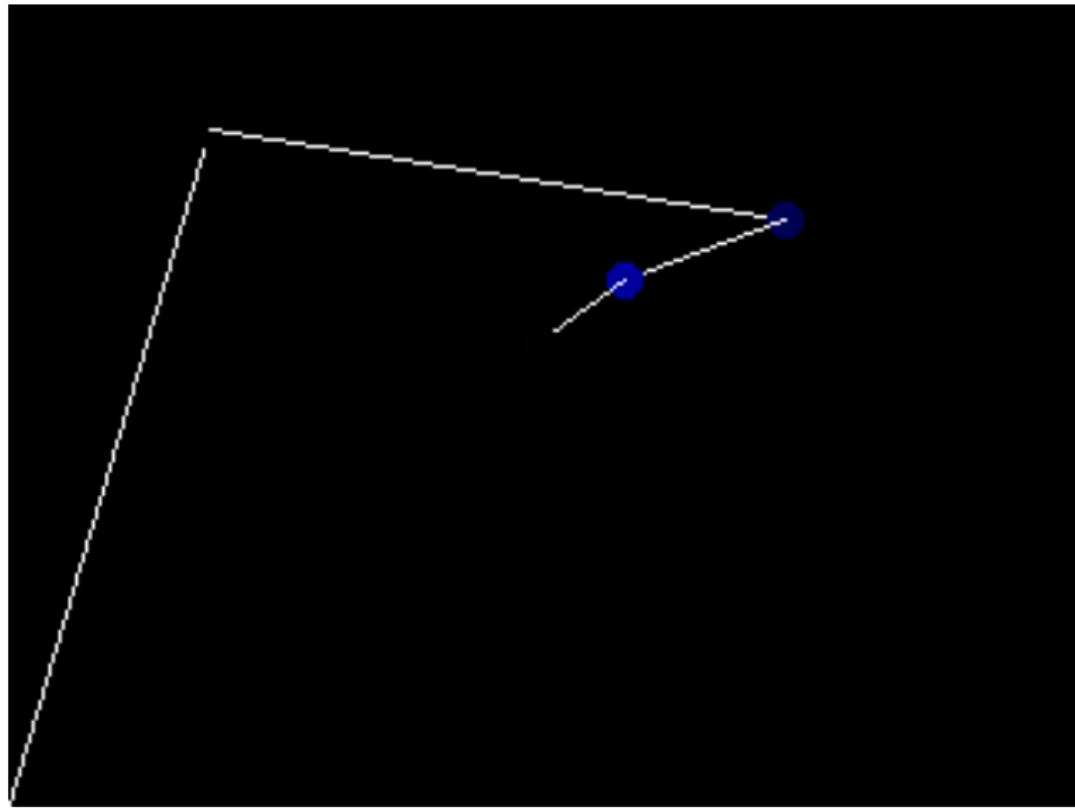
## Ejemplo 2



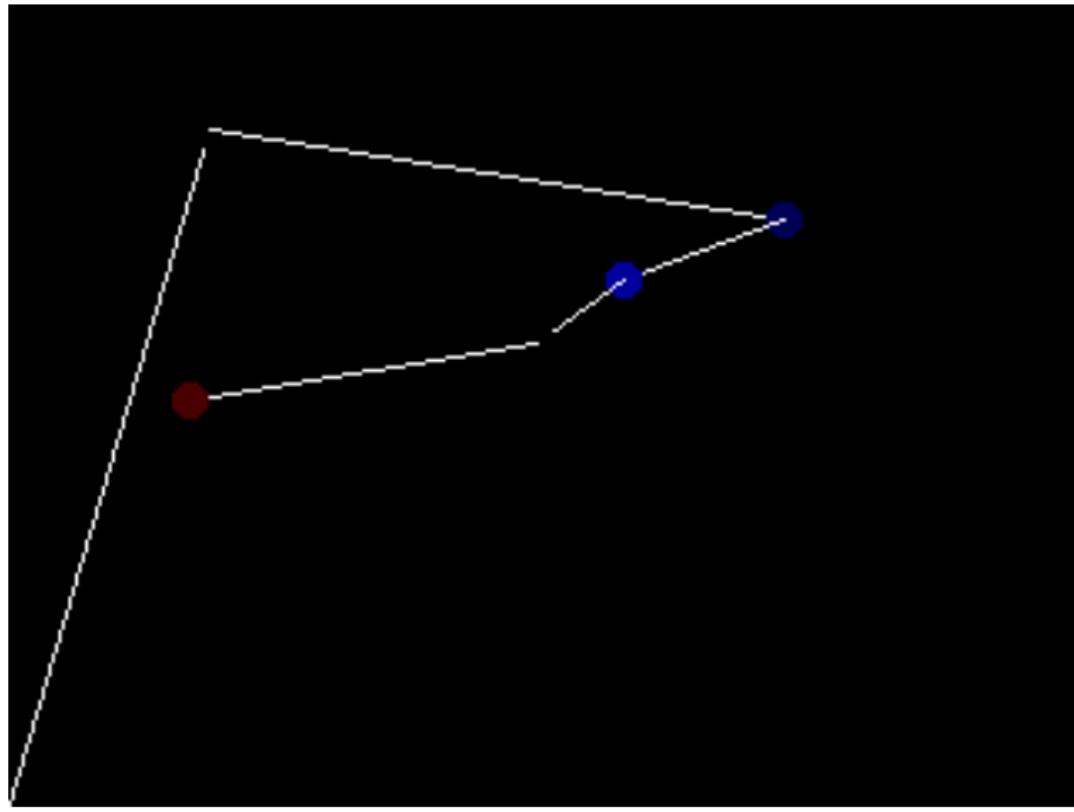
## Ejemplo 2



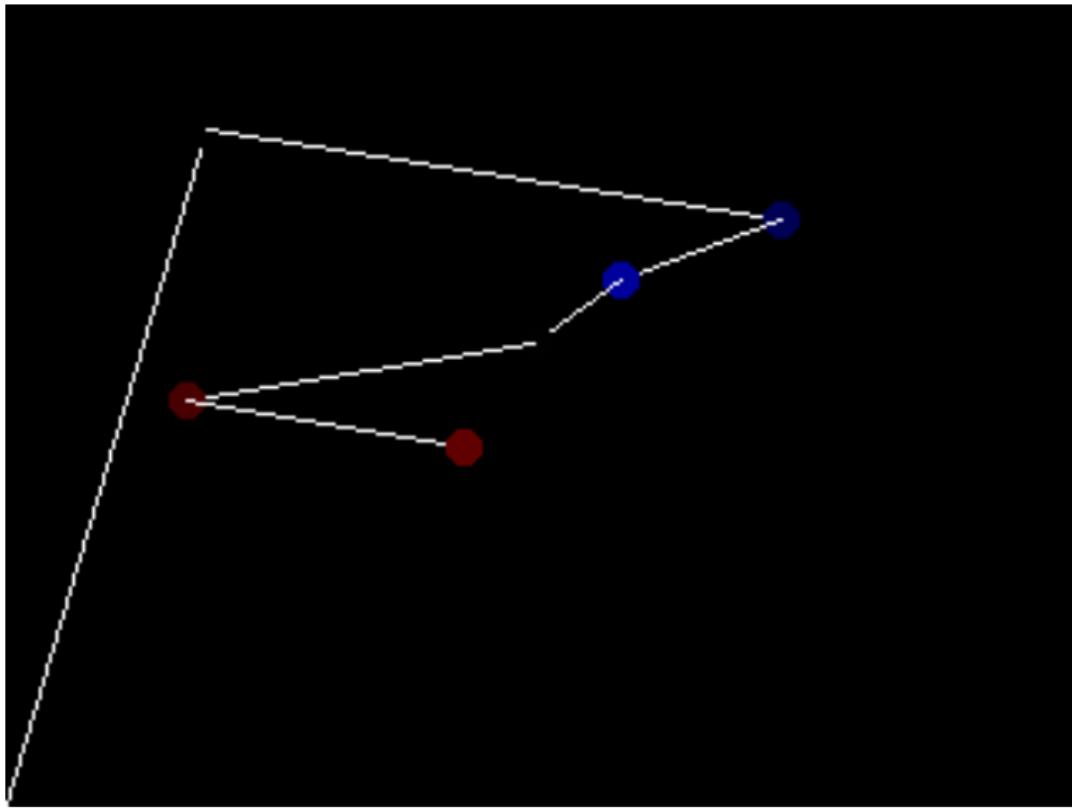
## Ejemplo 2



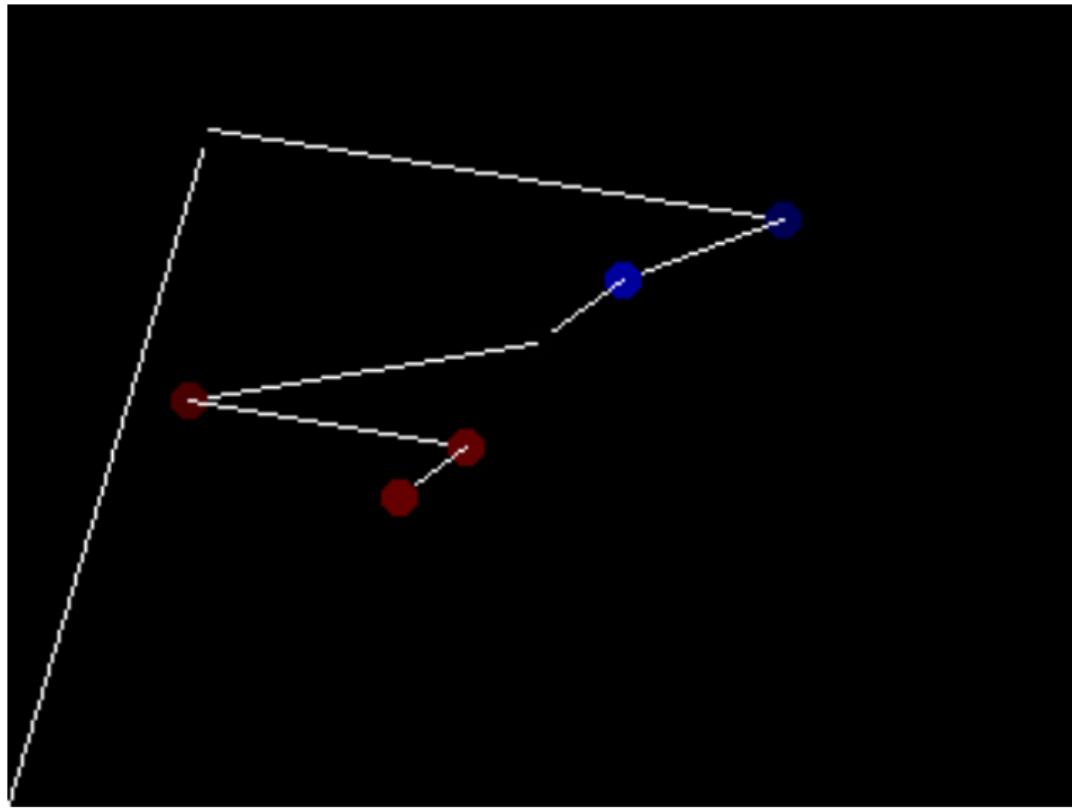
## Ejemplo 2



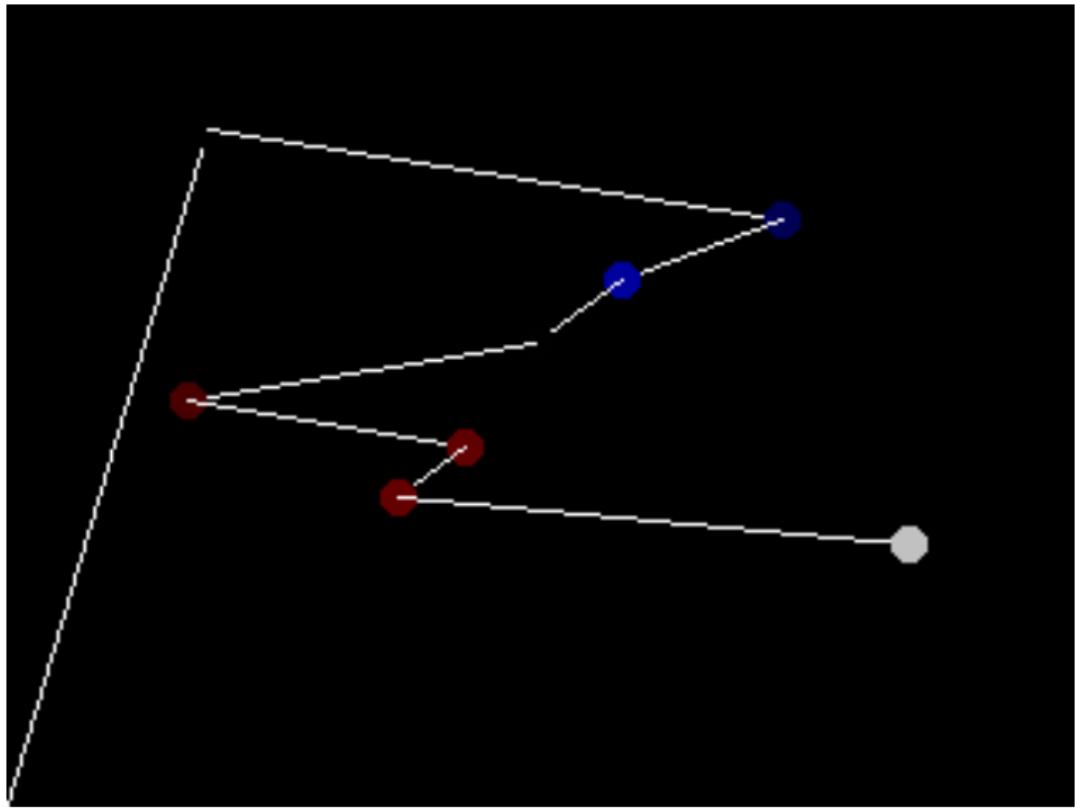
## Ejemplo 2



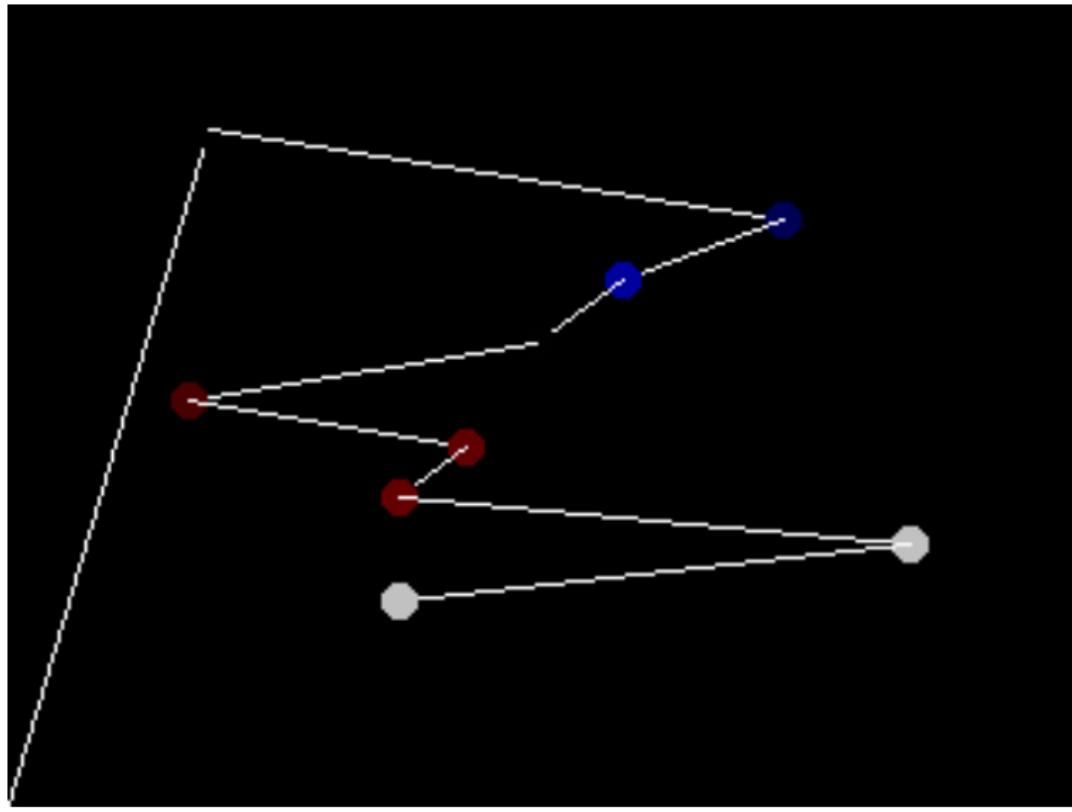
## Ejemplo 2



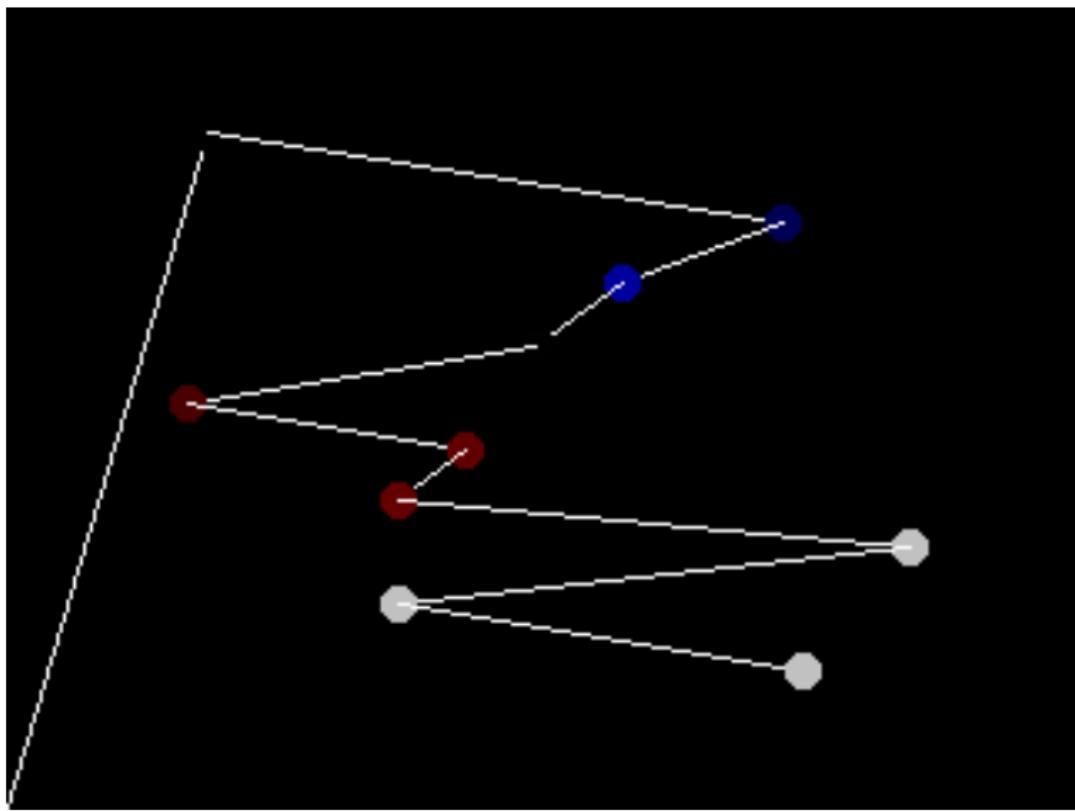
## Ejemplo 2



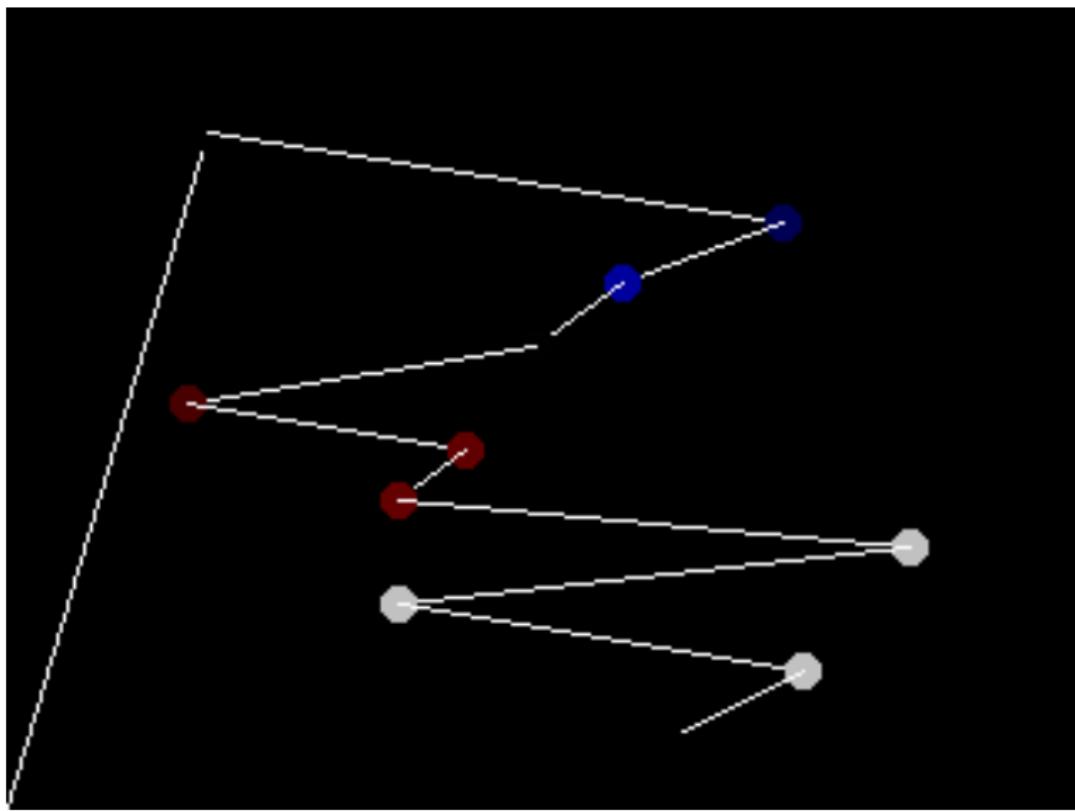
## Ejemplo 2



## Ejemplo 2



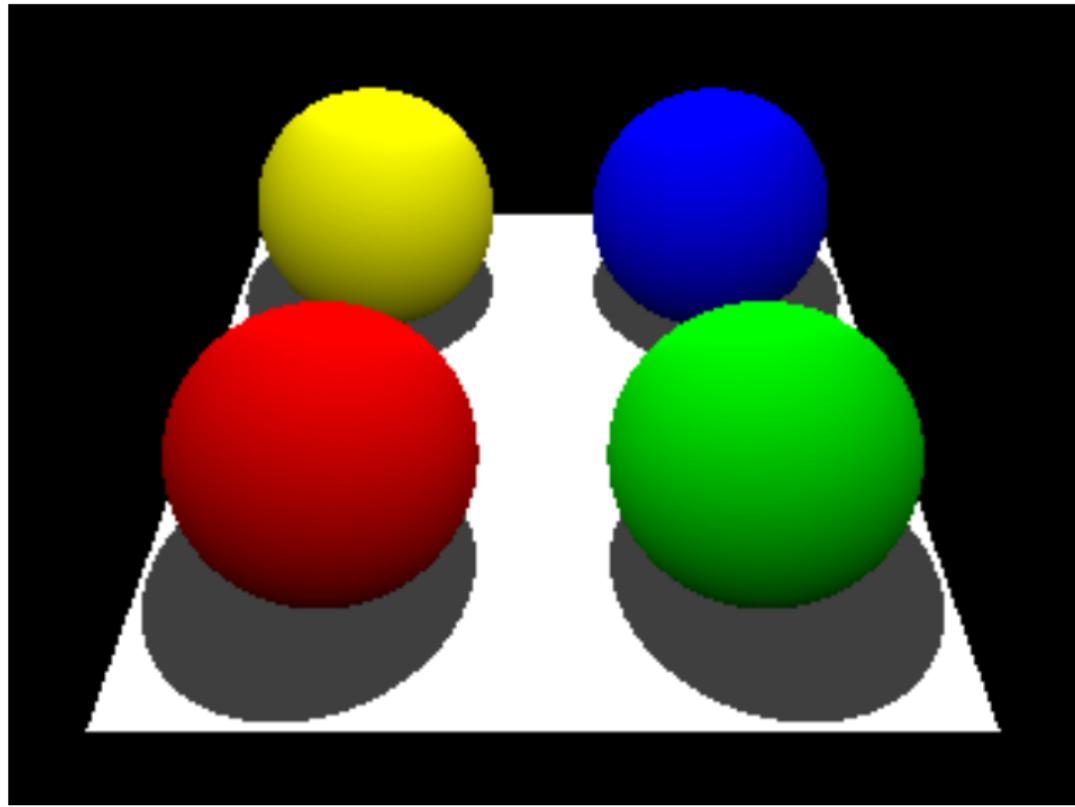
## Ejemplo 2



## Ejemplo 2



# Simulación



# Simulación

```
#####
PS C:\Users\apari\OneDrive\Documentos\Code & c:/Users/apari/AppData/Local/Programs/Python/Python39/.venv/Scripts/python.exe c:/Users/apari/OneDrive/Documentos/Code/hello.py
# Creating Renderer: Simple Raytracer
# Shadow Enabled
# RENDER STATISTICS#####
TIME FOR RENDERING: 11.11564588546753s
NUMBER OF PRIMARY RAYS: 76800
NUMBER OF SECONDARY RAYS: 0
NUMBER OF SHADOW RAYS: 55280
RAYS/s: 11882.350459965615
#####
```

Creating Renderer: Simple Raytracer

Shadow Enabled

RENDER STATISTICS

TIME FOR RENDERING: 11.11564588546753s

NUMBER OF PRIMARY RAYS: 76800

NUMBER OF SECONDARY RAYS: 0

NUMBER OF SHADOW RAYS: 55280

RAYS/s: 11882.350459965615

Andy



# Bibliografía

- -*Radiometry* -Author: Steve Marchner -Cornell University (2012) -[Url](#)
- -*Particle-Transport Simulation with the Monte Carlo Methods*  
-Author: L. Carter and E. Cashwell's -Los Alamos Scientific Lab., N.Mex. (USA) (1975)
- -*Bidirectional Reflectance: An Overview with Remote Sensing Applications Measurement Recommendations* -Author: James R. Shell - Rochester Institute of Technology, New York (2004) -[Url](#)
- -*Guía Básica de Conceptos de Radiometría y Fotomería* -Author: Prof. Dr. Emilio Gómez González -ESI- Universidad de Sevilla (2006) -[Url](#)
- *Implementación del algoritmo de path tracing en la GPU* -Author: Romo Ferrer, Joaquim -Universitat Pompeu Fabra (2014) -[Url](#)

# Materiales y métodos

-Python 3.9.0 –[Python.org](https://www.python.org) (Compilador)

-Microsoft Visual Studio Code – [MSVCode](https://code.visualstudio.com) (Editor)

-pyRt (GitHub)– [pyRT](https://github.com/ellisonbg/pyRT) (biblioteca)

\*\*Si estás usando python 3.9.0, las bibliotecas adicionales: pillow, matplotlib y numpy estarán instaladas de serie.

-Vídeo explicativo de Disney Studios –[Disney](https://disneystudios.com)

-Wikipedia

- [Rendering Equation](#)
- [BRDF](#)
- [Irradiance](#)
- [Radiance](#)
- [Radiant Flux](#)