Especificaciones del Entorno

1 Introducción

A los fines de que el grupo pueda desarrollar el trabajo practico se entregara un proyecto base que puede ser importado en el IDE Eclipse.

Este proyecto contiene el paquete entorno.jar que contiene la clase Entorno y la clase Herramientas.

La clase entorno al ser instanciada permite crear un objeto capaz de encargarse de la interfaz gráfica y de la interacción con el usuario. Así, el grupo solo tendrá que encargarse de la implementación de la lógica del juego.

2 La clase Juego

Dentro del proyecto se entrega otro paquete llamado juego que contiene una clase llamada Juego que debe respetar la siguiente signatura.:

```
package juego;

import entorno.Entorno;
import entorno.InterfaceJuego;

public class Juego extends InterfaceJuego
{
	// El objeto Entorno que controla el tiempo y otros
	private Entorno entorno;

	// Variables y métodos propios de cada grupo
	// ...

	Juego()
	{
		// Inicializa el objeto entorno
		this.entorno = new Entorno(this, "Demo entorno",
800, 600);

		// Inicializar lo que haga falta para el juego
		// ...


		// Inicia el juego!
		this.entorno.iniciar();
	}

	/**
	 * Durante el juego, el método tick() será ejecutado en
cada instante y por lo tanto es el método más importante de
esta clase. Aquí se debe actualizar el estado interno del
```

```
        juego para simular el paso del tiempo (ver el enunciado del
        TP para mayor detalle).
                */
               public void tick()
               {
                       // Procesamiento de un instante de tiempo
                       // ...

               }


               @SuppressWarnings("unused")
               public static void main(String[] args)
               {
                       Juego juego = new Juego();
               }
        }
```

Notar que las palabras clave “extends InterfaceJuego” en la definición de la clase son fundamentales para el buen funcionamiento del juego: no se brindan mas detalles al respecto dado que escapan al alcance de nuestra materia.

La clase Juego tal como se entrega solo contiene una única variable de instancia que es un objeto Entorno al cual llamamos entorno. El constructor del Juego construye una instancia del objeto Entorno con la siguiente sintaxis:

```
this.entorno = new Entorno(this, "Demo entorno", 800, 600);
```

El primer argumento es la palabra reservada this (La fundamentación de esto escapa al alcance de la materia), El segundo es un Sting que contiene el nombre de la ventana del objeto y los últimos argumentos son dos enteros que corresponden al ancho y el alto de la ventana medido en pixeles.

El constructor invoca el método “iniciar()” del objeto entorno. Cabe aclarar que la clase Juego deberá contener también la declaración de todos los objetos que el grupo decida incluir en el juego. En el constructor del juego se deberán inicializar los objetos que deben aparecer al inicio del juego. Los objetos que no aparezcan al inicio del juego (balas, nuevos personajes, etc) se inicializaran en el momento en el cual deban aparecer.

Finalmente tenemos el método tick() que deberá ser completado por el grupo con la lógica del juego que básicamente consiste en mostrar los objetos en la pantalla (utilizando los métodos adecuados del entorno que se explicaran luego), mover los objetos (modificando sus coordenadas a través de los métodos que el grupo defina), actualizar su estado (modificando las variables de instancia de los objetos a través de los métodos que el grupo defina), crear objetos (instanciándolos mediante new) y eliminar objetos (haciéndolos null) así como también analizar las órdenes del jugador (usuario humano del juego) a través del teclado y/o mouse y actuar en consecuencia. Por último, también suele ser necesario verificar si hay objetos interactuando entre si ya sea porque sus imágenes se superponen o hay cercanía entre sus coordenadas. Estos métodos suelen denominarse métodos de colisión y deben definirse de acuerdo con las condiciones especifica de cada juego.

3 Funcionamiento del entorno

En resumen, el objeto Entorno muestra una ventana en la pantalla dentro de la cual podemos dibujar los distintos elementos del juego y ejecuta el método tick() de manera indefinida hasta que cerremos la ventana. La secuencia funcionamiento es la siguiente:

1-El entorno borra la ventana

2-El entorno dibuja la ventana (con los gráficos definidos en el tick anterior)

3-Llama el método tick()

4-Ejecuta el bloque de código definido dentro del método tick()

5-Borra los buffers de teclado y mouse.

6-Espera 10milisegundos

7-Vuelve al paso 1:

El ciclo total del tick dura entre 15 y 30 milisegundos dependiendo de la velocidad de hardware donde sea ejecutado

El método main solo instancia un objeto Juego

```
public static void main(String[] args)
    {

        Juego juego = new Juego();
    }
```

El objeto entorno creado en el constructor del Juego recibe el juego en cuestión y mediante el método entorno.iniciar() se inicia el simulador. A partir de ahí, en cada instante de tiempo que pasa, el entorno ejecuta el metodo tick() del juego. Este es el método más importante de la clase Juego y aquí el juego debe actualizar su estado interno para simular el paso del tiempo.

Como mínimo se deben realizar las siguientes tareas:

Actualizar el estado interno de todos los objetos involucrados en la simulación.

Dibujar los mismos en la pantalla (ver más abajo como hacer esto).

Verificar si algún objeto aparece o desaparece del juego.

Verificar si hay objetos interactuando entre sí (colisiones, por ejemplo).

Verificar si los usuarios están presionando alguna tecla y actuar en consecuencia (ver más adelante como hacer esto).

4 Métodos del entorno

4.1 Métodos para dibujar

Para dibujar en pantalla y capturar las teclas presionadas, el objeto entorno dispone de los siguientes métodos, entre otros:

```
public void dibujarRectangulo(double x, double y,
double ancho, double alto, double angulo, Color color)
```

Dibuja un rectángulo centrado en el punto (x,y) de la pantalla, rotado en el ángulo dado.

```
public void dibujarTriangulo(double x, double y, int
altura, int base, double angulo, Color color)
```

,→ Dibuja un triángulo centrado en el punto (x,y) de la pantalla, rotado en el ángulo dado.

```
    public void dibujarCirculo(double x, double y,
double diametro, Color color),
```

→ Dibuja un circulo centrado en el punto (x,y) de la pantalla, del tamano dado

4.1.1 Dibujar imágenes

Para dibujar imágenes el entorno dispone de un método especifico Dibuja la imagen centrada en el punto (x,y) de la pantalla rotada en el ángulo dado

```
public void dibujarImagen(Image imagen, double x,
double y, double angulo)
```

El ángulo debe estar en radianes.

Este método esta sobrecargado y admite una versión con un parámetro adicional que es la escala. La escala debe ser un número estrictamente mayor que cero. Valores menores a 1 reducen la imagen y superiores a 1 la amplían.

```
public void dibujarImagen(Image imagen, double x,
double y, double angulo, double escala)
```

El método dibujarImagen() requiere como argumento de entrada un objeto de la clase Image de Java. Para obtenerlo debemos crear un objeto Image a partir de un archivo de imágenes. Esto se realiza con el método cargarImagen de la clase Herramientas.

La misma tiene la siguiente sintaxis:

```
public static Image cargarImagen(String archivo)
```

El String archivo debe contener (entre comillas) la ruta de acceso al archivo. Los formatos permitidos son bmp, jpg, png y gif. Es posible utilizar gif animados y png o gif con transparencia.

4.1.2 Ángulos

En los métodos anteriores los ángulos deben estar en radianes. En clase Herramientas se puede usar el método:

public static double radianes(**double** grados) que transforma de grados a radianes

4.2 Métodos para reproducir sonido

El entorno no tiene un soporte específico para a reproducción de sonido, pero puede utilizar el soporte de sonido estándar de Java

4.2.1 Java Clips

Es posible la reproducción de objetos de tipo Clip (del standard de Java).La clase Herramientas tiene 2 métodos que les facilitan esta tarea.

```
public static void play(String file)
```

Reproduce una vez el sonido

```
public static void loop(String file)
```

Reproduce el sonido indefinidamente, es decir reiniciándolo cada vez que termina.

El String file debe contener (entre comillas) la ruta de acceso al archivo. Los formatos permitidos son AU, WAV o AIFF.

ATENCIÓN:

Java no soporta de forma nativa el mp3. Si quieren usar sonido mp3 conviertanlos primero a wav

La mayoría de las computadoras de la universidad no tienen placa de sonido. Si quieren usar sonido deberían traer sus propias computadoras.

Los métodos play y loop lanzan una excepción si la computadora no tiene el hardware adecuado para reproducir sonido. Se recomienda el uso de estos métodos dentro de un bloque try catch

4.2.2 Secuencias midi

Se puede incorporar la reproducción de archivo de secuencias midi (.mid) para esto hay que instanciar un objeto Sequence y un objeto Sequencer en el constructor del entorno que previamente deben definirse como variables de instancia del juego.

Por ejemplo:

Sequence secuencia;

Sequencer secuenciador;

Y luego en el constructor del Juego usar lo siguiente:

```
try {
                    secuencia =
MidiSystem.getSequence(ClassLoader.getSystemResource("archivo.mid"));
                secuenciador = MidiSystem.getSequencer();
                  secuenciador.open();
                  secuenciador.setSequence(secuencia);
                  secuenciador.setTempoFactor( velocidad); //velocidad es un entero
                  secuenciador.setLoopCount(n); n es un entero
                  secuenciador.start();


                } catch (InvalidMidiDataException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                } catch (MidiUnavailableException e) {
                        e.printStackTrace();
                }
```

4.3 Métodos para escribir texto

```
public void escribirTexto(String texto, double x,
double y)
```

,→ Escribe el texto en las coordenadas x e y de la pantalla.

```
public void cambiarFont(String font, int tamano, Color
color)
     {
```

,→ Cambia la fuente para las próximas escrituras de texto según los párámetros recibidos.

Ejemplo:

```
entorno.cambiarFont("Arial", 18, Color.CYAN);
```

En el parámetro String font se debe colocar el nombre de alguna fuente que este cargada en la implementación de Java en particular que estén usando. En el APENDICE1 pueden encontrar un listado de las fuentes típicas que suelen estar habilitadas.

Si el sistema no reconoce el nombre del Font que eligieron usar utilizara el font por defecto.

También pueden consultar el Array entorno.fontDisponibles que es una variable de instancia del tipo Array de Strings que contiene la lista de todos los fonts.

Si agregan esta instrucción al constructor del juego:

```
System.out.println(Arrays.toString(entorno.fontDisponib
les));
```

Podrán ver en la consola todas las fonts disponibles apenas el juego arranque.

```
public void cambiarFont(String font, int tamano, Color
color, int tipo)
```

Este método es similar al anterior, pero permite modificar el tipo

```
entorno.cambiarFont("Tahoma", 18, miColor,
entorno.NEGRITA);
```

Los tipos admitidos son NEGRITA, ITALICA y NORMAL

4.4 Manejo del Color

4.4.1 Definición de colores

Tanto para el texto como para el fondo o figuras geométricas que se representen en el entorno pueden usar los colores disponibles en la clase Color o bien definir el que quieran con esta sencilla instrucción:

```
Color miColor = new Color (int R, int G, int B);
```

Donde los enteros R,G y B pueden tomar valores entre 1 y 255.

Pueden utilizar algún selector de colores online para que los ayude a determinar estos valores, por ejemplo:

https://htmlcolorcodes.com/es/selector-de-color/

4.4.2 Transparencia

El entorno soporta el manejo del canal Alpha El canal alfa en el contexto de RGB se refiere a un cuarto componente que se añade a los tres colores básicos (Rojo, Verde y Azul) para representar la transparencia de un píxel.

El canal alfa controla cuán visible o transparente es un píxel.

Su valor va de:

0 → completamente transparente

255 → completamente opaco

La sintaxis para incluir el canal Alpha es la siguiente:

```
Color miColor = new Color (int R, int G, int B,int A);
```

Donde los enteros R,G, B y A pueden tomar valores entre 1 y 255.

Por ejemplo, Color(0,0,255,128) representa un azul con aproximadamente un 50% de transparencia.

4.5 Métodos para detectar teclas presionadas o liberadas

```
public boolean estaPresionada(char key)
```
Indica si la tecla t esta presionada por el usuario en ese momento.

```
public boolean sePresiono(char key)
```

Indica si la tecla t fue presionada en durante ese tick (es decir, no estaba presionada en la última llamada a tick(), pero si en ésta). Este método puede ser útil para identificar

eventos particulares en un único momento, omitiendo tick() futuros en los cuales el usuario mantenga presionada la tecla en cuestión.

```
public boolean seLevanto(char key)
```

Indica si la tecla t fue levantada en durante ese tick (es decir, estaba presionada en la última llamada a tick(), y se liberó durante el tick). Este método puede ser útil para identificar eventos particulares en el momento exacto en que una tecla es liberada.

Notar que estos métodos reciben como parámetro un char que representa “la tecla” por la cual se quiere consultar, e.g., sePresiono(’A’) o estaPresionada(’+’). Algunas teclas no pueden escribirse directamente como un char como por ejemplo las flechas de dirección del teclado. Para ellas, dentro de la clase entorno se encuentran definidas las constantes que se detallan en el APENDICE 2

De esta manera, para ver, por ejemplo, si el usuario está presionando la flecha hacia arriba se puede consultar, por ejemplo, si estaPresionada(entorno.TECLA_ARRIBA)

4.6 Métodos para detectar uso de botones del mouse

```
public boolean sePresionoBoton(int bot)
```

```
public boolean estaPresionado(int bot)
```

```
public boolean seLevantoBoton(int bot)
```

Estos métodos se utilizan para determinar las acciones que el jugador realiza con los botones del mouse

El argumento bot solo admite 3 valores

entorno. BOTON_IZQUIERDO

entorno. BOTON_CENTRAL

entorno. BOTON_DERECHO

4.7 Métodos para determinar la posición del mouse.

```
public int mouseX() y public int mouseY()
```

Estos métodos devuelven las coordenadas x y del puntero del mouse al inicio del tick

Este método devuelve las coordenadas cuando el puntero esta dentro de la ventana del entorno independientemente de si algún botón se encuentra presionado o no. Si presionamos algún botón dentro del entorno y arrastramos el mouse con el botón presionado puede devolver las coordenadas también por fuera del entorno. En este caso los valores de las coordenadas podrán ser mayores al alto y ancho del entorno o negativas dependiendo de la posición del cursor relativa al extremo superior izquierdo de la ventana del entorno.

```
public boolean mousePresente()
```

Este método devuelve TRUE si el mouse esta dentro del entorno al inicio del tick o FALSE en caso contrario.

4.8 Métodos generales del entorno

```
public int ancho()
```

Devuelve la cantidad de pixels de ancho que tiene el entorno.

```
public int alto()
```

Devuelve la cantidad de pixels de alto que tiene el entorno.

```
public int numeroDeTick()
```

Devuelve el número de tick, es decir la cantidad del ticks transcurridos desde el inicio del juego.

```
public int tiempo() {
```

Devuelve la cantidad de milisegundos transcurridos desde el inicio del Juego

```
public void colorFondo(Color c)
```

Este método modifica el color de fondo del entorno. Por default al arrancar es negro.

5 APENDICE 1

Fuentes disponibles en las implementaciones estándar de Java. (Esto puede variar para distintos tipos de implementaciones de software).

Agency FB
Algerian
Arial
Arial Black
Arial Narrow
Arial Rounded MT Bold
Bahnschrift
Baskerville Old Face
Bauhaus 93
Bell MT
Berlin Sans FB
Berlin Sans FB Demi
Bernard MT Condensed
Blackadder ITC
Bodoni MT
Bodoni MT Black
Bodoni MT Condensed
Bodoni MT Poster Compressed
Book Antiqua
Bookman Old Style
Bookshelf Symbol 7
Bradley Hand ITC
Britannic Bold
Broadway
Brush Script MT
Calibri
Calibri Light
Californian FB
Calisto MT
Cambria
Cambria Math
Candara
Candara Light

Castellar
Centaur
Century
Century Gothic
Century Schoolbook
Chiller
Colonna MT
Comic Sans MS
Consolas
Constantia
Cooper Black
Copperplate Gothic Bold
Copperplate Gothic Light
Corbel
Corbel Light
Courier New
Curlz MT
Dialog
DialogInput
Dubai
Dubai Light
Dubai Medium
Ebrima
Edwardian Script ITC
Elephant
Engravers MT
Eras Bold ITC
Eras Demi ITC
Eras Light ITC
Eras Medium ITC
Felix Titling
Footlight MT Light
Forte
Franklin Gothic Book

Franklin Gothic Demi
Franklin Gothic Demi Cond
Franklin Gothic Heavy
Franklin Gothic Medium
Franklin Gothic Medium Cond
Freestyle Script
French Script MT
Gabriola
Gadugi
Garamond
Georgia
Gigi
Gill Sans MT
Gill Sans MT Condensed
Gill Sans MT Ext Condensed Bold
Gill Sans Ultra Bold
Gill Sans Ultra Bold Condensed
Gloucester MT Extra Condensed
Goudy Old Style
Goudy Stout
Haettenschweiler
Harlow Solid Italic
Harrington
High Tower Text
HoloLens MDL2 Assets
Impact
Imprint MT Shadow
Informal Roman
Ink Free
Javanese Text

Jokerman
Juice ITC
Kristen ITC
Kunstler Script
Leelawadee
Leelawadee UI
Leelawadee UI Semilight
Lucida Bright
Lucida Calligraphy
Lucida Console
Lucida Fax
Lucida Handwriting
Lucida Sans
Lucida Sans Typewriter
Lucida Sans Unicode
Magneto
Maiandra GD
Malgun Gothic
Malgun Gothic Semilight
Marlett
Matura MT Script Capitals
Microsoft Himalaya
Microsoft JhengHei
Microsoft JhengHei Light
Microsoft JhengHei UI
Microsoft JhengHei UI Light
Microsoft New Tai Lue
Microsoft PhagsPa
Microsoft Sans Serif
Microsoft Tai Le

Microsoft Uighur
Microsoft YaHei
Microsoft YaHei Light
Microsoft YaHei UI
Microsoft YaHei UI Light
Microsoft Yi Baiti
MingLiU-ExtB
MingLiU_HKSCS-ExtB
Mistral
Modern No. 20
Mongolian Baiti
Monospaced
Monotype Corsiva
MS Gothic
MS Outlook
MS PGothic
MS Reference Sans Serif
MS Reference Specialty
MS UI Gothic
MT Extra
MV Boli
Myanmar Text
Niagara Engraved

Niagara Solid
Nirmala UI
Nirmala UI Semilight
NSimSun
OCR A Extended
Old English Text MT
Onyx
Palace Script MT
Palatino Linotype
Papyrus
Parchment
Perpetua
Perpetua Titling MT
Playbill
PMingLiU-ExtB
Poor Richard
Pristina
Rage Italic
Ravie
Rockwell
Rockwell Condensed
Rockwell Extra Bold
Sans Serif Collection
SansSerif
Script MT Bold

Segoe Fluent Icons
Segoe MDL2 Assets
Segoe Print
Segoe Script
Segoe UI
Segoe UI Black
Segoe UI Emoji
Segoe UI Historic
Segoe UI Light
Segoe UI Semibold
Segoe UI Semilight
Segoe UI Symbol
Segoe UI Variable
Serif
Showcard Gothic
SimSun
SimSun-ExtB
Sitka Text
Snap ITC
Stencil
Sylfaen
Symbol
Tahoma
Tempus Sans ITC
Times New Roman
Trebuchet MS

Tw Cen MT
Tw Cen MT Condensed
Tw Cen MT Condensed Extra Bold
Verdana
Viner Hand ITC
Vivaldi
Vladimir Script
Webdings
Wide Latin
Wingdings
Wingdings 2
Wingdings 3
Yu Gothic
Yu Gothic Light
Yu Gothic Medium
Yu Gothic UI
Yu Gothic UI Light
Yu Gothic UI Semibold
Yu Gothic UI Semilight]

6 APENDICE 2

Valor	Tecla Representada
TECLA_ARRIBA	Flecha hacia arriba
TECLA _ABAJO	Flecha hacia abajo
TECLA_DERECHA	Flecha hacia la derecha
TECLA_IZQUIERDA	Flecha hacia la izquierda
TECLA_ENTER	Tecla “enter”
TECLA_ESPACIO	Barra espaciadora

TECLA_CTRL	Tecla “control”
TECLA_ALT	Tecla “alt”
TECLA_SHIFT	Tecla “shift”
TECLA_INSERT	Tecla “ins”
TECLA_DELETE	Tecla “del” (o “supr”)
TECLA_INICIO	Tecla “start” (o “inicio”)
TECLA_FIN	Tecla “end” (o “fin”)
TECLA_ESCAPE	Tecla scape

INDICE

Tabla de contenido