



# Game Builder Diagram Class Description



# INDICE

---

Paquete: kernel.gui.....	3
Clase: Renderizable.....	4
Clase: StaticEntity.....	6
Clase: Animation.....	7
Clase: Layer.....	8
Clase: Manager.....	9
Clase: GuiAmbassador.....	10
Paquete: kernel.logic.....	11
Clase: LogicAmbassador.....	12

# Paquete: kernel.gui

**Dependencia:** **kernel.logic**

Este paquete tiene la finalidad de:

- pintar objeto bidimensionales.
- Interactuar con estos objetos mediante entrada de teclado.
- Manejar los objetos pintados, ya sea eliminarlos, pintarlos, agregarlos, actualizarlos .
- Crear una abstracción tan alta que no requiera de una parte lógica por completo sino solamente una dependencia con una única clase embajadora de un paquete lógico, esta clase es llamada : **LogicAmbassador**.

## Clase: Renderizable

**Descripción:** Renderizable es una clase abstracta que actúa según el patrón de diseño fabrica facilita el uso de los objetos renderizables por separado mediante la abstracción, dicho de otra forma es el punto de abstracción mas alto del conjunto de clases que se pintan en un lienzo llamado “g”.

**Hereda de:** Ninguno.

**Dependencias:** Ninguna.

### ATRIBUTOS

---

- **dimension (private):Rectangle** = Es el conjunto de dimensiones que toma el objeto virtual para “ubicarse espacialmente”, básicamente es el rectángulo R con las dimensiones x, y, width, height que ubican al objeto renderizable en el lugar que se quiere pintar.
- **g (protected):Graphics** = Es el “lienzo” en el que se pinta.
- **id (private):String** = Es el identificador único del objeto, esto para ubicarlo mediante listas y demás.
- **layer (private):Layer** = Es un puntero a la capa en la que se encuentra.

### METODOS

---

- **render():void** = Método **no concreto**, osea virtual, corresponde a la forma en que se pinta cada una de las implementaciones de las clases pertenecientes a la jerarquía de clases que compartan este método. Desde pintar una imagen o una serie de imágenes hasta pintar diferentes animaciones una tras de otra.
- **update():void** = Método **no concreto**, osea virtual, que corresponde a la forma en que un objeto actualiza su estado, posición, frame de animación, imagen de secuencia o cualquier cosa relacionado a las físicas del juego tales como colisiones o saltos.
- **colision(renderizable:Renderizable):bool** = Retorna si un objeto renderizable colisiona con otro del mismo.
- **getRectangle():Rectangle** = Retorna sus coordenadas y dimensiones empaquetadas en una clase llamada Rectangle.
- **getX():int** = Retorna su posición en el eje x mas la posicion de la capa en la que

se encuentra, ejemplo: **return dimension.x + layer.x.**

- **getY():int** = Retorna su posición en el eje y.
- **getHeight():int** = Retorna su altura.
- **getWidth():int** = Retorna su anchura.
- **getId():String** = Retorna su ID único.
- **setX(x:int):void** = Reescribe el valor perteneciente a su posición en el eje x.
- **setY(y:int):void** = Reescribe el valor perteneciente a su posición en el eje y.
- **getLayer():void** = Retorna el puntero de la capa en la que se encuentra.
- **setLayer(layer:Layer):void** = Reescribe el puntero de la capa en la que se encuentra.

## Clase: StaticEntity

**Descripción:** StaticEntity es una clase que contiene una sola imagen existente, esta debe ser diferente de null, Esta imagen se mueve según el movimiento del jugador. Se mueve en la dirección contraria al jugador.

**Hereda de:** **Renderizable.**

**Dependencias:** BufferedImage.

### ATRIBUTOS

---

- **sprite (private):BufferedImage** = es la imagen a pintar en el “lienzo g”.

### METODOS

---

- **render():void** = pinta la imagen según la posición entregada por los métodos getX() y getY().
- **update():void** = Actualiza la posición de la imagen.
- **StaticEntity(x:int, y:int, width:int, height:int, sprite:BufferedImage, layer:Layer)** = Es el método constructor los cuatro primeros valores pueden tomar valores cuales quiera mientras que sprite es diferente de un puntero nulo al igual que layer.

## Clase: Animation

**Descripción:** Animation es una clase que contiene una lista circular de punteros a imágenes existentes siguiendo el patrón de diseño “peso mosca”. Esta serie de imágenes se mueve según el movimiento del jugador. Se mueve en la dirección contraria al jugador.

**Hereda de:** **Renderizable.**

**Dependencias:** **BufferedImage, CircularList<E>, CircularIterator.**

### ATRIBUTOS

---

- **spriteCollection (private):CircularList** = es la lista de imágenes a pintar en el “lienzo g”.
- **spriteIterator (private):CircularIterator** = es el iterador de la lista de sprites.

### METODOS

---

- **render():void** = pinta la imagen que esta apuntada por el iterador, y pinta según la posición entregada por los métodos getX() y getY().
- **update():void** = La posición de la imagen y la imagen a usar o sea la siguiente a usar.
- **Animation(x:int, y:int, width:int, height:int, layer:Layer)** = Es el método constructor los cuatro primeros valores pueden tomar valores cuales quiera mientras que layer es diferente de un puntero nulo.
- **addFinalFrame(sprite:BufferedImage):void** = Agrega un frame al final de la lista de la animación y genera un nuevo iterador.
- **GenerateIterador():void** = genera un iterador de la lista.

## Clase: Layer

**Descripción:** Layer es una clase que contiene una doble lista objetos renderizables, sirve para empaquetar por capas la renderizacion, esto con respecto a una clase superior **Manager**. Se mueve en la dirección contraria al jugador.

**Hereda de:** **Renderizable**.

**Dependencias:** **Renderizable**, **DoubleList<E>**, **DoubleIterator<E>**.

### ATRIBUTOS

---

- **group:DoubleList** = Es el grupo de objetos que contiene la capa.
- **groupIterator:DoubleIterator** = Es el itarador de la lista.

### METODOS

---

- **render():void** = pinta los objetos Renderizables que contiene.
- **update():void** = posiciona la capa según el movimiento del personaje.
- **Layer()** = Genera una lista, posiciona la capa en el punto (0,0).
- **getGroup():DoubleList** = retorna el atributo group.
- **updateIterador():void** = genera un iterador de la lista.



## Clase: Manager

**Descripción:**

**Hereda de:** Ninguno.

**Dependencias:** Renderizable, DoubleList<E>.

### ATRIBUTOS

---

- **layerCollection (private):CircularList** = es la lista de imágenes a pintar en el “lienzo g”.
- **spriteIterator (private):CircularIterator** = es el iterador de la lista de sprites.

### METODOS

---

- **render():void** = pinta la imagen que esta apuntada por el iterador, y pinta según la posición entregada por los métodos getX() y getY().
- **update():void** = La posición de la imagen y la imagen a usar o sea la siguiente a usar.
- **run():void** =
- **getLayers():DoubleList** =

## Clase: GuiAmbassador

**Descripción:**

**Hereda de:** Ninguno.

**Dependencias:** BufferedImage, Renderizable.

### ATRIBUTOS

---

- **instance:GuiAmbassador =**

### METODOS

---

- **getInstance():GuiAmbassador =**
- **addLayer(index:int):void =**
- **removeLayer(index:int):boolean =**
- **addRendezable(i:int,j:renderizable:Renderizable): boolean =**
- **createStatic()**
- **createAnimation() =**
- **createEntity()=**
- **createInteractor()=**

# Paquete: kernel.logic

**Dependencia:** `kernel.gui`

Este paquete tiene la finalidad de:

- pintar objeto bidimensionales.
- Interactuar con estos objetos mediante entrada de teclado.
- Manejar los objetos pintados, ya sea eliminarlos, pintarlos, agregarlos, actualizarlos .
- Crear una abstracción tan alta que no requiera de una parte lógica por completo sino solamente una dependencia con una unica clase embajadora de un paquete logico.

**Clase: LogicAmbassador**