Game Builder Diagram Class Description

INDICE

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 vitual 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 método 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>, CicularIterator.

ATRIBUTOS

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

METODOS

- **render():void** = pinta la imagen que esta apuntada por el iterador, y pinta según la posición entregada por los método 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(psprite: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: Es el “pintor original” pues de esta clase es que cada clase pinta por si misma, esta clase utiliza el patrón de diseño **cadena de responsabilidades** a través de los métodos “render():void” y “update():void”, utilizando la abstracción y mediante la recursividad logra enfocar este patrón de diseño.

Hereda de: Ninguno.
Dependencias: Renderizable, Layer, DoubleList<E>.

ATRIBUTOS

- **actor:Actor** = Es el actor del juego por el momento solo soporta un jugador.
- **LayerCollection:DoubleList** = Es la lista de capas que contiene esta clase.
- **zflag:bool** = Es la bandera del botón z.
- **xflag:bool** = Es la bandera del botón x.
- **upflag:bool** = Es la bandera del botón arriba.
- **downflag:bool** = Es la bandera del botón abajo.
- **leftflag:bool** = Es la bandera del botón izquierda.
- **rightflag:bool** = Es la bandera del botón derecha.
- **pauseflag:bool** = Es la bandera del botón pausa.
- **selectflag:bool** = Es la bandera del botón selección.
- **lflag:bool** = Es la bandera del botón l.
- **rflag:bool** = Es la bandera del botón r.

METODOS

- **render():void** = Pinta la lista de capas que contiene esta clase.
- **update():void** = Se encarga de analizar las físicas por separado de cada clase, utilizando la recursividad.
- **run():void** = Es el núcleo del juego es este quien se encarga de realizar el llamado “gameloop”.
- **getLayers():DoubleList** = Retorna la lista de capas.

Clase: GuiAmbassador

Descripción: Es la clase embajadora del paquete, se encarga de comunicar el paquete **kernel.gui** con el resto del programa, Además de actuar como una fabrica de objetos renderizables y sigue el patrón de diseño Singleton.

Hereda de: Ninguno.
Dependencias: Renderizable, Manager, BufferedImage.

ATRIBUTOS

- **instance:GuiAmbassador** = Es la única instancia de la clase GuiAmbassador, cumpliendo así el patrón de diseño simgleton, además de ser un objeto estático o sea dura todo el tiempo de ejecución del programa.

METODOS

- **getInstance():GuiAmbassador** = Retorna la única instancia existente de la clase GuiAmbassador.
- **addLayer(index:int):void** = Crea una nueva capa en la clase Manager en el índice indicado por el parámetro.
- **removeLayer(index:int):boolean** = Remueva la capa en indicada por el índice que es ingresado en el método, retorna true si la capa a sido borrada.
- **addRendezable(i:int,j:renderizable:Renderizable): boolean** = Agrega un objeto renderizable en el punto i,j de la matriz que contiene la clase Manager, i corresponde al numero de capa, mientras que j representa el índice de la capa:Layer.
- **createStatic()** = Crea y retorna un objeto renderizable estático de la clase **StaticEntity**.
- **createAnimation()** = Crea y retorna un objeto del tipo **Animation**.
- **createEntity()**= Crea una entidad de los tipos actor o enemy, aun se desconoce como sirve el método.
- **createInteractor()**= Se encarga de crear eventos del tipo Interactor.

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 única clase embajadora de un paquete lógico.

Clase: LogicAmbassador