TECNOLOGIA DE LA PROGRAMACIÓN DE VIDEOJUEGOS
Facultad de Informática
Curso 2016-2017

PRÁCTICA 3: EL PINCHA GLOBOS (3)
Fecha de entrega: 18 de enero de 2017

El objetivo de esta práctica es introducir las excepciones, los atributos funcionales y transiciones entre estados del juego.

EL JUEGO (3ª parte)

El juego de la práctica 2 se amplía a un juego con cuatro estados: Menú, Play, Game over y Pausa. El estado Menú permite elegir entre jugar al juego (botón "Play") o salir de la aplicación (botón "Exit"). El estado Play se corresponde con el juego actual. Al estado Game over se llega al terminar la partida, y permite ver los puntos obtenidos y cambiar al estado Menú. Al estado Pausa se llega pulsando la tecla "Esc" desde el estado Play, y permite volver a Play para continuar la partida, o cambiar al estado Menú. Las transiciones son:

- Menú -> Play, al pulsar en el botón "Play". Si se pulsa el botón "Exit" terminará la aplicación.
- Play -> Game over, al terminar la partida.
- Play -> Pausa, al pulsar la tecla "Esc".
- Game over -> Menú, al pulsar el botón "Menu". Si se pulsa el botón "Score" se mostrarán los puntos obtenidos.
- Pausa -> Menú, al pulsar en el botón "Menu".
- Pausa -> Play, al pulsar el botón "Resume".

Cada uno de los estados sigue el esquema de un juego. La actual clase **`JuegoPG`**, consta del estado Play, la inicialización y finalización de las librerías, la carga de texturas y sonido, y la gestión de eventos.
Ahora separamos del **`JuegoPG`** el estado Play y dejamos los otros elementos en la clase **`JuegoPG`**.

Además la gestión de errores de se realizará con excepciones que, recogerá **`main`** para informar y terminar la aplicación. Para mostrar los mensajes de error utiliza la función **`SDL_ShowSimpleMessageBox()`**.

Añade al proyecto las siguientes clases:

- **EstadoJuego:** Clase abstracta pura (interfaz) raíz de la jerarquía que representa los estados de un juego, con los métodos abstractos: **`void draw()`**, **`void onClick()`**y **`void update()`**.
- **EstadoPG:** Clase abstracta que hereda de **`EstadoJuego`** y que implementa utilidades para las subclases (**`MenuP`**, **`PlayPG`**, **`GameOver`** y **`Pausa`**): puntero al juego, vector de objetos,…
- **PlayPG:** Para el estado Play, hereda de **`EstadoPG`**.
- **Boton:** Clase que hereda de **`ObjetoPG`** con un atributo para la función a ejecutar en caso de ser pulsado (**`CallBack_t * cb`**). El tipo de la función será: **`typedef void CallBack_t(JuegoPG* jg);`** Añade a la constructora un parámetro del mismo tipo que el atributo.
- **MenuPG:** Para el estado Menú, hereda de **`EstadoPG`** y consta de dos botones ("Play" y "Exit"). Para cada botón define una función de tipo **`CallBack_t`**.
- **GameOver:** Para el estado Game over, hereda de **`EstadoPG`** y consta de uno o dos botones ("Score" y "Menu"). Para cada botón define una función de tipo **`CallBack_t`**.
- **Pausa:** Para el estado Pausa, hereda de **`EstadoPG`** y consta de dos botones ("Menu" y "Resume"). Para el botón "Resume" define una función de tipo **`CallBack_t`**.
- **Error:** Para las excepciones que lanzará el juego al iniciarse. Con un atributo de tipo **`string`** para el mensaje del error, constructora con un argumento para el mensaje, y el método **`const string& mensaje()`** para consultarlo. Puedes definir subclases para los errores de las distintas librerías que utilizas (SDL, Mix,…).

Además deberás modificar la clase **`JuegoPG`** eliminando todo lo referente a la clase **`PlayPG`** y añadiendo un atributo, **`estados`**, para la pila de estados del juego. Al estado en curso se podrá acceder a través del método **`EstadoJuego* topEstado()`**, y las transiciones entre estados se realizarán con los métodos públicos: **`void changeState(EstadoJuego* newSt)`**, **`void pushState(EstadoJuego* newState)`** y **`void popState()`**. Añade también **`void setSalir()`** para salir del juego.

Sugerencia: Empieza con las excepciones: define las clases de excepciones, con raíz **`Error`**, modifica la clase **`JuegoPG`** para genere excepciones en los casos de error, y la función **`main`** para que las capture, muestre un mensaje y libere todos los recursos.

Añade y define las clases **`EstadoJuego`**, **`EstadoPG`**, y **`PlayPG`** modificando la clase **`JuegoPG`**. Una vez que funcione correctamente (como la práctica 2), añade las clases **`Boton`**, **`MenuPG`** y **`GameOver`**, y las transiciones. Por último añade la clase **`Pausa`** y sus transiciones. Utiliza: tipo de evento SDL_KEYUP (en evento.type) y tecla SDLK_ESCAPE (en `evento.key.keysym.sym`).
Cada una de estas clases debe realizar la gestión de errores con excepciones

Entrega de la práctica: En la tarea del campus virtual "Entrega práctica 3" dentro de la fecha límite (18 de enero) debes subir el directorio de la solución comprimido (.zip). Recuerda incluir un archivo autores.txt con los nombre de los componentes del grupo y limpiar la solución antes de comprimirla.

Evaluación: el miércoles 18 de enero en la clase de laboratorio.

Mejoras:
Excepciones: Si has puesto sonido y se produce un error al iniciar la librería o cargar los archivos, se muestra un mensaje y el juego se ejecuta sin sonido. Si utilizas texto y se produce un error al iniciar la librería o cargar la fuente, se muestra un mensaje y el juego se ejecuta sin contador.

Anima los botones. Utiliza el evento SDL_MOUSEMOTION y la función SDL_GetMouseState(&mx, &my) para actualizar la posición del ratón en el juego. La clase botón usará la posición del ratón en update().
Define clases para la ventana, la máquina de estados, la tabla de texturas.
Trata los efectos de sonido de la misma forma que las texturas.