

Programación de videojuegos para consolas portátiles

Israel López Fernández (Puck2099)

ilopez@retrowip.com

<http://www.retrowip.com>

Introducción

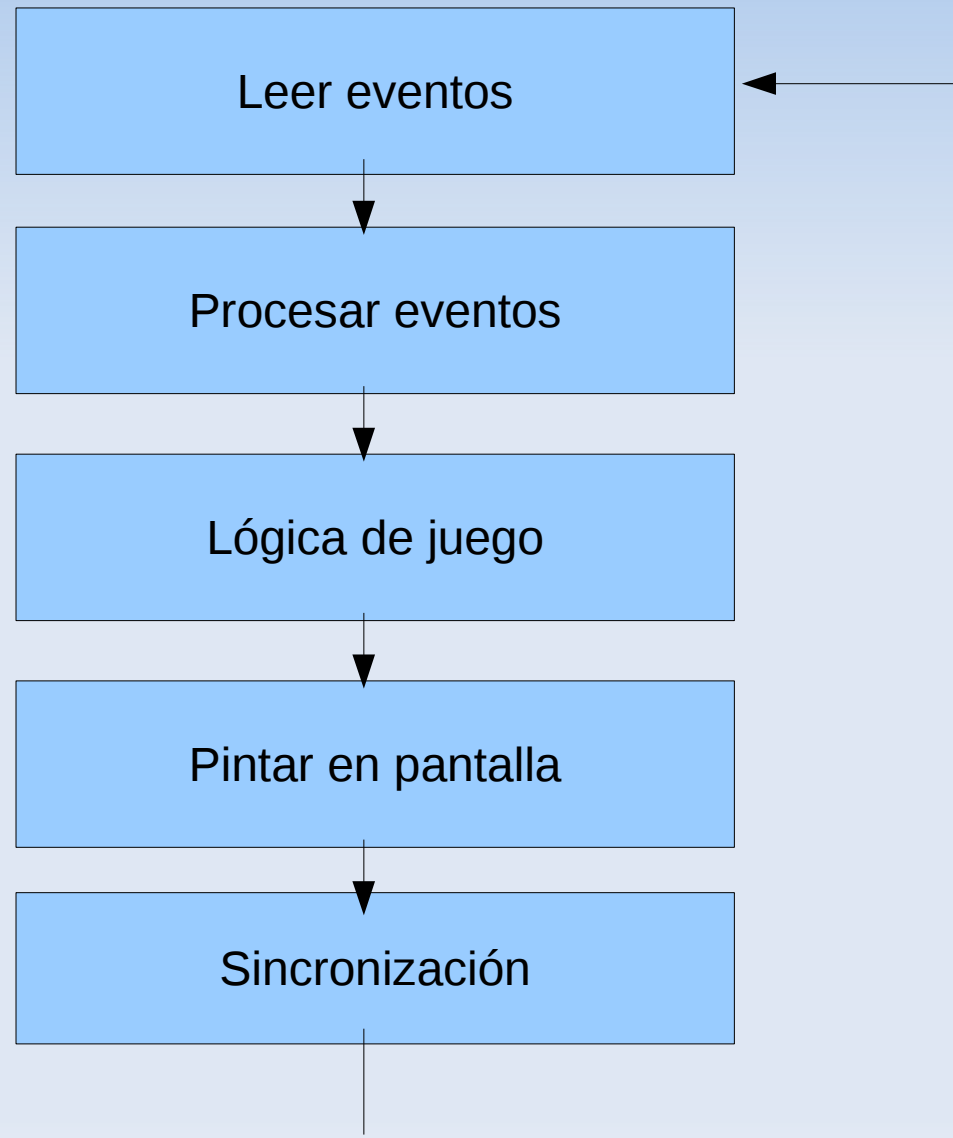
Lenguajes de programación

- Compilados (C/C++, ASM, Pascal, ADA...)
 - Rápidos y potentes.
 - Complejos.
 - Usados para el motor.
- Interpretados (JAVA, Python, Fenix...)
 - Lento y más limitados.
 - En general más fácil de usar y portables.
 - No necesita recompilar todo el código.
 - Útiles para la Inteligencia Artificial.

Bibliotecas multimedia

- Abstraen acceso al hardware.
- Muchas disponibles: SDL, Allegro, DirectX, OpenGL...
- Ventajas SDL:
 - Multiplataforma (PC, GP2X, PSP, Dreamcast...).
 - Eficientes.
 - Cubre todo aspecto multimedia.

El "game loop"



Subsistema de video

Mostrando gráficos (I)

- Toda imagen definida por una Superficie (la pantalla también).
- Configura pantalla:
 - `SDL_Surface* SDL_SetVideoMode(int width, int height, int bitsperpixel, Uint32 flags);`
- Carga imagen:
 - `SDL_Surface *SDL_LoadBMP(const char *file);`
- Copia superficies ("blittings"):
 - `int SDL_BlitSurface(SDL_Surface *src, SDL_Rect *srcrect, SDL_Surface *dst, SDL_Rect *dstrect);`

Mostrando gráficos (II)

- Define rectángulos:

```
typedef struct{  
    Sint16 x, y;  
    Uint16 w, h;  
} SDL_Rect;
```

- Intercambia buffers de pantalla:

- `int SDL_Flip(SDL_Surface* screen);`

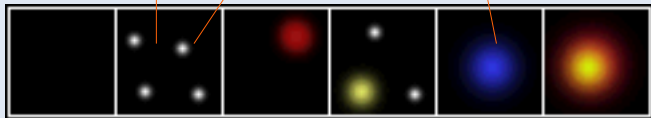
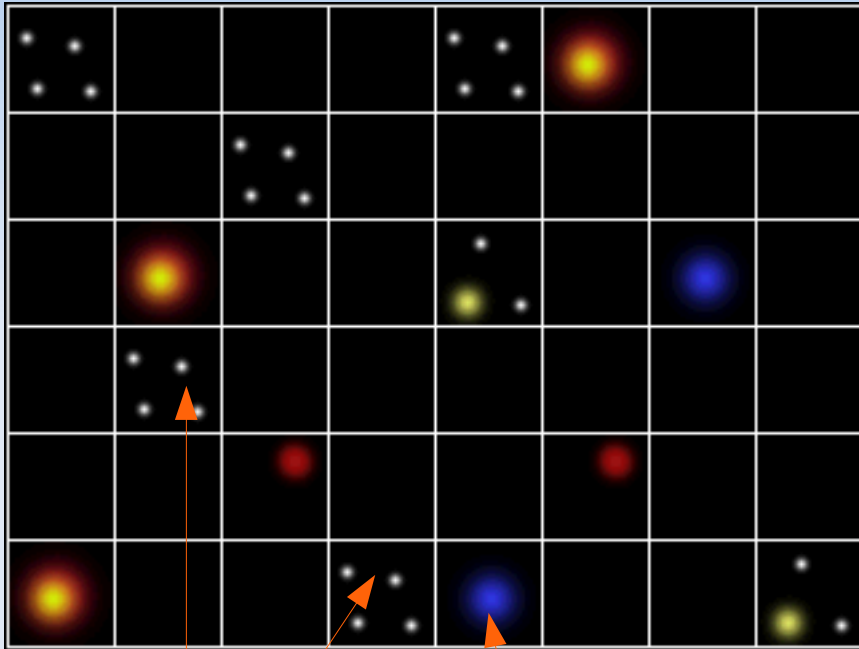
Transparencia y canal alfa

- Establece color transparente:
 - `int SDL_SetColorKey(SDL_Surface *surface, Uint32 flag, Uint32 key);`
- Cambia valor del canal alfa:
 - `int SDL_SetAlpha(SDL_Surface *surface, Uint32 flags, Uint8 alpha);`

Tiles y Sprites

- Tiles:
 - Imágenes con posición prefijada (mapa de tiles).
 - Usados para pintar fondos.
 - Permite ahorrar memoria repitiendo pequeñas imágenes a modo de mosaico.
- Sprites:
 - Imágenes con posición variable.
 - Normalmente pueden colisionar entre ellos.
 - Usados para pintar personajes y objetos.

Mapa de tiles



1	0	0	0	1	5	0	0
0	0	1	0	0	0	0	0
0	5	0	0	3	0	4	0
0	1	0	0	0	0	0	0
0	0	2	0	0	2	0	0
5	0	0	1	4	0	0	3

Colisiones entre sprites

- Precisión: comprobar píxel a píxel.
- Velocidad: encuadrar sprites en rectángulos.
 - Problema: Falsos positivos.



Subsistema de sonido

Reproduciendo sonidos (I)

```
typedef struct{
    int freq;
    Uint16 format;
    Uint8 channels;
    Uint8 silence;
    Uint16 samples;
    Uint32 size;
    void (*callback)(void *userdata, Uint8 *stream, int
    len);
    void *userdata;
} SDL_AudioSpec;
```

Reproduciendo sonidos (II)

- Abre el dispositivo de audio:
 - `int SDL_OpenAudio(SDL_AudioSpec *desired, SDL_AudioSpec *obtained);`
- Pausa o reproduce el audio:
 - `void SDL_PauseAudio(int pause_on);`

SDL_Mixer: Sonidos

- Los sonidos se definen como "chunks".
- Cargar sonido:
 - `Mix_Chunk *Mix_LoadWAV(const char *fname);`
- Reservar canales:
 - `int Mix_AllocateChannels(int numchans);`
- Reproducir sonido en canal:
 - `int Mix_PlayChannel (int channel, Mix_Chunk *chunk, int loops);`

SDL_Mixer: Melodías

- Canal exclusivo para melodías.
- Carga melodía:
 - `Mix_Music * Mix_LoadMUS (const char * file)`
- Reproduce melodía:
 - `Int Mix_PlayMusic (Mix_Music * music, int loops)`

Detección y gestión de eventos

Gestión de eventos (I)

- Cualquier interacción del usuario con el juego es un evento.
- Los eventos generados se anidan de forma que no se pierda ninguno.
- Eventos principales:
 - Teclado.
 - Joystick.
 - Ratón.

Gestión de eventos (II)

- Espera indefinidamente hasta el siguiente evento:
 - `int SDL_WaitEvent(SDL_Event *event);`
- Extrae un evento de la cola de eventos:
 - `int SDL_PollEvent(SDL_Event *event);`
- Inserta un evento en la cola de eventos:
 - `int SDL_PushEvent(SDL_Event *event);`

Teclado (I)

- Evento de teclado:

```
typedef struct{  
    Uint8 type;  
    Uint8 state;  
    SDL_keysym keysym;  
} SDL_KeyboardEvent;
```

Teclado (II)

- Reconocimiento de tecla:

```
typedef struct{
    Uint8 scancode;
    SDLKey sym;
    SDLMod mod;
    Uint16 unicode;
} SDL_keysym;
```

Joystick (I)

- Pulsación de un eje:

```
typedef struct{  
    Uint8 type;  
    Uint8 which;  
    Uint8 axis;  
    Sint16 value;  
} SDL_JoyAxisEvent;
```

Joystick (II)

- Pulsación de un botón:

```
typedef struct{  
    Uint8 type;  
    Uint8 which;  
    Uint8 button;  
    Uint8 state;  
} SDL_JoyButtonEvent;
```

Ratón (I)

- Movimiento del ratón:

```
typedef struct{  
    Uint8 type;  
    Uint8 state;  
    Uint16 x, y;  
    Sint16 xrel, yrel;  
} SDL_MouseMotionEvent;
```

Ratón (II)

- Pulsación de un botón del ratón:

```
typedef struct{  
    Uint8 type;  
    Uint8 button;  
    Uint8 state;  
    Uint16 x, y;  
} SDL_MouseButtonEvent;
```

Temporización

Medir tiempos

- Pausar la ejecución:
 - `void SDL_Delay(Uint32 ms);`
- Medir el tiempo:
 - `Uint32 SDL_GetTicks(void)`
- Poner un temporizador:
 - `SDL_TimerID SDL_AddTimer(Uint32 interval, SDL_NewTimerCallback callback, void* param);`

Semáforos

- Si semáforo ≤ 0 entonces hilo detenido.
- Crear semáforo:
 - `SDL_sem *SDL_CreateSemaphore(Uint32 initial_value);`
- Subir semáforo (incrementa contador):
 - `int SDL_SemPost(SDL_sem *sem);`
- Bajar semáforo (decrementa contador):
 - `int SDL_SemWait(SDL_sem *sem);`

Aplicación en game loop

- Esperas fijas (SDL_Delay):
 - Lleva a tasa de refresco irregular.
- Espera activa (comparación de ticks):
 - Buena temporización, pero desperdicia recursos.
- Semáforos:
 - Buena temporización y espera pasiva (duerme el hilo), pero incrementa complejidad.

Enlaces de interés

- <http://www.retrowip.com>
- <http://www.gp32spain.com>
- <http://www.stratos-ad.com>
- <http://www.libsdl.org>
- <http://softwarelibre.uca.es/wikijuegos>
- <http://www.agserrano.com/publi.html>