La Clase SDLUtils
(y sus clases auxiliares)

TPV 2

Samir Genaim

SDLUtils y sus clases auxiliares

- Conjunto de clases, que vamos a usar en TPV2, para facilitar el uso de SDL
- Lo uso en clases, vosotros podéis usar en las prácticas, es opcional. Pero si usa también en el código que os proporcionamos para el examen.
- Incluye clases para usar texturas, fuentes, y sonido que se pueden usar de manera más fácil (son wrappers de estructuras/operaciones de SDL)
- Tablas de texturas, fuentes, sonido, que se cargan al principio del juego (en lugar de crear durante el juego).
- Generador de números aleatorios
- Timer virtual
- ...

La Clase Font

Es un wrapper de TTF_Font

```
Font f("arial.ttf",18);
```

- Crea fuente de tamaño 18 a partir del archivo "arial.ttf"
- Su método renderText se usa para renderizar texto, devuelve SDL_Surface* (se usa principalmente en la clase Texture, casi nunca se usa directamente) — en los siguientes ejemplos, color y colorFondo son de tipo SDL_Color

```
f.renderText("Game Over!", color);
f.renderText("Game Over!", color, colorFondo);
```

La Clase Texture

Es un wrapper de SDL_Texture (para texto y imagen)

```
Texture t(renderer, "image.png") ;
Texture t(renderer, "Game Over!", font, color);
Texture t(renderer, "Game Over!", font, color, colorFondo);
```

- Para crear la textura se necesita SDL_Renderer* (el primer parámetro). Normalmente usamos el que se ha creado en la instancia de SDLUtils (ver la parte sobre la clase SDLUtils).
- El parámetro "font" es de tipo Font (que acabamos de ver), y color y colorFondo son de tipo SDL_Color

```
// consultar la anchura/altura de la textura
t.width();
t.height();
```

La Clase Texture

```
// Renderizar toda la textura en la posición (x,y) de la
// ventana - (x,y) es la equina superior-izquierda
t.render(x,y);
```

```
// Renderizar la parte 'src' (SDL_Rect) de la textura en
// la parte 'dest' (SDL_Rect) de la ventana
t.render(src, dest);
```

```
// Renderizar toda la textura en la parte 'dest' de la
// ventana
t.render(dest);
```

```
// Renderizar toda la textura en la parte 'dest' de la
// ventana  con rotación 'r' (respect al centro de 'dest')
t.render(dest,r);
```

La Clase Texture

```
// Renderizar la parte 'src' (SDL_Rect) de la textura en
// la parte 'dest' (SDL_Rect) de la ventana con rotación
// 'r' respecto al punto 'p' y voltear según el valor 'flip'
//
t.render(src, dest, r, p, flip);
```

- "p" es un puntero a SDL_Point, si es nullptr la rotación se hace respecto al centro de 'dest'
- "flip" es de tipo SDL_RendererFlip, los posibles valores son SDL_FLIP_NONE, SDL_FLIP_HORIZONTAL, SDL_FLIP_VERTICAL
- "flip" y "p" son opcionales (usan SDL_FLIP_NONE y nullptr como valores por defecto)

La Clase SoundEffect

Es un wrapper de Mix_Chunk (efecto de sonido de la librería SDL_mixer)

```
SoundEffect s("shoot.wav");
```

La librería SDL_mixer permite reproducir el sonido en uno de los canales (que se mezclan para reproducir el sonido a la vez)

```
// reproducir el sonido en el primer canal libre
s.play();

// reproducir el sonido en el primer canal libre y repítelo
// 3 veces (en total serían 4)
s.play(3);

// reproducir el sonido en el canal 2 y repítelo 3 veces
// (en total serían 4)
s.play(3,2);
```

La Clase SoundEffect

```
// detener (la reproducción de sonido) de todos los
// canales o de un canal específico
SoundEffect::pauseChannel();
SoundEffect::pauseChannel(3);
```

```
// reanudar (la reproducción de sonido) de todos los
// canales o de un canal específico
SoundEffect::resumeChannel();
SoundEffect::resumeChannel(3);
```

```
// parar (la reproducción de sonido) de todos los canales
// o de un canal específico — no se puede reanudar
SoundEffect::haltChannel();
SoundEffect::haltChannel(3);
```

La Clase SoundEffect

```
// cambiar el volumen de todos los canales o de un canal
// específico — el valor entre 0 y 128
SoundEffect::setChannelVolume(57);
SoundEffect::setChannelVolume(57,3);
```

```
// cambiar el número de canales - por defecto la
// instancia de SDLUtils lo inicializa a 8
SoundEffect::setNumberofChannels(8);
```

La Clase Music

Es un wrapper de Mix_Music (música de SDL_mixer). Se usa para música de fondo, etc. El sonido tiene mas calidad del sonido de SoundEffect. Hay sólo un canal que se reproduce en paralelo a los efectos de sonido

```
Music s("imperial_march.wav") ;
```

```
// reproducir la música en bucle - cancela lo que está sonando
s.play();
```

```
// reproducir la música y repítela 3 veces (en total serían 4)
s.play(3);
```

```
// cambiar el volumen de la música — el valor entre 0 y 128
Music::setMusicVolume(57);
```

```
// detener/reanudar/parar la reproducción de música
Music::pause();
Music::resume();
Music::halt();
```

Inicializar antes de Usar

Las clases Font, Texture, SoundEffect y Music se pueden usar sólo después de haber inicializado las librerías SDL_ttf, SDL_image y SDL_mixer. Esto se hace automáticamente si usas la clase SDLUtils para crear la ventana (ver mas adelante)

```
// initialize SDL_ttf
int ttfInit_r = TTF_Init();
assert(ttfInit_r == 0);

// initialize SDL_image
int imgInit_ret = IMG_Init(IMG_INIT_JPG | IMG_INIT_PNG | ... );
assert(imgInit_ret != 0);

// initialize SDL_mixer
int mixOpenAudio = Mix_OpenAudio(44100, MIX_DEFAULT_FORMAT, 2, 2048);
assert(mixOpenAudio == 0);
int mixInit_ret = Mix_Init(MIX_INIT_FLAC | MIX_INIT_MOD | ...);
assert(mixInit_ret != 0);
SoundEffect::setNumberofChannels(8); // we start with 8 channels
```

RandomNumberGenerator

Una clase para generar números aleatorios ...

```
RandomNumberGenerator r(seed);

// Devuelve un entero aleatorio
r.nextInt();

// Devuelve un entero aleatorio n tal que low ≤ n < high
r.nextInt(low,high);
```

- La semilla 'seed' es opcional, la constructora por defecto usa std::time(0).
- Usando la misma semilla genera la misma secuencia de números — útil para depurar programas que usan números aleatorios (para reproducir el mismo comportamiento)

VirtualTime

Es un timer virtual, que permite detener el tiempo, etc.

```
VirtualTime vt;

// Resetear el tiempo (empieza de nuevo desde 0)
vt.reset();

// Devuelve el tiempo (virtual) actual
vt.currTime();

// Detener el tiempo
vt.pause();

// Reanudar el tiempo
vt.resume();
```

La Clase SDLUtils

La clase SDLUtils es un Singleton. Podemos inicializar usando

```
SDLUtils::init("Ping Pong", 800, 600, "resources.json")
```

- Abre una ventana de tamaño 800x600 con el titulo "Ping Pong"
- Inicializa las tablas de imágenes, texto, fuentes, sonido, y música usando la información desde "resources.json"
- Se puede acceder a la instancia usando SDLUtils::instance() que devuelve un puntero a una instancia de SDLUtils
- … o usando el método sdlutils() que devuelve una referencia (no puntero) al objeto, es decir, devuelve *SDLUtils::instance()

Ejemplo de resources.json

```
{
 "fonts" : [
    { "id" : "ARIAL16", "file" : "resources/fonts/ARIAL.ttf", "size": 16 },
    { "id" : "ARIAL24", "file" : "resources/fonts/ARIAL.ttf", "size": 24 }
 ],

  "images" : [
    { "id" : "sdl_logo", "file" : "resources/images/SDL_logo.png" },
    { "id" : "tennis_ball", "file" : "resources/images/tennis_ball.png" },
    { "id" : "star", "file" : "resources/images/star.png"}
  ],

  "messages" : [
    { "id" : "PressAnyKey", "text" : "Press Any Key", "font" : "ARIAL24", "color" : "0x112233ff", "bg" : "0xffffffff" },
    { "id" : "HelloSDL", "text" : "Hello SDL!", "font" : "ARIAL48", "color" : "0x0000ffff" }
  ],

  "sounds" : [
    { "id" : "gunshot", "file" : "resources/sound/gunshot.wav" },
    { "id" : "explosion", "file" : "resources/sound/explosion.wav" }
  ],

  "musics" : [
    { "id" : "beat", "file" : "resources/sound/beat.wav" },
    { "id" : "imperial_march", "file" : "resources/sound/imperial_march.wav" }
  ]
}
```

Font

Texture

Texture

SoundEffect

Music

El value de "id" se usa como identificador para referirse al recurso correspondiente en el programa

La Clase SDLUtils

```
// Devuelve la anchura/altura de la ventana
sdlutils().width();
sdlutils().height();

// Devuelve los punteros a SDL_Renderer y SDL_Window que se han
// usado para crear la ventana – útil para crear instancias de Texture
sdlutils().renderer(); // usar cuando creas instancias de Texture
sdlutils().window();

// Borrar/presentar el rendered – llamar antes/después de renderizar
// los objetos de juego, etc.
sdlutils().clearRenderer();
sdlutils().presentRenderer(color); // color es de tipo SDL_Color (opcional)

// Cambia el modo de pantalla a completa/ventana.
sdlutils().toggleFullScreen();

// Mostrar/ocultar el cursor
sdlutils().showCursor();
sdlutils().hideCursor();
```

La Clase SDLUtils

```
// Devuelve el tiempo real actual (usando SDL_GetTicks())
sdlutil().currRealTime()

// Devuelve una referencia a VirtualTimer (siempre la misma instancia)
auto &vt = sdlutil().virtualTimer()

// Devuelve una referencia a RandomNumberGenerator (siempre la
// misma instancia)
auto &r = sdlutil().rand()

// Devuelve una referencia (no puntero) al recurso correspondiente (de los
// declarados en resources.json). Si lo quieres como puntero usar
// &sdlutils()...
auto &font = sdlutil().fonts().at("id"); // referencia a Font
auto &img = sdlutil().images().at("id"); // referencia a Texture
auto &msg = sdlutil().msgs().at("id"); // referencia a Texture
auto &sound = sdlutil().soundEffects().at("id"); // referencia a SoundEffect
auto &music = sdlutil().music().at("id"); // referencia a Music
```

macros.h

Incluye algunos macros para facilitar la creación de SDL_Color, SDL_Rect, etc.

- build_sdlcolor: construye un SDL_Color a partir de un número 0x1244caff or un string "0x1244caff" (0x es para números hexadecimal en c++). Cada 2 dígitos representan un atributo <r,g,b,a> de SDL_Color. En este ejemplo construye el SDL_Color { 0x12, 0x44, 0xca, 0xff }.

- build_sdlrect: construye SDL_Rect a partir de un Vector2D y anchura/altura, simplemente evita hacer static_cast<int>(...)

- COLOREXP(color): un macro que se expande a la secuencia de valores color.r, color.g, color.b, color.a — es para evitar escribir toda la secuencia, p.ej., al llamar a SDL_RenderDrawLine

InputHandler

Ver las diapositivas de Input Handler (Controler)