DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Desarrollo de videojuegos

© Fernando Berzal, berzal@acm.org

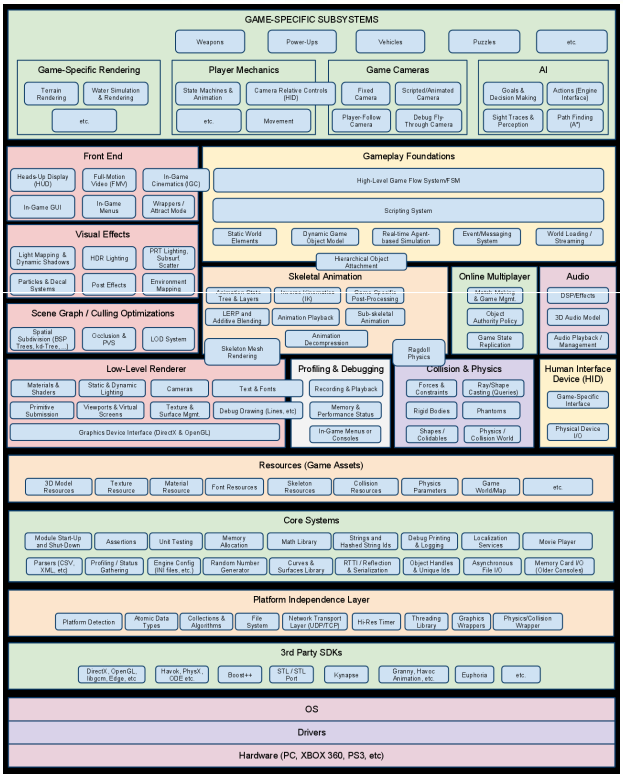
Game Engines

Motores de videojuegos [game engines]
http://en.wikipedia.org/wiki/Game_engine

Sistemas diseñados
para la creación y desarrollo de videojuegos

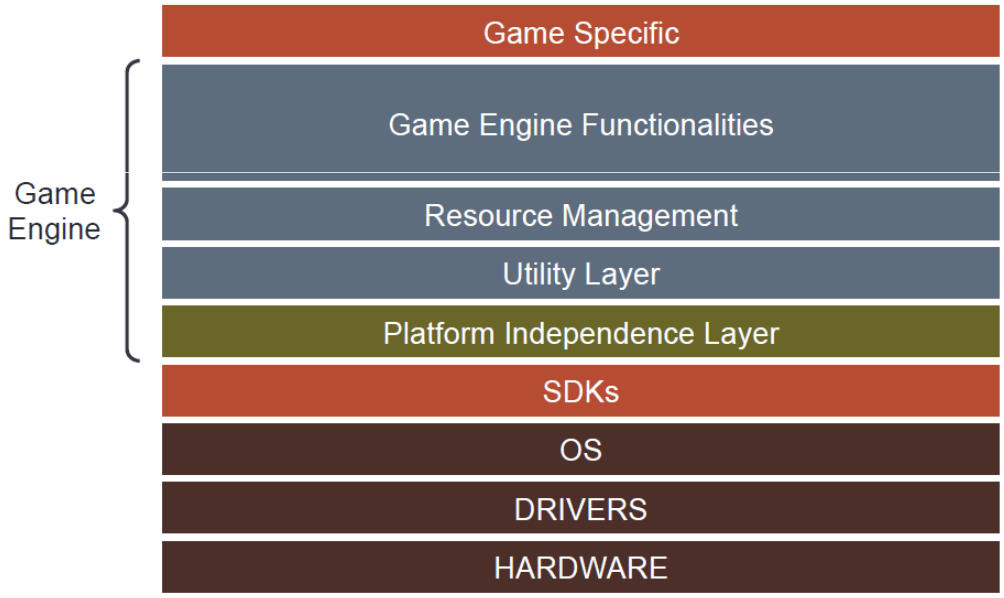
- La misma infraestructura se puede reutilizar para distintos juegos (del mismo género).
- La misma implementación de un juego se puede portar a distintas plataformas (PC/móvil/consola).

Ejemplo



Arquitectura

Diseño basado en capas...



Arquitectura

Platform Independence Layer

SDL Simple Directmedia Layer
http://www.libsdl.org/

- C/C++/C#/Python
- Licencia zlib
- Gráficos, sonido, acceso al hardware, redes…

Arquitectura

Utility Layer

Funcionalidad no ofrecida por los SDKs pero que es necesaria para la implementación de videojuegos:

- Gestión de memoria (la reserva estándar de memoria puede no ser suficientemente eficiente para un videojuego, de ahí que se utilicen implementaciones a medida).
- Algoritmos y estructuras de datos (geometría, generación de números aleatorios, ficheros de configuración…)

Arquitectura

Resource Management Layer

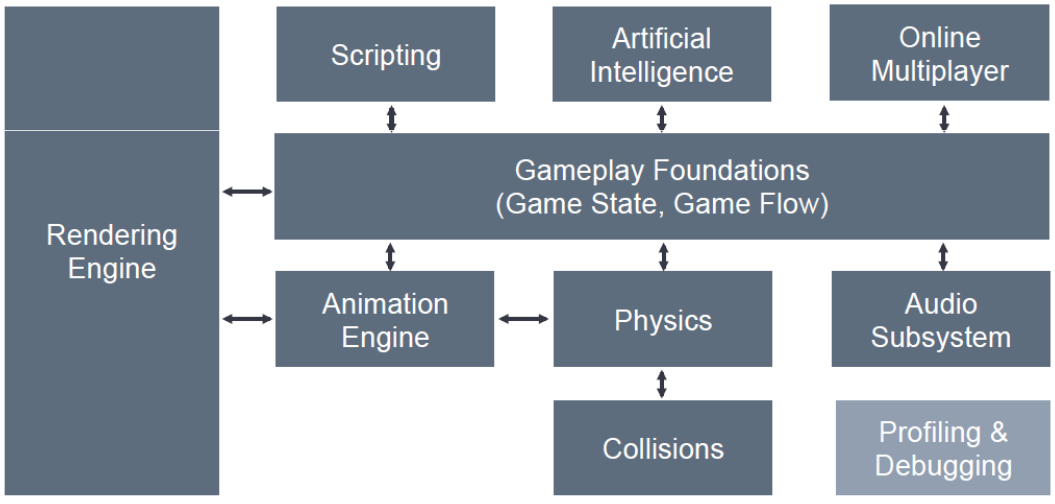
Gestión de recursos [assets]:

- Modelos 3D: meshes, texturas, skeletons, animaciones.
- Modelos 2D: sprites [bitmaps], skeletons, animaciones.
- Mapas
- Sonidos
- Fuentes (tipos de letra)
- …

Arquitectura

Game Engine Functionalities

Capa más compleja, compuesta por múltiples subsistemas.



El estado del videojuego

Conjunto de estructuras de datos que representan el estado actual del videojuego, p.ej.

- Estado del mapa (parte dinámica).
- Estado físico de los objetos.
- Estado de los personajes.
- Estado del jugador (inventario, estadísticas…)

El motor de renderizado [rendering engine] visualiza el estado del videojuego en pantalle y otros módulos se encargan de actualizarlo (física, IA…).

El estado del videojuego

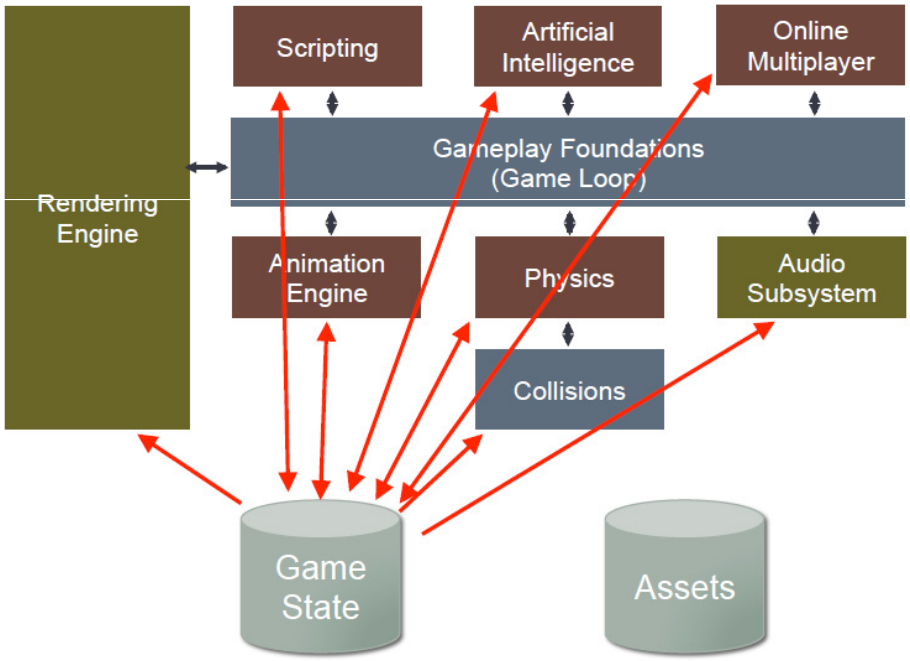
No forman parte del estado del juego: ranking de mejores puntuaciones, configuración del teclado…

Delimitar correctamente el estado del videojuego simplifica determinadas tareas,

p.ej. guardar/reanudar una partida consiste en serializar el objeto que representa el estado del juego y guardarlo en lugar seguro.

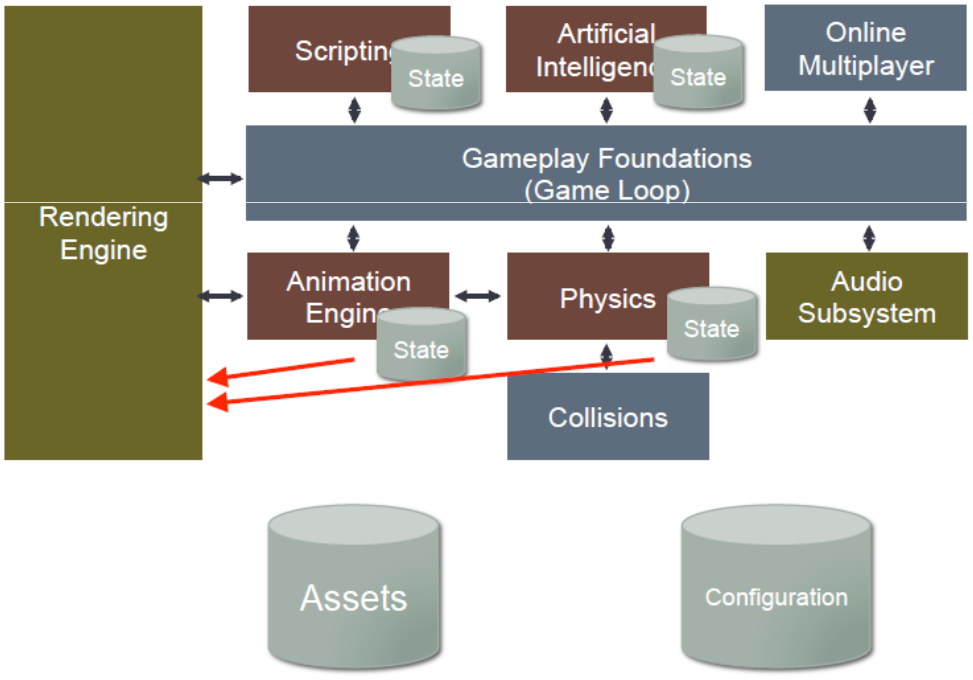
El estado del videojuego

Diseño deseable



El estado del videojuego

Diseño problemático



The "game loop"

- El bucle principal de un videojuego, responsable de controlar el avance del tiempo e integrar todos los módulos del motor de videojuegos.
- Versión simplificada:

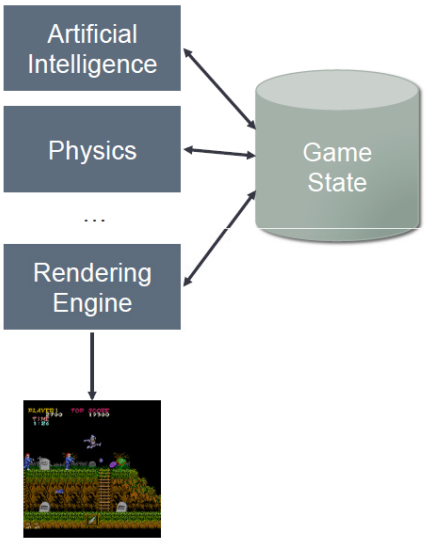
repetir 50 veces por segundo:
obtener entrada del usuario
actualizar el estado del juego
renderizar gráficos

The "game loop"

```
G = new GameEngine();

while (!quit) {
    I = getInput(&quit);
    G.updateGameState(I);
    G.render();
    FPScontrol();
}

delete G;
```



The "game loop"

```
G = new GameEngine();

while (!quit) {
    pollForOSMessages();
    I = getInput(&quit);
    if (timeForUpdatingAI())
        G.updateAI(&I);
    if (timeForUpdatingPhysics())
        G.updatePhysics(I);
    updateStatistics();
    if (timeForRendering())
        G.render();
    FPScontrol();
}
delete G;
```


The "game loop"

Lo que no hay que hacer…

```
G = new GameEngine();

while (!quit) {
    I = getInput(&quit);
    G.updateGameState(I);
    G.render();
    delay(10ms);
}

delete G;
```

The "game loop" (SDL, 50fps)

```
int time = SDL_GetTicks();
int REDRAWING_PERIOD = 20; // ms
bool need_to_redraw = true;

while (!quit) {
    int current_time=SDL_GetTicks();
    if (current_time-time>=REDRAWING_PERIOD) {
        time+=REDRAWING_PERIOD;
        keyboard->cycle();
        if (!game->cycle(k)) quit=true;
        need_to_redraw=true;
    }
    if (need_to_redraw) {
        game->draw();
        need_to_redraw=false;
    }
    SDL_Delay(1);   // Para dar tiempo de CPU a otros procesos
}
```

The "game loop" (SDL, 50fps)

```
int time = SDL_GetTicks();
int REDRAWING_PERIOD = 20; // ms
int MAX_FRAME_SKIP = 10;
bool need_to_redraw = true;

while (!quit) {
    int current_time=SDL_GetTicks();
    int frames = 0;
    if (current_time-time>=REDRAWING_PERIOD
       && frames<MAX_FRAME_SKIP)  {
        time+=REDRAWING_PERIOD;
        keyboard->cycle();
        if (!game->cycle(k)) quit=true;
        need_to_redraw=true;
        current_time=SDL_GetTicks();
        frames++;
    }
    if (time < act_time) time = act_time;
    ....
```

Si el ordenador va demasiado lento para el FPS deseado, realizamos más de una actualización por frame.

The "game loop" (SDL, 50fps)

```
...
while (!quit) {

    while ( SDL_PollEvent( &event ) ) {
        switch ( event.type ) {
            case SDL_KEYDOWN:
                if (event.key.keysym.sym == SDLK_F12)  quit = true;
                break;
            case SDL_MOUSEBUTTONDOWN:
                game->MouseClick(event.button.x,event.button.y);
                break;
            case SDL_QUIT:
                quit = true;
                break;
        }
    }
    ...
}
```

Manejo de eventos del sistema operativo

The "game loop" (t. continuo)

Tiempo continuo (en vez de FPS fijo): Se actualiza el estado del juego con tanta frecuencia como sea posible.

```
int lastTimeRendered = getTime();
while(!quit) {
    I = getInput(&quit);
    int time = getTime();
    int delta = time – lastTimeRendered;
    int lastTimeRendered = time;
    G.updateGameState(I, delta);
    G.render();
    FPScontrol();
}
```

100% CPU

The "game loop" (en red)

Servidor

```
While(!quit) {
   I = getInputfromClient(&quit);
   G.updateGameState(I);
   FPScontrol();
}
```

Physics ... Artificial Intelligence

Game State

Cliente

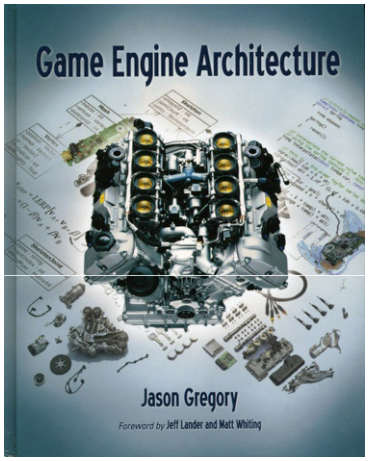
```
While(!quit) {
   I = getInput(&quit);
   sendInputToServer(I);
   getUpdateFromServer();
   G.render();
   FPScontrol();
}
```

Game State Copy

Rendering Engine

Referencias

- Jason Gregory:
  "Game Engine Architecture"
  AK Peters / CRC Press, 2009
  ISBN 1568814135
  http://www.gameenginebook.com/



- Santiago Ontañón:
  "Game Engine Programming"
  Drexel University, CS 480/680
  https://www.cs.drexel.edu/~santi/teaching/2013/CS480-680/intro.html