Arkanoid

Roland Coeurjoly Lechuga, 100276197
Daniel de la Flor Aceituno, 100277140

10/01/2012

Tabla de contenidos

Table of contents

Índice de clases

Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:

Índice de clases

Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Documentación de las clases

Referencia de la Clase ladrillo

Diagrama de herencias de ladrillo

Métodos públicos

1 **ladrillo** (int **x**, int y, int ancho, int altura)

Crea el ladrillo a partir de los parametros que le pasas.

2 void **dibujar** () const

Dibuja el ladrillo con los vertices hallados a partir de los parámetros del ladrillo.

Documentación del constructor y destructor

ladrillo::ladrillo (int*x*, int*y*, int*ancho*, int*altura*)

Crea el ladrillo a partir de los parametros que le pasas.

Parámetros:

x	Posicion horizontal del ladrillo
y	Posicion vertical del ladrillo
ancho	Ancho del ladrillo
largo	Largo del ladrillo

Documentación de las funciones miembro

void ladrillo::dibujar () const `[virtual]`

Dibuja el ladrillo con los vertices hallados a partir de los parámetros del ladrillo.

Nota:

el color depende de la coordenada horizontal (x)

Implementa **obj_dib** (*p.7*).

La documentación para esta clase fue generada a partir de los siguientes ficheros:

3 ladrillo.h
4 ladrillo.cpp

Referencia de la Clase misil

Diagrama de herencias de misil

Métodos públicos

5 **misil** (int **x**, int y, double vy)

Crea el misil a partir de los parametros que le pasas.

6 void **dibujar** () const

Dibuja el misil con los vertices hallados a partir de los parámetros del ladrillo.

7 void **mover** (**obj_dib** *objetochocado, int &bloques_restantes, int &vidas)

mueve el misil

8 double **getVy** ()

consigues la velocidad del misil

9 virtual void **setVy** (double vy)

modificas la velocidad del misil

Documentación del constructor y destructor

misil::misil (int*x*, int*y*, double*vy*)

Crea el misil a partir de los parametros que le pasas.

Nota:

no necesitamos el ancho y el largo porque siempre es el mismo por defecto. Solo tiene velocidad vertical el misil

Parámetros:

x	Posicion horizontal del misil
y	Posicion vertical del misil
vy	velocidad vertical del misil

Documentación de las funciones miembro

void misil::dibujar () const [virtual]

Dibuja el misil con los vertices hallados a partir de los parámetros del ladrillo.

Nota:

el color es siempre azul (0,0,1) segun el rgb

Implementa **obj_dib** (*p.7*).

double misil::getVy () [inline, virtual]

consigues la velocidad del misil

Devuelve:

vy para comprobar que tiene velocidad

Reimplementado de **obj_dib** (*p.10*).

void misil::mover (obj_dib ***objetochocado*, int &*bloques_restantes*, int &*vidas*) [virtual]

mueve el misil

Nota:

es similar al mover de la pelota solo que solo incluye el choque inferior

Parámetros:

objetochocado	comprueba si se choca con ese objeto
bloques_restantes	Resta un bloque si choca contra él
vidas	nunca resta una vida. esta pensado para cuando la pelota choca contra la pared inferior

Reimplementado de **obj_dib** (*p.10*).

void misil::setVy (double*vy*) `[virtual]`

modificas la velocidad del misil

Nota:

se utiliza para parar el misil cuando choca contra algo

Reimplementado de **obj_dib** (*p.11*).

La documentación para esta clase fue generada a partir de los siguientes ficheros:

10 misil.h
11 misil.cpp

Referencia de la Clase nave

Diagrama de herencias de nave

Métodos públicos

12 **nave** (float **x**, float y)
creas la nave

13 virtual **~nave** ()
destructor de la nave

14 void **pressedKey** (unsigned char c)
mueves la nave cuando pulsas (a) o (d)

15 void **dibujar** () const
dibujas la nave en funcion de sus parametros

Documentación del constructor y destructor

nave::nave (float*x*, float*y*)

creas la nave

Nota:

el ancho el siempre el mismo y el largo solo se modifica con el bonus de alargar nave

virtual nave::~nave () [inline, virtual]

destructor de la nave

Nota:

no lo utilizamos en ningun momento, porque no es posible conocer la posicion en el mundo en la que esta tal y como hemos hecho el codigo

---

Documentación de las funciones miembro

void nave::pressedKey (unsigned char*c*) [virtual]

mueves la nave cuando pulsas (a) o (d)

Nota:

se mueve siempre entre unos limites

Reimplementado de **obj_dib** (*p.11*).

La documentación para esta clase fue generada a partir de los siguientes ficheros:

16 nave.h
17 nave.cpp

Referencia de la Clase obj_dib

Diagrama de herencias de obj_dib

Métodos públicos

18 **obj_dib** (string nombre)

inicias el objeto con el nombre asignado

19 virtual void **dibujar** () const =0

funcion virtual pura. cada objeto se dibuja de su forma

20 void **getBoundaries** (float &x_izq, float &y_ab, float &x_der, float &y_arr) const

calculas los limites izquierdo, derecho, superior, e inferior.

21 virtual void **mover** (**obj_dib** *objetochocado, int &bloques_restantes, int &vidas)

funcion virtual de mover. se implementa para misil y pelota. el resto de casos no hace nada

22 virtual void **pressedKey** (unsigned char c)

funcion virtual de presionar botones. se implementa para nave. el resto de casos no hace nada

23 void **setX** (float **x**)

Funciones set.

24 void **setY** (float y)

funcion set de la y. se utiliza para cambiarla

25 void **setWidth** (float w)

funcion set de la x_size. se utiliza para cambiarla

26 void **setHeight** (float h)

funcion set de la y_size. se utiliza para cambiarla

27 virtual void **setVy** (double vy)

sirve para cambiar la velocidad de la pelota o el misil fuera de sus clases

28 float **getX** ()

sacas el valor de la x

29 float **getY** ()

sacas el valor de la y

30 float **getX_SIZE** ()

sacas el valor de la x_size (la mitad del ancho)

31 float **getY_SIZE** ()

sacas el valor de la y_size (la mitad de la altura)

32 virtual double **getVx** ()

sacas el valor de la velocidad en x. se utiliza para la pelota

33 virtual double **getVy** ()

sacas el valor de la velocidad en . se utiliza para la pelota

34 bool **choque** (**obj_dib** *obj, int &c)

ves si hay choque y devuelves el tipo de choque que se produce (inf, sup, izq, derecho) sobrecarga de funcion, porque tienen distintos parametros que les pasas

35 bool **choque** (**obj_dib** *obj)

ves si hay choque, llamas dentro a la anterior funcion y devuelves el tipo de choque que se produce (inf, sup, izq, derecho)

36 string **getName** ()

consigues el nombre del objeto dibujable. muy util para comprobar que chocas con lo quieres. o que estas trabajando con el objeto requerido

Atributos protegidos

37 string **name**

nombre del objeto

38 float **x**

coordenadas

39 float **y**

40 float **x_size**

mitad del ancho y mitad de la altura

41 float **y_size**

Documentación del constructor y destructor

obj_dib::obj_dib (string*nombre*) `[inline]`

inicias el objeto con el nombre asignado

Parámetros:

nombre	el nombre en cuestion

Documentación de las funciones miembro

bool obj_dib::choque (obj_dib *obj, int &c)

ves si hay choque y devuelves el tipo de choque que se produce (inf, sup, izq, derecho) sobrecarga de funcion, porque tienen distintos parametros que les pasas

Devuelve:

0 si no hay choque, 1 si lo hay

bool obj_dib::choque (obj_dib *obj)

ves si hay choque, llamas dentro a la anterior funcion y devuelves el tipo de choque que se produce (inf, sup, izq, derecho)

Devuelve:

0 si no hay choque, 1 si lo hay

void obj_dib::getBoundaries (float &x_izq, float &y_ab, float &x_der, float &y_arr) const

calculas los limites izquierdo, derecho, superior, e inferior.

Parámetros:

x_izq	limite izquierdo por referencia
y_ab	limite inferior por referencia
x_der	limite derecho por referencia
y	_arr limite superior por referencia

string obj_dib::getName ()

consigues el nombre del objeto dibujable. muy util para comprobar que chocas con lo quieres. o que estas trabajando con el objeto requerido

Devuelve:

name

virtual double obj_dib::getVx () [inline, virtual]

sacas el valor de la velocidad en x. se utiliza para la pelota

Devuelve:
la vx
Reimplementado en **pelota** (*p.14*).

virtual double obj_dib::getVy () [inline, virtual]

sacas el valor de la velocidad en . se utiliza para la pelota

Devuelve:
la vy
Reimplementado en **misil** (*p.5*) y **pelota** (*p.15*).

float obj_dib::getX () [inline]

sacas el valor de la x

Devuelve:
la x

float obj_dib::getX_SIZE () [inline]

sacas el valor de la x_size (la mitad del ancho)

Devuelve:
la x_size

float obj_dib::getY () [inline]

sacas el valor de la y

Devuelve:
la y

float obj_dib::getY_SIZE () [inline]

sacas el valor de la y_size (la mitad de la altura)

Devuelve:
la y_size

virtual void obj_dib::mover (obj_dib *_objetochocado_, int &_bloques_restantes_, int &_vidas_) [inline, virtual]

funcion virtual de mover. se implementa para misil y pelota. el resto de casos no hace nada

Parámetros:

objetochocado	no se utiliza
bloques_restantes	no se utiliza
vidas	no se utiliza

Reimplementado en **misil** (*p.5*) y **pelota** (*p.15*).

virtual void obj_dib::pressedKey (unsigned char*c*) [inline, virtual]

funcion virtual de presionar botones. se implementa para nave. el resto de casos no hace nada

Parámetros:

c	no se utiliza

Reimplementado en **nave** (*p.7*).

void obj_dib::setHeight (float*h*) [inline]

funcion set de la y_size. se utiliza para cambiarla

Parámetros:

y_size	el valor por el que se quiere cambiar

virtual void obj_dib::setVy (double*vy*) [inline, virtual]

sirve para cambiar la velocidad de la pelota o el misil fuera de sus clases

Nota:

se utiliza para el misil y la pelota

Reimplementado en **misil** (*p.6*).

void obj_dib::setWidth (float*w*) [inline]

funcion set de la x_size. se utiliza para cambiarla

Parámetros:

x_size	el valor por el que se quiere cambiar

void obj_dib::setX (float*x*) [inline]

Funciones set.

funcion set de la x. se utiliza para cambiarla

Parámetros:

x	el valor por el que se quiere cambiar

void obj_dib::setY (float*y*) `[inline]`

funcion set de la y. se utiliza para cambiarla

Parámetros:

y	el valor por el que se quiere cambiar

Documentación de los datos miembro

string obj_dib::name `[protected]`

nombre del objeto

Parámetros:

name	nombre

float obj_dib::x `[protected]`

coordenadas

Parámetros:

x	horizontal
y	vertical

float obj_dib::x_size `[protected]`

mitad del ancho y mitad de la altura

Parámetros:

x_size	mitad ancho
y_size	mitad del alto

La documentación para esta clase fue generada a partir de los siguientes ficheros:

42 obj_dib.h
43 obj_dib.cpp

Referencia de la Clase pared

Diagrama de herencias de pared

Métodos públicos

44 **pared** (float **x**, float y, float ancho, float altura)

Crea el pared a partir de los parametros que le pasas.

45 void **dibujar** () const

Dibuja la pared con los vertices hallados a partir de los parámetros del ladrillo.

Documentación del constructor y destructor

pared::pared (float*x*, float*y*, float*ancho*, float*altura*)

Crea el pared a partir de los parametros que le pasas.

Parámetros:

x	Posicion horizontal del pared
y	Posicion vertical del pared
ancho	Ancho del pared
altura	Largo del pared

Documentación de las funciones miembro

void pared::dibujar () const [virtual]

Dibuja la pared con los vertices hallados a partir de los parámetros del ladrillo.

Nota:

el color siempre es blanco

Implementa **obj_dib** (*p.7*).

La documentación para esta clase fue generada a partir de los siguientes ficheros:

46 pared.h

47 pared.cpp

Referencia de la Clase pelota

Diagrama de herencias de pelota

Métodos públicos

48 **pelota** (float **x**, float y, double vx, double vy)

Crea la pelota a partir de los parametros que le pasas.

49 void **dibujar** () const

Dibuja la pelota en funcion del radio. creando vertices en su circunferencia mediante un for.

50 void **mover** (**obj_dib** *objetochocado, int &bloques_restantes, int &vidas)

mueve la pelota, detecta choques, quita vidas y destruye ladrillos si fuera necesario

51 double **getVx** ()

sacas el valor de la velocidad en x. se utiliza para la pelota

52 double **getVy** ()

sacas el valor de la velocidad en y. se utiliza para la pelota

---

Documentación del constructor y destructor

pelota::pelota (float*x*, float*y*, double*vx*, double*vy*)

Crea la pelota a partir de los parametros que le pasas.

Parámetros:

x	Posicion horizontal de la pelota
y	Posicion vertical de la pelota
vx	velocidad horizontal de la pelota
vy	velocidad vertical de la pelota

Nota:

no introduces el ancho ni la altura por que el radio de la pelota siempre es 5

Documentación de las funciones miembro

void pelota::dibujar () const [virtual]

Dibuja la pelota en funcion del radio. creando vertices en su circunferencia mediante un for.

Nota:

el color depende de la posicion vertical de la pelota

Implementa **obj_dib** (*p.7*).

double pelota::getVx () [inline, virtual]

sacas el valor de la velocidad en x. se utiliza para la pelota

Devuelve:

la vx

Reimplementado de **obj_dib** (*p.9*).

double pelota::getVy () [inline, virtual]

sacas el valor de la velocidad en y. se utiliza para la pelota

Devuelve:

la vy

Reimplementado de **obj_dib** (*p.10*).

void pelota::mover (obj_dib *_objetochocado_, int &_bloques_restantes_, int &_vidas_) [virtual]

mueve la pelota, detecta choques, quita vidas y destruye ladrillos si fuera necesario

Parámetros:

objetochocado	objeto con el que comprueba si choca
bloques_restantes	resta bloques si hay choque
vidas	resta vida si choca con el fondo de la pantalla

Nota:

dentro llamas a las funciones **getBoundaries()**; y **choque()**. Es el mismo que nos paso Alberto Valero, salvo que varia el color en funcion de la posicion y

Reimplementado de **obj_dib** (*p.10*).

La documentación para esta clase fue generada a partir de los siguientes ficheros:

53 pelota.h
54 pelota.cpp

Índice

INDEX