

Integrantes del equipo:

Nombre: Reyno Sofía

Legajo: 44.173.232

Email: [sofireynoo30@gmail.com](mailto:sofireynoo30@gmail.com)

Nombre: Bravo Joaquín

Legajo: 46.698.653

Email: [joaquinnazabravo@gmail.com](mailto:joaquinnazabravo@gmail.com)

Nombre: Fernandez Jeremias

Legajo: 45.622.641

Email: [jerefernandez114@gmail.com](mailto:jerefernandez114@gmail.com)

Grupo：12

**Índice**

1. **BIENVENIDOS AL JUEGO**

1.1 *Presentación*

1.2 *Modo de juego*

1.3 *Reglas del juego*

1. **CLASES**

2.1 *Clase Juego*  
 2.2 *Clase Personaje*  
 2.3 *Clase Isla*

2.4 *Clase Tortuga*

2.5 *Clase Gnomo* 2.6 *Clase Poder* 2.7 *Clase Fondo*

2.8 *Clase Casa*

1. **PROBLEMAS Y SOLUCIONES**
2. **CONCLUSIÓN**

**Introducción**

1. **BIENVENIDOS AL JUEGO**
   1. **Presentación**

El juego "Al rescate de los gnomos" es una experiencia interactiva donde el jugador controla a Totoro, quien debe rescatar a los gnomos mientras evita los peligros, especialmente a las tortugas, que pueden matarlo. El objetivo principal es rescatar a todos los gnomos antes de que Totoro muera o se pierdan demasiados gnomos.

**1.2 Modo de juego**

El jugador puede mover a Totoro hacia los lados, hacer que salte y lanzar una bola para atacar a las tortugas. Cada gnomo rescatado otorga puntos y se debe evitar perder gnomos o permitir que Totoro caiga al vacío o sea alcanzado por las tortugas..

**1.3 Reglas del Juego**

El jugador utiliza las teclas de dirección o WAD para mover a Totoro y el botón izquierdo del mouse para lanzar una bola (si no hay otra en pantalla). Gana cuando rescata a todos los gnomos. Y pierde si Totoro cae, si se pierden tres gnomos o si Totoro es alcanzado por una tortuga.

1. **CLASES**

Aquí se presenta una explicación general de cada clase implementada, incluyendo las variables de instancia y una breve descripción de los métodos principales.

**2.1** **Clase Juego**

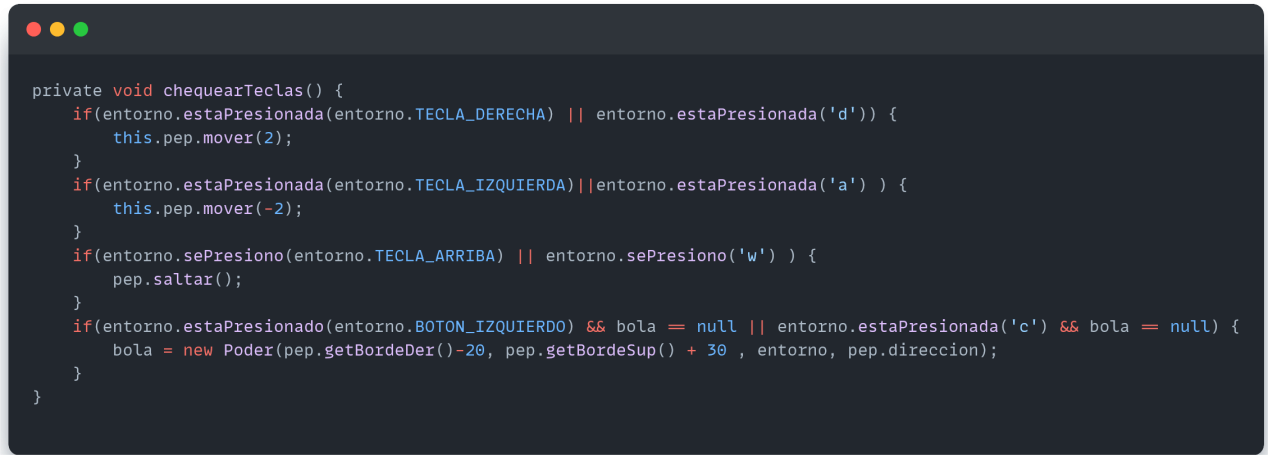
**Variables de instancia:**



En la ***clase Juego***, las variables de instancia incluyen el objeto *entorno*, que controla el entorno gráfico, y el *objeto pep*, una instancia de ***Personaje*** que representa al jugador. También están *islas* y *tortuga*, que son arrays de objetos ***Isla***y***Tortuga***, respectivamente, representando las plataformas y los enemigos. La variable *bola* es de tipo ***Poder*** y representa el proyectil que puede lanzar *Pep*. Los elementos visuales *fondo* y *casita* son instancias de ***Fondo*** y ***Casa***, y se utilizan para añadir detalles al entorno. Además, *gnomos* es un array de ***Gnomos*** que representan a los personajes que *Pep* debe rescatar. Los contadores *gnomosRescatados*, *gnomosPerdidos*, y *tortugasEliminadas* llevan un registro del progreso del jugador, mientras que *finJuego* es un booleano que indica si el juego ha terminado. Por último, *mensajeFinal* contiene el mensaje que se muestra al finalizar el juego.

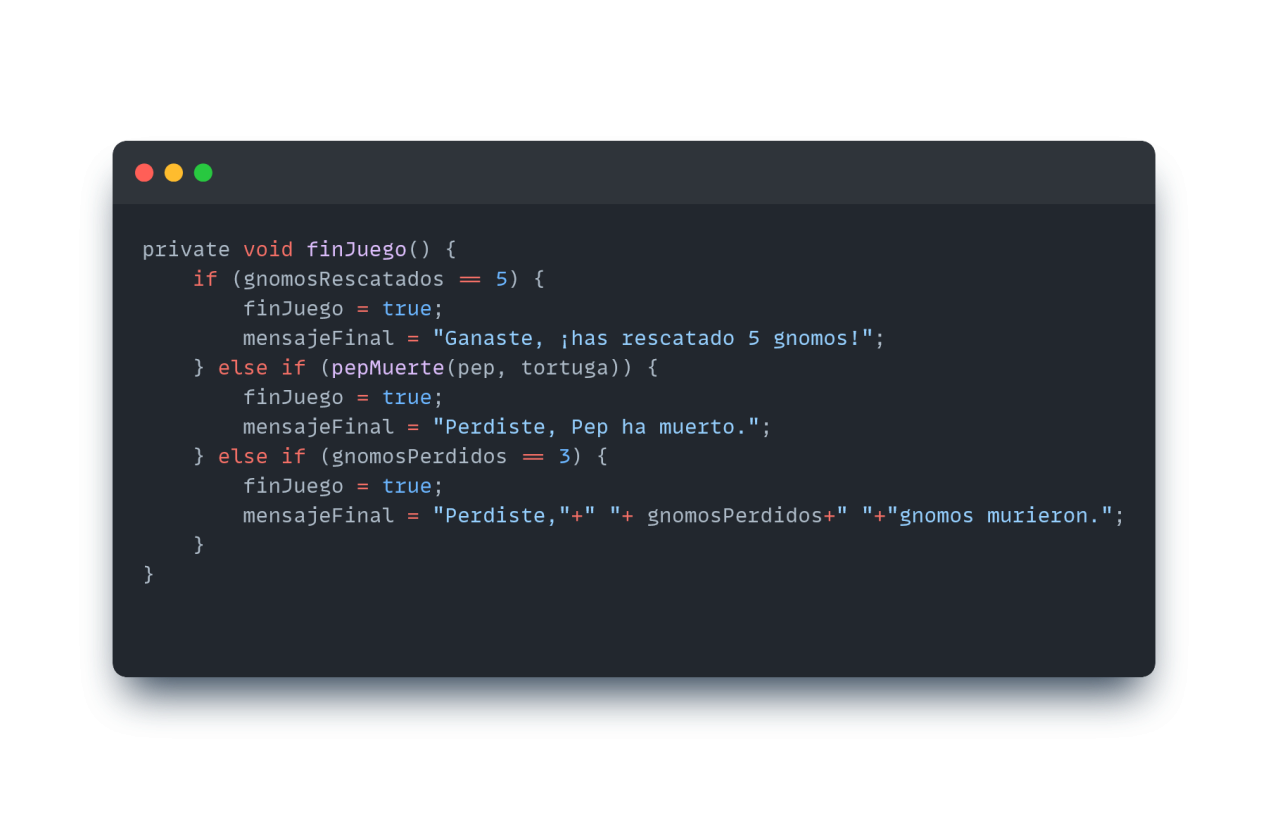
**Métodos principales:**

* **Tick():** Tick se encarga de actualizar el estado del juego en cada ciclo, revisando interacciones entre personajes y elementos. En cada ciclo, se procesan los movimientos de Totoro, el lanzamiento de la bola, y las colisiones.
* **chequearTeclas():**



Controla los movimientos de Totoro y permite al jugador lanzar una bola.

* **finJuego():**



Determina si el juego ha terminado revisando las condiciones de victoria o derrota. Si Totoro muere o se pierden suficientes gnomos, el juego se termina.

* **iniciarJuego():**

Reinicia el estado del juego, reposicionando los elementos en pantalla e inicializando los valores de las variables principales.

* **dibujarEstadisticas():**



Esta función muestra en la pantalla cosas como el tiempo de juego, gnomos rescatados y perdidos, y tortugas eliminadas.

* **pepMuerte():**

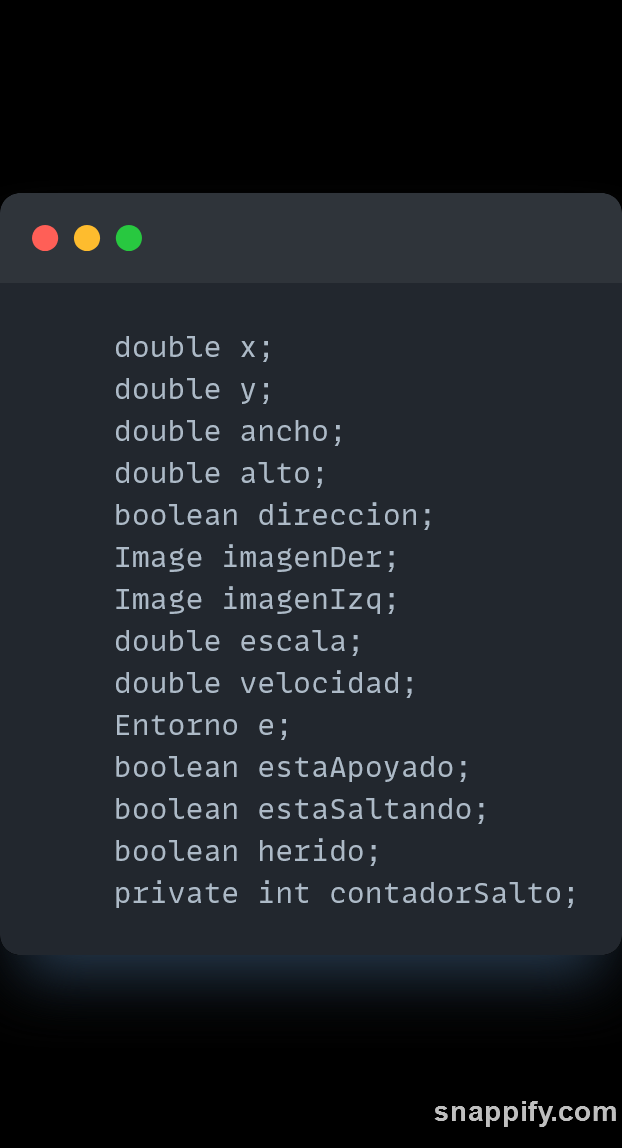
Verifica si Totoro ha colisionado con una tortuga, lo que resultaría en la muerte de Totoro y la finalización del juego.

* **verificarGnomosPisandoIsla():**

Comprueba si los gnomos están pisando una isla, permitiendo modificar su comportamiento y evitando que caigan.

**2.2** **Clase Personaje**

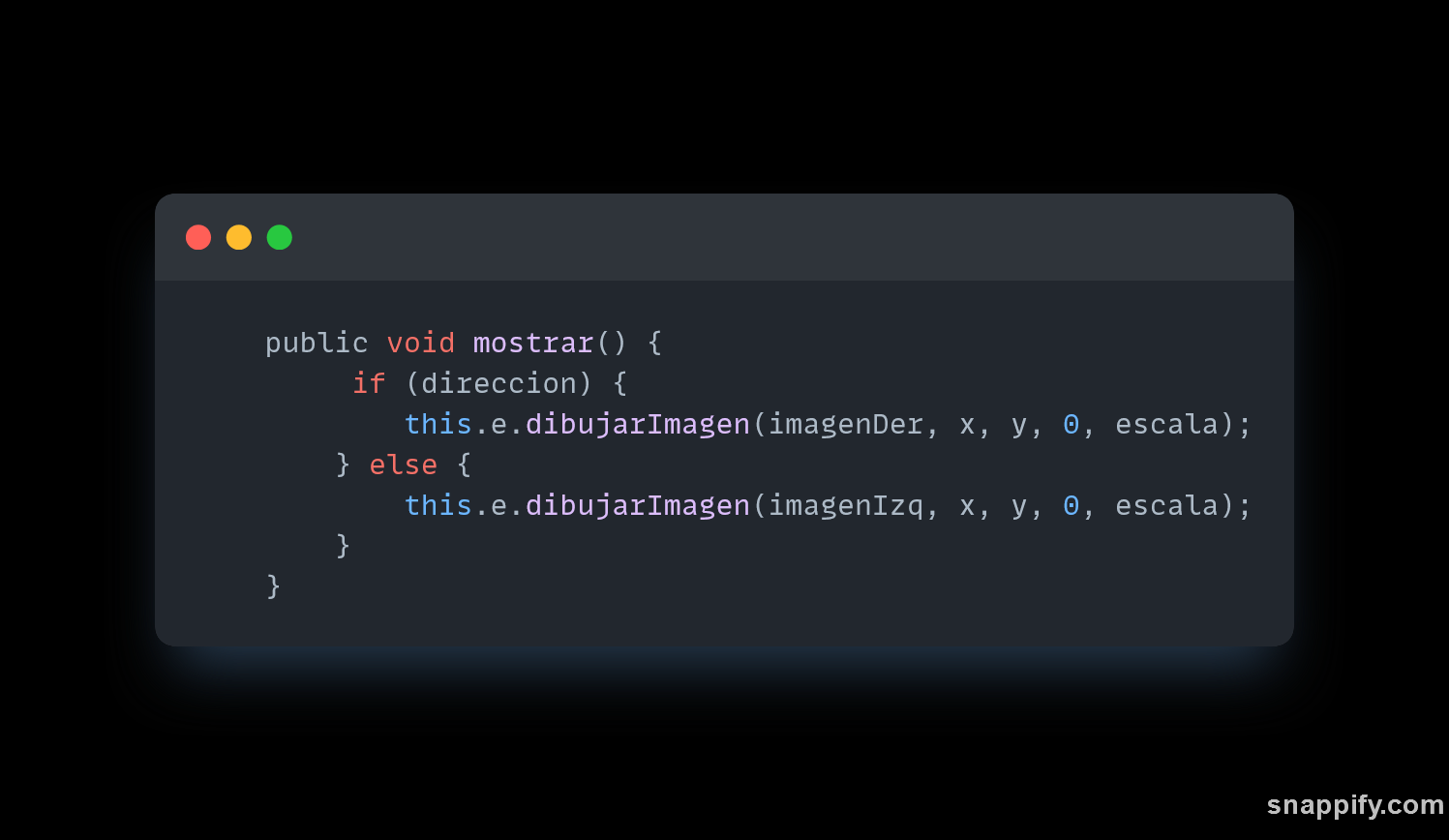
**Variables de instancia:**



La ***clase Personaje*** incluye las variables *x* y *y* para las coordenadas de *Pep*, y *ancho* y *alto* para su tamaño. La variable *direccion* indica si *Pep* mira hacia la derecha o la izquierda, y se usan las imágenes *imagenDer y* *imagenIzq* para representar los diferentes estados visuales del personaje. *escala* y *velocidad* controlan el tamaño y velocidad de movimiento, mientras que *estaApoyado* y *estaSaltando* indican si *Pep* está en el suelo o en el aire. También está la variable *herido*, que señala si *Pep* está herido. Además, *contadorSalto* gestiona la duración de los saltos, y *posicionInicialX* y *posicionInicialY* guardan las posiciones iniciales de *Pep*.

**Métodos principales:**

* **mostrar():**



Se encarga de dibujar a Pep en la pantalla.

* **movVertical():**



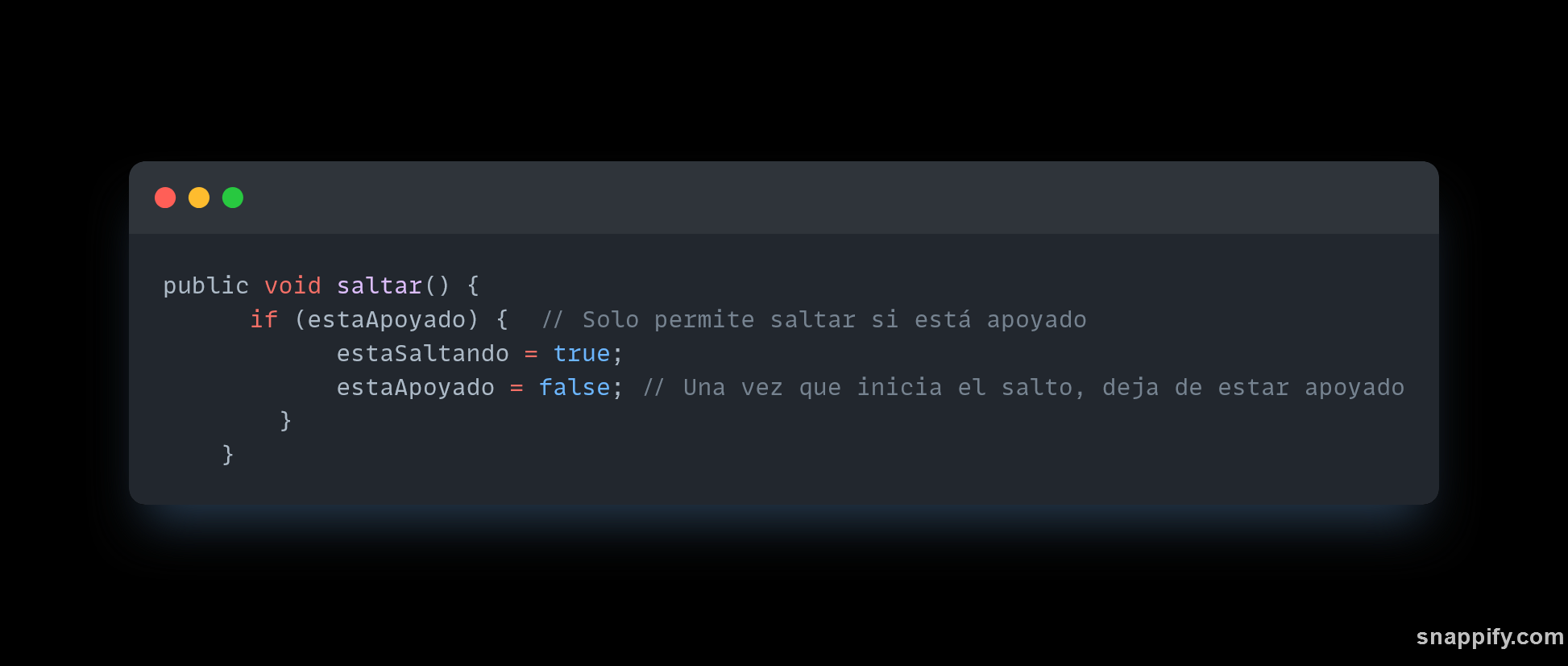
Gestiona el movimiento vertical (gravedad y salto).

* **mover(double v):**



Controla los movimientos de Pep horizontalmente.

* **saltar():**



Inicia el salto si Pep está en una plataforma.

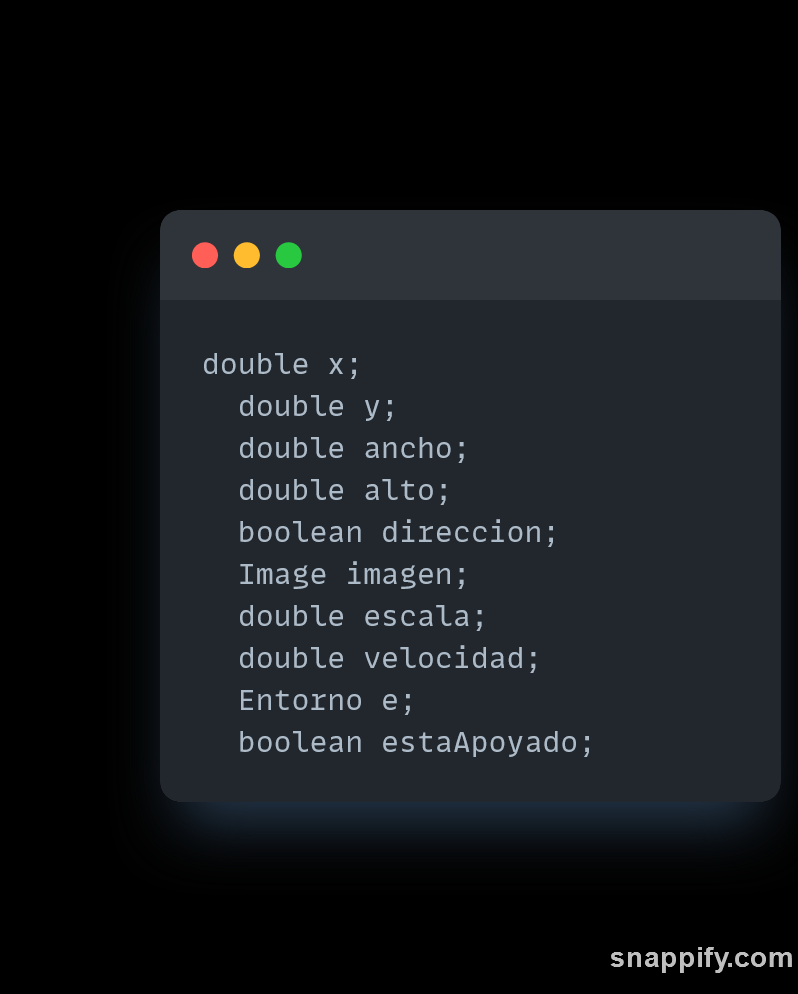
* **cancelarSalto():**



Finaliza el salto cuando Pep toca una superficie.

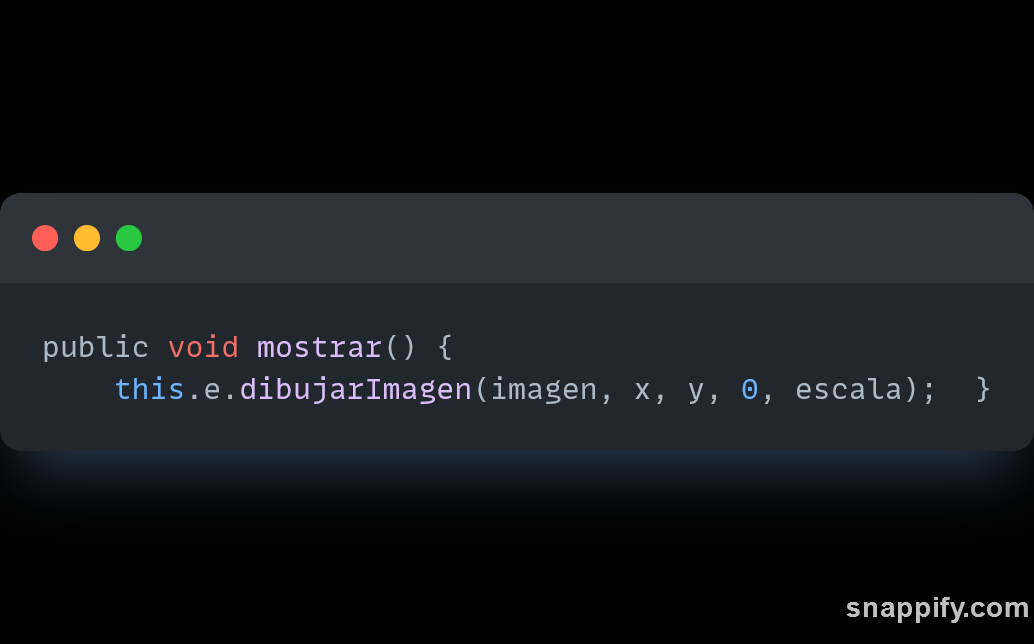
**2.3** **Clase Isla**

**Variables de instancia:**

En la ***clase Isla***, las variables *x* y *y* determinan la posición de la isla, mientras que *ancho* y *alto* definen su tamaño. *imagen* representa la imagen visual de la isla, y *escala* y *velocidad* permiten ajustar el tamaño y la velocidad de movimiento. La variable *estaApoyado* indica si la isla está en contacto con algún personaje.

**Métodos principales:**

* **mostrar():**



Dibuja las islas en la pantalla.

**2.4** **Clase Tortuga**

**Variables de instancia:**



La ***clase Tortuga*** contiene las variables *x* y *y* para la posición de la tortuga, así como *ancho* y *alto* para su tamaño. La dirección de movimiento está determinada por *direccion*, y las imágenes *imagenDer*, *imagenIzq* y *imagenHerida* representan distintos estados de la tortuga. *escala* y *velocidad* ajustan el tamaño y la velocidad, mientras que *herida* señala si la tortuga ha sido dañada.*VELOCIDAD\_CAIDA\_NORMAL* y *VELOCIDAD\_CAIDA\_HERIDA* marca la velocidades las cuales cae.

**Métodos principales:**

* **inicializarPosicion(Isla[] islas):**



Genera una posición inicial aleatoria para un objeto (probablemente un jugador o un elemento del juego) en un entorno de juego con islas. Utiliza un generador de números aleatorios para establecer las coordenadas x y y. Se asegura de que la posición x no se superponga con las primeras tres islas en el eje horizontal, verificando que se mantenga dentro de un margen definido. Si la posición es inválida, se genera una nueva hasta que se encuentre una válida.

* **mostrar():**



Dibuja la tortuga en pantalla en función de su estado (normal o herido) y dirección.

* **actualizar(Isla[] islas):**



Controla el movimiento del objeto en el juego. Si está herido, cae a una velocidad constante. Si no está herido, verifica si tiene apoyo en las islas: si no lo tiene, cae normalmente; si lo tiene, se mueve sobre la isla. También comprueba si el objeto ha salido de la pantalla y lo reinicia si es necesario.

* **herir():**

Marca a la tortuga como herida y provoca que caiga sin apoyo

* **reiniciar(Isla[] islas):**



Reinicia las tortugas, estableciendo una nueva posición para el objeto tras salir de la pantalla. Genera coordenadas aleatorias para x dentro de un rango específico y establece y por encima de la pantalla. Asegura que la nueva posición no colisione con las primeras tres islas. Además, restablece el estado del objeto, marcándolo como no herido, no apoyado, y asigna una dirección aleatoria.

* **moverEnIsla(Isla isla):**



Controla el movimiento de la tortuga sobre una isla, invirtiendo su dirección al llegar a los bordes.

**2.5** **Clase Gnomos**

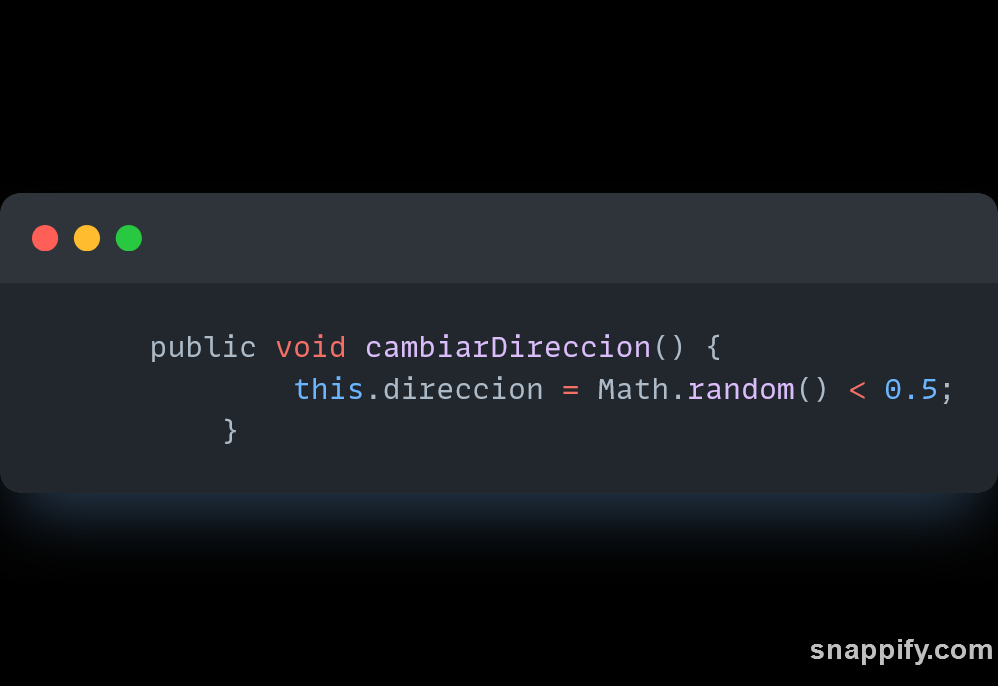
**Variables de instancia:**



La ***clase Gnomos*** cuenta con *x* y *y* para la posición, *direccion* para el sentido del movimiento, y *imagenDer* e *imagenIzq* para las representaciones visuales del gnomo. *tamanio*, *ancho* y *alto* definen el tamaño del gnomo, mientras que *estaApoyado* y *enColision* indican su estado respecto al suelo y colisiones. *saltando* marca si el gnomo está en el aire, y *velocidadSalto* y *velocidad* controlan la rapidez de sus movimientos.

**Métodos principales:**

* **cambiarDireccion():**



Cambia aleatoriamente la dirección en la que se mueve el gnomo.

* **movimientoGnomo():**



Mueve al gnomo en función de su dirección.

* **actualizar():**



Actualiza el estado del gnomo, aplicando el salto y la gravedad si corresponde.

* **reaparecerEnIsla0():**



Reaparecen los gnomos en la isla 0.

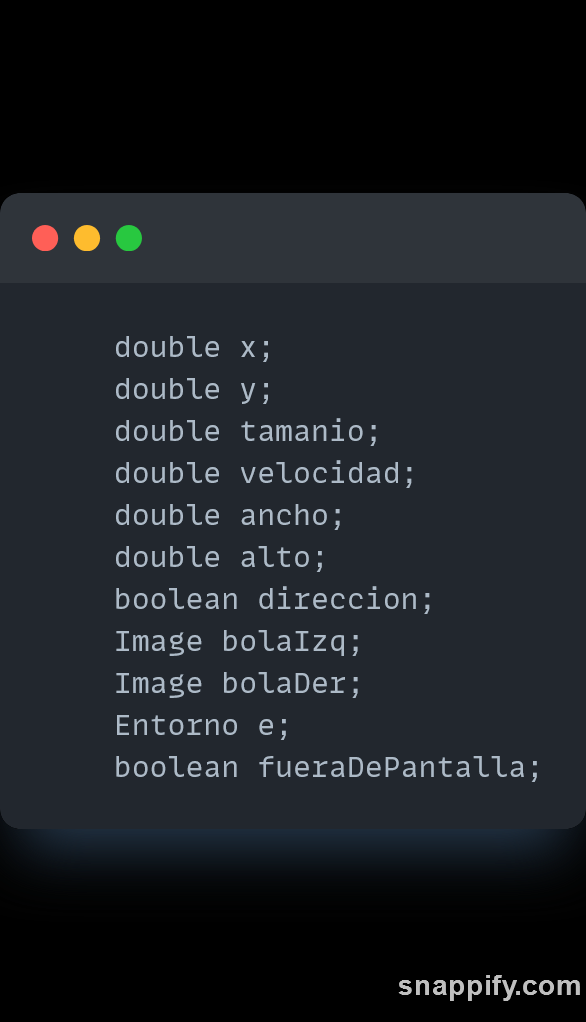
* **dibujarGnomo(Entorno e):**



Dibuja al gnomo en pantalla.

**2.6** **Clase Poder**

**Variables de instancia:**



En ***Poder***, las variables *x* y *y* indican la posición del proyectil, mientras que *tamanio* y *velocidad* ajustan su tamaño y velocidad de desplazamiento. *direccion* controla la dirección en la que se mueve la bola, y las imágenes *bolaIzq* y *bolaDer* representan el proyectil desde distintos ángulos. Finalmente, *fueraDePantalla* indica si el proyectil ha salido del campo de visión.

**Métodos principales:**

* **dibujarBola(Entorno e):**

Dibuja el proyectil en pantalla en función de su dirección.

* **lanzarBola():**

Mueve el proyectil en la dirección establecida.

**2.7** **Clase Fondo**

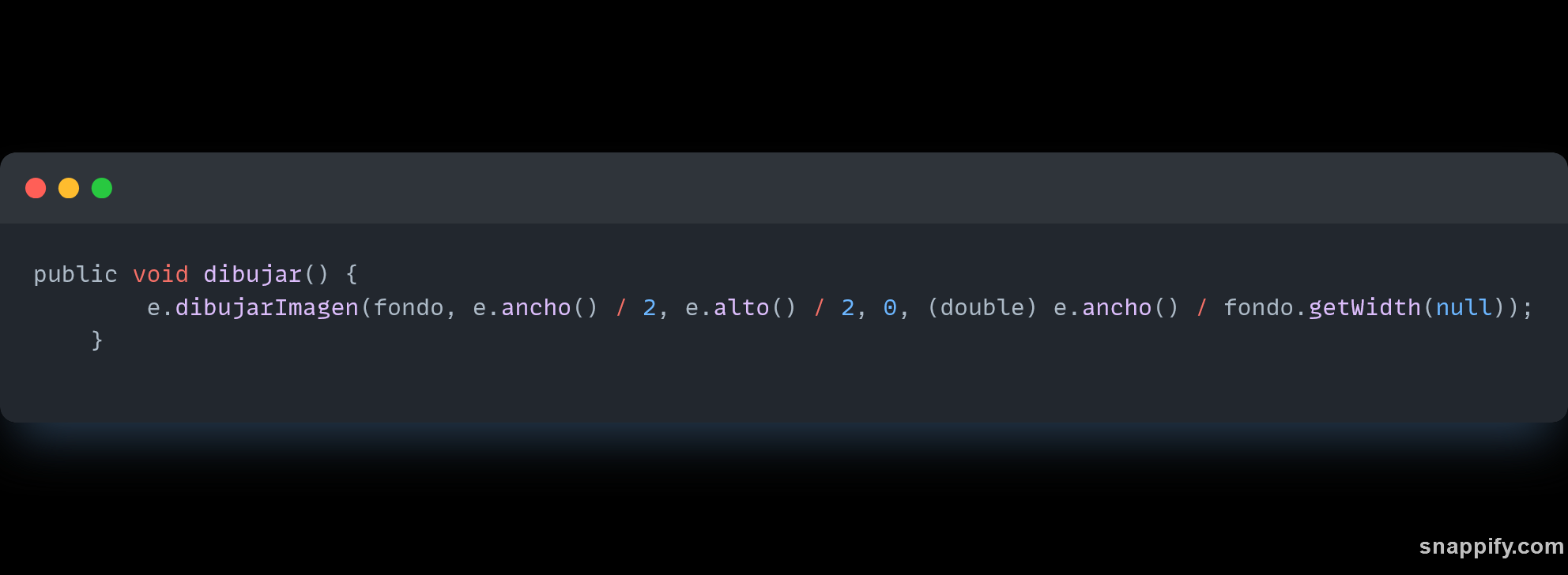
**Variables de instancia:**



En ***Fondo***, las variables de instancia incluyen *fondo*, que contiene la imagen de fondo, y *e*, que representa el entorno gráfico donde se dibuja.

**Métodos principales:**

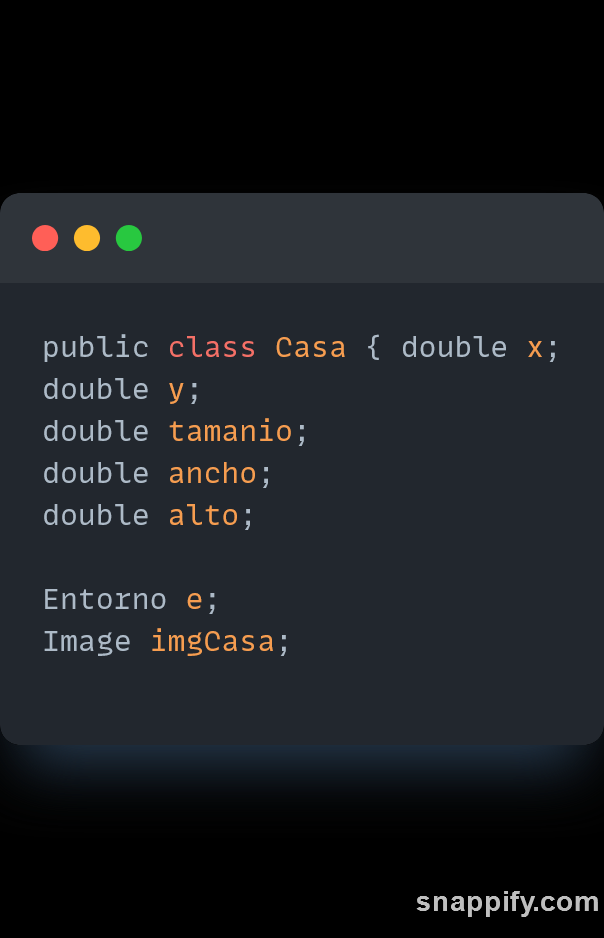
* **dibujar():**



Dibuja la imagen de fondo centrada en la pantalla, ajustándola al ancho del entorno.

**2.8** **Clase Casa**

**Variables de instancia:**



La ***clase Casa*** utiliza las variables *x* y *y* para posicionar la casa en el entorno, *tamanio* para definir su escala, *imgCasa* como imagen visual, y *ancho* y *alto* para establecer sus dimensiones.

**Métodos principales:**

* **dibujarCasa(Entorno e):**



Dibuja la casa en pantalla en la posición especificada.

## 3.0 **Problemas y Soluciones**

1. **Problema de posicionamiento de islas**: Hubo dificultad en la distribución inicial de las islas en múltiples filas.

**Solución**: Se utilizó un cálculo geométrico en iniciarJuego() para posicionarlas de forma escalonada.

1. **Colisiones y apoyo de las tortugas en las islas**: Uno de los problemas más complicados fue hacer que las tortugas colisionaran correctamente con las islas. En ocasiones, las tortugas se quedaban atrapadas en las esquinas o intentaban apoyarse en islas incorrectas, como cuando caían entre dos islas y no determinaban cuál era la correcta. Esto causaba que las tortugas "saltaran" entre islas o se bugearan al cambiar de posición.

* **Solución**: Para resolver este problema, se mejoró la lógica de detección de colisiones y apoyo en la clase ***Tortuga***. En el método *verificarApoyo(Isla[] islas)*, se ajustaron las condiciones para que las tortugas solo se apoyaran cuando estaban claramente dentro de los límites de una isla. Además, en el método *estaEnIsla(Isla isla)*, se añadieron márgenes para que las tortugas no intenten apoyarse en islas que solo están parcialmente debajo de ellas. Estos ajustes aseguraron que las tortugas se apoyaran de manera más precisa y no se quedaran atrapadas en las esquinas.

1. **Colisionaba el personaje con la parte inferior de las islas**: El personaje atravesaba la isla

**Solución**: Se implementó el método para que tenga en cuenta la parte inferior de las isla y así lograr que el personaje deje de atravesarla.

4.0 **CONCLUSIÓN**

Este proyecto permitió aplicar conceptos clave de la programación orientada a lo visto en clases. Aprendimos la importancia de planificar la estructura del código y la gestión de colisiones, lo cual fue un desafío en ciertos puntos del desarrollo. En general, el resultado es satisfactorio, y el juego refleja un buen nivel de complejidad y funcionalidad, aplicando los conceptos aprendidos.