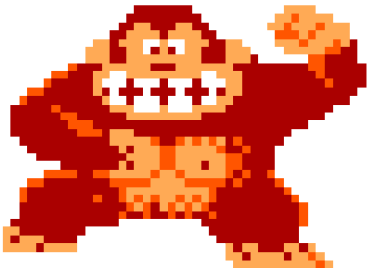
Clase: Donkey



Alto Original:	350px
Ancho Original:	500px
Escala utilizada:	0.19
Alto Utilizado:	66px
Ancho Utilizado:	95px
Distancia centro-suelo:	33px

La clase Donkey permite generar un objeto que se encargará, de arrojar barriles según una elección aleatoria del tiempo. También, desde el apartado gráfico, genera una animación del antagonista del juego, que varía entre estar enojado de forma constante, y la de simular que arroja barriles en la creación de uno de estos.

VARIABLES

- **`int ultimoLanzamiento;`**
  Indica el momento (número de tick desde que comenzó la ejecución) en que se realizó el último lanzamiento de un barril.

- **`int lanzarRandom;`**
  Indica el momento (número de tick desde que comenzó la ejecución) en que se realizará el próximo lanzamiento de un barril.

- **`String violencia;`**
  Sirve para indicar el estado de donkey con respecto a si debe arrojar barriles o no.

MÉTODOS *

- **Constructor** (No requiere parámetros adicionales)

  El constructor solamente asigna como último lanzamiento el momento cero. Y el estado de violencia como “violento”.

- **Gorilear** (Debe recibir el entorno como parámetro y el momento actual expresado como un número entero, donde cero es el inicio de ejecución)

  Este método genera una constante animación. Es solamente un decorado.

  Si el último lanzamiento se realizó hace menos de 30 ticks debe mostrar la animación “tirar”. De lo contrario debe mostrar la simple animación llamada “gorilear”.

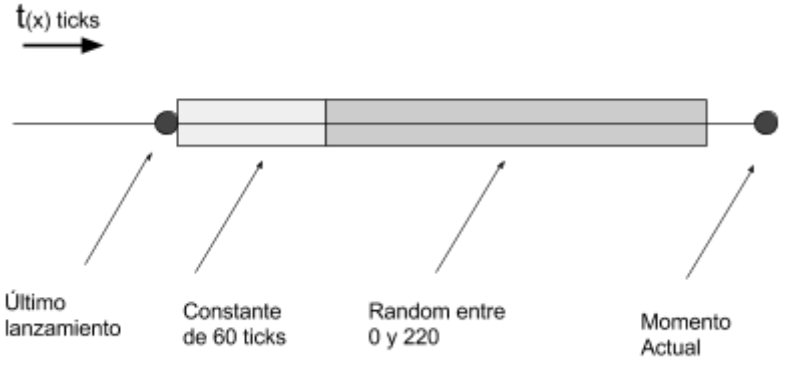
- **decidir**(Debe recibir el momento actual expresado como un número entero, donde cero es el inicio de ejecución)

  Crea y utiliza una variable local: `Random rnd = new Random();`

  Es el método que se encarga de ejecutar un algoritmo que decide de forma aleatoria en cual tick del futuro (momento del juego) se lanzará el siguiente barril. También es el método que indica que debe ser lanzado el barril en ese preciso instante, si el momento actual es igual al tick que fue planeado su lanzamiento. Lo anterior ocurre si la **`violencia`** está seteada en "violento".

  Retorna **`true`** si el momento actual es igual a **`lanzarRandom`**. Retorna **`false`** para los demás casos.
  Una vez lanzado el barril, **`lanzarRandom`** pasa a tener valor 0 (no hay lanzamiento a posterior planificado).

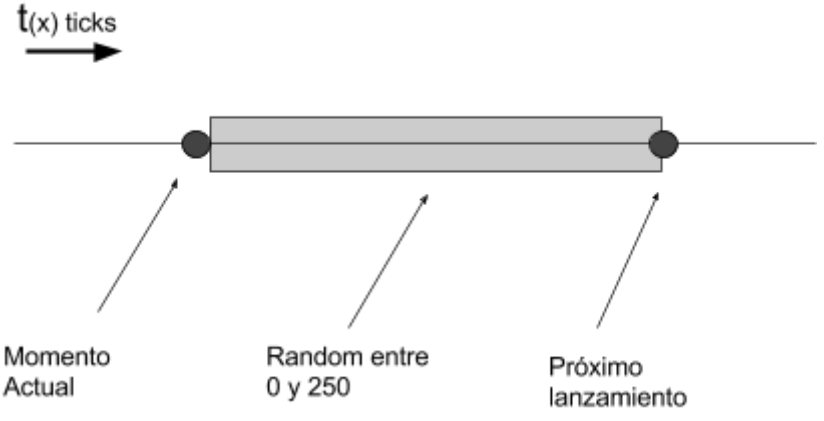
  Existe un algoritmo basado en dos randoms diferentes para decidir el próximo lanzamiento.



  Se decidirá una planificación de un lanzamiento futuro, si ocurre primero que el momento actual es mayor o igual al último lanzamiento más 60 ticks y un random de entre 0 y 220 ticks.

  Esto garantiza que no se dé un paso inicial hacia lanzar barriles cada menos de 60 ticks (aprox 2 segundos o menos).

  Cuando se cumple lo anterior. Se establece un random entre 0 y 250 hacia el futuro para planificar el próximo lanzamiento.



- **noMasViolencia**(No requiere parámetros adicionales)

  Este método cambia la variable **`violencia`** a “noviolento”. Lo que impide al método “decidir“ de arrojar barriles o planificar futuros lanzamientos. Esta función es llamada desde la clase principal cuando el juego termina, ya sea al perder o al ganar.

- **arribaOabajo**(No requiere parámetros adicionales)

  Crea y utiliza una variable local: Random rnd = new Random();

  Este método genera un random entre 0 y 60. Si el número elegido aleatoriamente es múltiplo de 3, entonces se decide arrojar el barril directamente hacia la viga inferior, de lo contrario se arroja por la misma viga donde está donkey.

  Este método se diseñó para agregarle dificultad al juego.

  Se utiliza un random y que el resultado sea múltiplo de 3 para que en general se respete que haya un aproximado de 33% de posibilidades de arrojar el barril por debajo y un 66% por la viga normal.

  Se retorna -3 o -1 con la intención de que el resultado de esta función le reste dichas unidades al valor length del arreglo de Vigas[].

ANOTACIONES

Para la correcta aparición gráfica de Donkey se recomienda colocar ciertos valores de **entorno.dibujarImagen**

```
viga = Viga (pos = 6)
y = distancia ( viga.y - viga.alto / 2)
```

La distancia entre el "y" de la viga (con pos = 6) menos la mitad de su alto

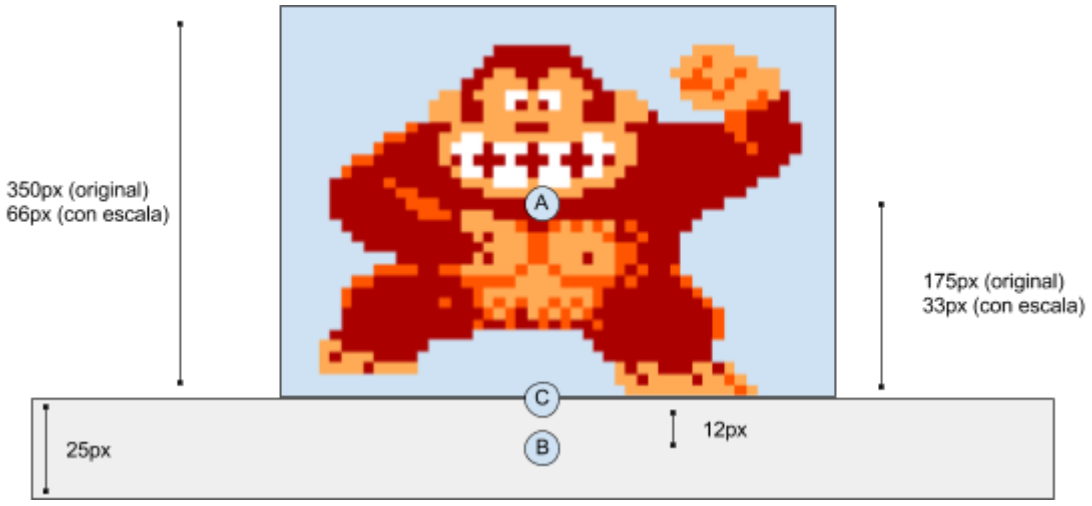
Para una viga en posición 6 con los siguientes valores:

```
x = 325;
y = 75;
largo = 700;
alto = 25;
```

Se recomienda ubicar a Donkey en:

```
x = 50
y = 30
escala = 0.19
```

Esto produce que el último píxel inferior del Donkey pise (o límite inmediata y superiormente con) el último píxel superior de la viga).



A = Punto centro de la imagen (x,y) Donkey.
B = Punto centro del rectángulo que dibuja la Viga.
C = Límite entre la viga y la imagen.