🔎 **ANALISIS DEL CODIGO**

Este sketch dibuja una escena especial con:

* Un **fondo estrellado**.
* Una **nave espacial**.
* Varios **asteroides**.

Actualmente todo está en funciones sueltas (dibujarNave, dibujarAsteroide), sin usar clases. Eso funciona, pero dificulta la escalabilidad si después querés agregar más lógica (movimiento, colisiones, etc.). Pero este es un trabajo sencillo pensado para ser solo una escena ya que el objetivo principal del trabajo práctico es aprender a manejar los comandos básicos de Git.

🎨 **DISEÑO ORIENTADO A OBJETOS**

Para que sea más entendible podemos separar el código en clases para que mi escenario sea más entendible.

 **Clase Nave**

* Atributos:
  + x, y (posición)
  + tam (tamaño)
* Método:
  + dibujar()

 **Clase Asteroide**

* Atributos:
  + x, y (posición)
  + radio (tamaño base)
* Método:
  + dibujar()

 **Clase Juego** (o simplemente usar setup y draw)

* Atributos:
  + Una lista de **asteroides**
  + Una **nave**
* Métodos:
  + setup() (inicializa)
  + draw() (pinta fondo, estrellas, nave y asteroides)

**🔧 Mejoras básicas**

**Separar en clases (OOP):**  
Ahora todo está en funciones. Si lo pasás a clases (Nave, Asteroide), se podría poder manejar más fácil movimiento, colisiones y estados.

## 🎮 Mejoras de jugabilidad

**Movimiento de la nave y asteroides:**

* + La nave podría responder a teclas (keyPressed → mover arriba/abajo/izquierda/derecha).
  + Los asteroides podrían moverse lentamente hacia la izquierda para simular que la nave avanza.

**Colisiones:**  
Detectar si la nave choca con un asteroide (usando distancia entre puntos).

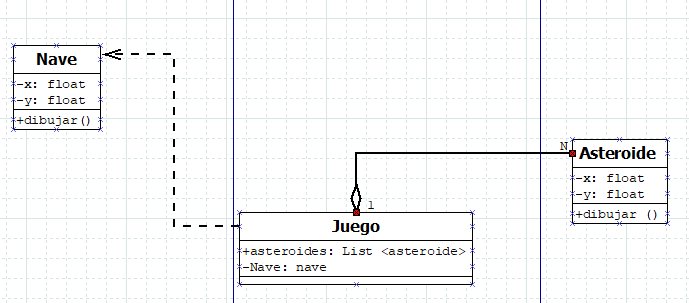
**Velocidad y física básica:**

* + Agregar velocidad (dx, dy) a los asteroides.
  + Hacer que roten un poco con cada frame.

## 📦 Mejoras de estructura

1. **Encapsular dibujar en métodos de objetos:**  
   En vez de pasar parámetros cada vez, la nave y los asteroides deberían saber su x, y, tamaño. Así solo llamás nave.dibujar();.
2. **Colecciones dinámicas:**  
   Usar ArrayList<Asteroide> en vez de un for fijo. Eso permite agregar o eliminar asteroides en tiempo real.
3. **Optimización:**  
   Si después agregás muchos objetos, podés limitar el uso de random() y cos/sin precalculando ángulos para los asteroides.

**UML**



## 📑 GDD (Game Design Document) – Versión inicial



### 1. **Título del juego**

🚀 Space Escape

### 2. **Género**

* Juego arcade espacial en 2D.

### 3. **Plataforma**

* PC (Processing/Java).
* Ventana: 800 x 600 píxeles.

### 4. **Descripción general**

El jugador controla una **nave espacial** que aparece en el costado izquierdo de la pantalla. En el espacio se encuentran **asteroides generados aleatoriamente** y un **fondo estrellado**.  
El objetivo es **sobrevivir el mayor tiempo posible** esquivando los asteroides.

### 5. **Mecánicas del juego**

* **Nave:**
  + Se dibuja en la posición inicial (100, height/2).
  + Puede moverse (en futuras versiones) con las teclas de dirección.
* **Asteroides:**
  + Se generan aleatoriamente en la pantalla, con distintos radios.
  + En la versión actual son estáticos; en versiones futuras se moverán hacia la nave.
* **Fondo:**
  + Estrellas aleatorias simulando el espacio.

### 6. **Reglas**

* Si la nave choca con un asteroide → Fin de la partida.
* El jugador intenta esquivar el mayor tiempo posible.

### 7. **Objetivos del jugador**

* Evitar los asteroides.
* Lograr una mayor puntuación (segundos sobrevividos o asteroides esquivados).

### 8. **Arte y estética**

* Estilo minimalista 2D.
* Colores:
  + Fondo negro (background(0)).
  + Estrellas blancas.
  + Asteroides en tonos grises (fill(120)).
  + Nave en tonos azulados y blancos.

### 9. **Sonido (futuro)**

* Música espacial de fondo.
* Efecto de sonido cuando la nave se mueve.
* Efecto de explosión al chocar con asteroide.

### 10. **Interfaz**

* Pantalla de inicio con título y botón Jugar.
* Pantalla de juego mostrando nave, asteroides y fondo.
* Pantalla de Game Over mostrando la puntuación.

### 11. **Progresión**

* Nivel inicial: asteroides lentos y pocos.
* Niveles avanzados: más asteroides, más rápidos, tamaños variados.

### 12. **Ideas de expansión**

* Implementar disparos para destruir asteroides.
* Power-ups (escudos, velocidad extra).
* Puntuación y ranking.