Arquitectura de videojuegos

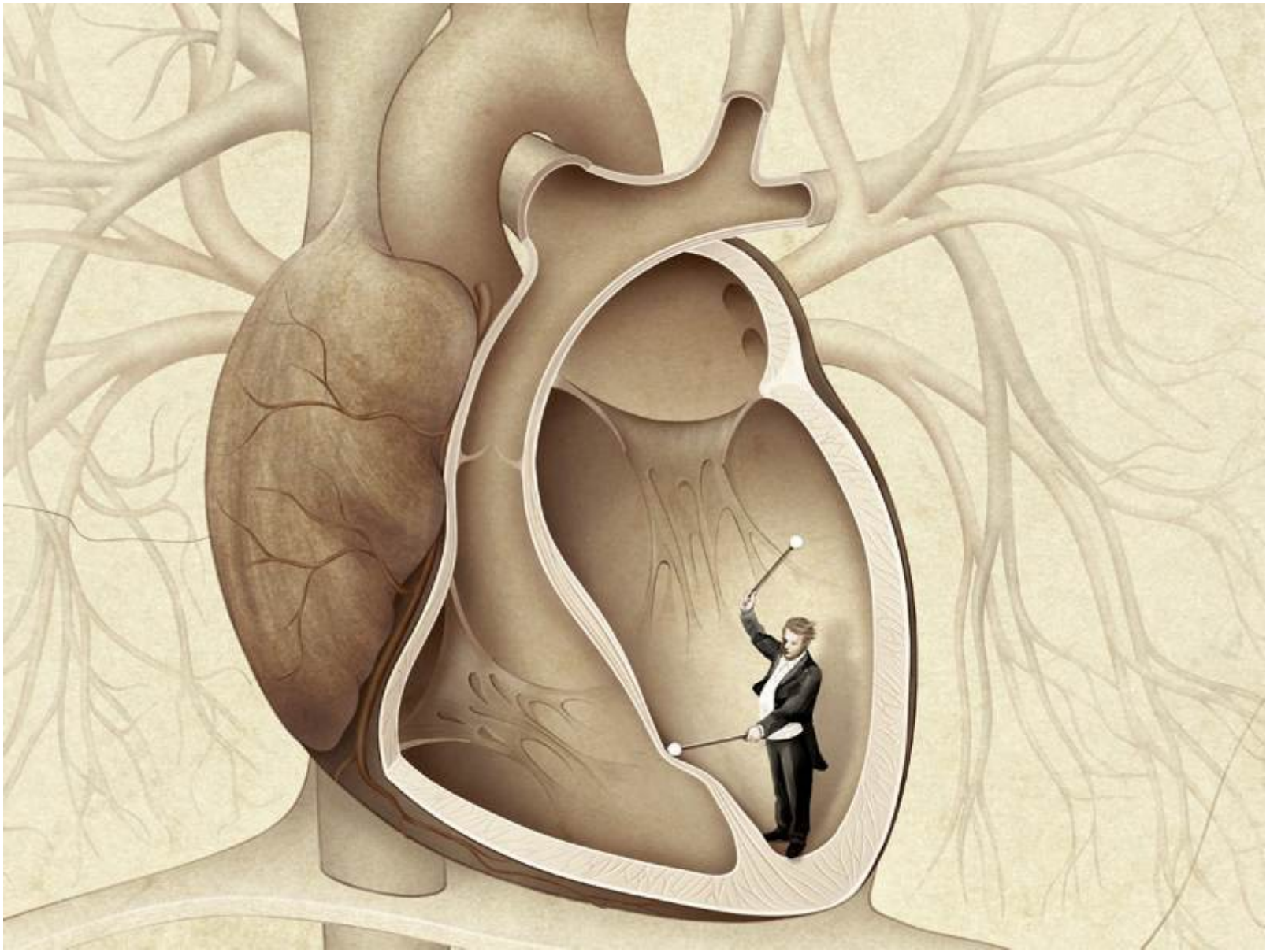
Máster en Programación de videojuegos

Javier Arévalo

2º Trimestre
Curso 2013-2014

Arquitectura de videojuegos

Temporizacion



Temporizacion

- Control del tiempo, ritmo y velocidad durante la ejecución del juego
- Tipos de Tiempo
- Pasos de tiempo fijo y variable
- Pausas en la linea de tiempo
- Otros problemas y soluciones

Bucle Principal

```
CurrentState = null
WantedState = InitialLoad

while (true):
  if (CurrentState != WantedState):
    CurrentState.Deactivate().Unload().Destroy()
    if (WantedState == null):
      break
    CurrentState = WantedState
    CurrentState.Initialize().Load().Activate()
    WantedState = null

  CurrentState.Run()
```

A que velocidad se ejecuta el juego?

Bucle principal

- Por defecto, se ejecuta tan rápido como permita el hardware
- El hardware aumenta de prestaciones
  - Aunque ya no tan rápido como antes
- El ritmo o frecuencia de ejecución del estado de juego seria mayor con mejor hardware
- Depende de la implementacion, esto es un problema

Mal...

```
while (true):
  CurrentState.Run()
...
function Run():
  PlayerX = PlayerX + 1
```

Pero el problema de fondo es este: no hay unidades de referencia respecto al “tiempo real”

```
while (true):
  CurrentState.Run()
...
function Run():
  PlayerY = PlayerY + PlayerVelocityY
  VelocityY = VelocityY + 9.8
```

Tiempo real

- El tiempo que transcurre desde el punto de vista del jugador
- Necesitamos alguna forma de medirlo dentro del código
- Y así poder ajustar nuestros cálculos

Tiempo real

- Métodos clásicos:
  - Indirectamente a través de características del hardware
  - V-blank: detectar cada refresco de imagen en la pantalla, sabiendo que va a 50Hz o 60Hz
  - Usando contadores de reloj en la CPU
  - Usando temporizadores programables

Tiempo Real

- V-Blank
  - El mas popular en consolas clásicas
  - El juego hace un Run() cada uno o dos V-Blank
  - Constantes de juego adaptadas a esa frecuencia
  - Juegos mas lentos en Europa (50Hz) que USA o Japon (60Hz) – nadie cambia las constantes!

```
while (true):
  CurrentState.Run()
...
function Run():
  PlayerY = PlayerY + PlayerVelocityY
  VelocityY = VelocityY + 0.196
```

Tiempo Real

- Contadores de ciclos de CPU
  - P.ej. RDTSC en Intel x86
  - A que MHz o GHz va la CPU?
  - Y cuando esa frecuencia cambia (power saving)?
  - Y si tenemos múltiples cores, cada uno lleva una cuenta diferente?
- Temporizadores programables
  - Margen de error tremendo

Tiempo Real

- A día de hoy asumimos el problema resuelto
  - Cada plataforma es diferente
  - Pero en general podemos medir el tiempo transcurrido con precisión de 1 milisegundo
  - Llamemoslo **`GetTime()`**
- Ya conocemos el tiempo real en el juego, como lo aplicamos en nuestra logica?

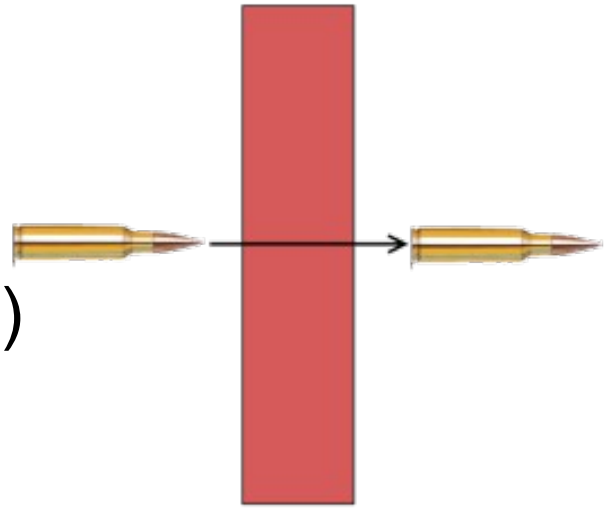
Pasos de tiempo variable

- Funcion GetTime() devuelve el tiempo en segundos desde un momento fijo en el pasado
  - Epoch, arranque del sistema, arranque del juego...
  - En coma flotante con precisión de milisegundos

```
previousTime = GetTime()
while (true):
  currentTime  = GetTime()
  elapsed = currentTime - previousTime
  previousTime = currentTime
  CurrentState.Run(elapsed)
...
function Run(elapsed):
  PlayerY = PlayerY + PlayerVelocityY*elapsed
  VelocityY = VelocityY + 9.8*elapsed
```

Pasos de tiempo variable

- Hay que multiplicar por 'elapsed' todas las computaciones que impliquen tiempo
- Al ser muchas sumas de valores pequeños y variables, podemos acumular error
- Se pueden producir grandes variaciones en el valor de 'elapsed'
  - Desde cero hasta... horas!
  - Afecta a colisiones, etc
- Lógica que se ejecuta una vez por Run()
  - No se ajusta por el 'elapsed'



Pasos de tiempo fijo

- Cuantizamos el paso del tiempo antes de llamar a Run()
  - En cada ciclo de juego ejecutamos varias veces el Run()
  - La lógica de Run() siempre recibe un 'elapsed' fijo

```
fixedTick = 1000/60
previousTime = GetTime()
while (true):
  currentTime  = GetTime()
  elapsed = currentTime - previousTime
  previousTime = currentTime
  while (elapsed >= fixedTick):
    CurrentState.Run(fixedTick)
    elapsed = elapsed - fixedTick
```

Pasos de tiempo fijo

- Errores tipicos
  - Al dejar de llamar a Run(), todavia queda un resto en 'elapsed'.
  - Acumularlo, o no descartarlo

```
fixedTick = 1000/60
previousTime = GetTime()
while (true):
  currentTime  = GetTime()
  elapsed = currentTime - previousTime
  previousTime = currentTime
  while (elapsed >= fixedTick):
    CurrentState.Run(fixedTick)
    elapsed = elapsed - fixedTick
```

```
fixedTick = 1000/60
previousTime = GetTime()
elapsed = 0
while (true):
  currentTime  = GetTime()
  elapsed += currentTime - previousTime
  previousTime = currentTime
  while (elapsed >= fixedTick):
    CurrentState.Run(fixedTick)
    elapsed = elapsed - fixedTick
```

```
fixedTick = 1000/60
previousTime = GetTime()
while (true):
  currentTime  = GetTime()
  elapsed = currentTime - previousTime
  while (elapsed >= fixedTick):
    CurrentState.Run(fixedTick)
    elapsed = elapsed - fixedTick
    previousTime += currentTime
```

Espiral de la muerte

http://aidan8500.deviantart.com/art/It-s-a-maelstrom-205582697

Pasos de tiempo fijo

- Espiral de la muerte
  - El coste de ejecutar un paso de Run() es mas o menos constante, al margen del valor de 'elapsed'
  - Si la maquina es lenta, en pasos fijos ejecutamos Run() con mas frecuencia que con variables
  - Pero si lo ejecutamos mas veces, nos cuesta mas tiempo de proceso
  - Y si ejecutar todos esos Run() nos cuesta mas tiempo que el que simulamos?

Problemas de temporizacion

- Detención del tiempo:
  - Por pararse en un breakpoint al depurar
  - Por una operación de carga o de calculo que tarda mucho en realizarse
- Aberraciones:
  - Tiempos 'elapsed' de cero
  - O peor aun, negativos

Problemas de temporizacion

- Soluciones:
  - Limitar el numero de pasos de tiempo fijo
  - Descartar tiempos de 'elapsed' por encima de X
  - Ofrecer una función para descartar todo el tiempo acumulado

Lineas de tiempo

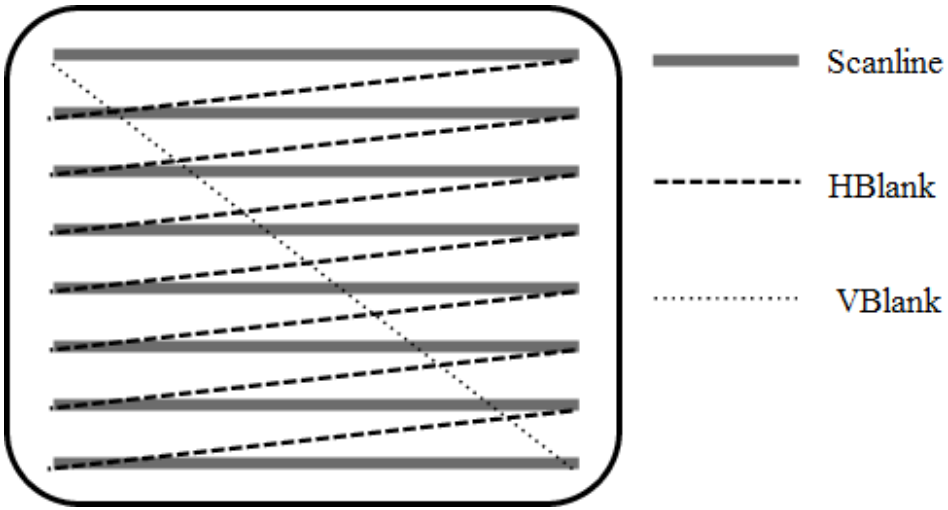
- Cuantos contadores de tiempo tenemos?
  - Tiempo real
  - Tiempo medido para el Run()
  - Tiempo según el hardware gráfico y la pantalla
    - Y el Audio
  - Tiempo percibido por el jugador
  - Tiempo en la lógica del juego
  - Tiempo de otras maquinas en red
  - Tiempo que pasa mientras el juego no corre

Lineas de tiempo

http://forums.watchuseek.com/f71/sotc-how-many-watches-can-i-fit-my-wrist-589205.html

Tiempo en el display

- El tiempo en el display esta cuantizado
- Esperar a VBlank
- O hacer triple buffering
- O aceptar *tearing*



- Si nuestros tiempos de lógica no van asociados a estos intervalos, habrá saltos

Tiempo percibido

- El jugador percibe el tiempo según su estado de inmersión en el juego
  - Acción: estimula la percepción rápida y precisa
  - Si introducimos retrasos entre sus acciones, la reacción del juego, y la visualización del resultado, la experiencia de juego sufre
- Ante gran densidad de información, puede no percibir ralentizaciones

Tiempo de logica

- Elementos del juego que cambian con el tiempo
  - Macro: ciclos de dia/noche, regeneración de objetos y enemigos
  - Micro: el movimiento de cada objeto, esperas entre disparos, etc
- Pausas
  - Pausa logica: el tiempo de logica no transcurre
  - Pausa activa: se permite dar ordenes
    - Implica que el tiempo si transcurre para el interfaz

Tiempo de red

- Lineas de tiempo activas en cada maquina
- Necesidad de sincronizacion: inicio, corregir desviaciones y errores acumulados, etc
- Uso de un servidor maestro
- Normalmente ignoramos las lineas de tiempo visual y nos centramos en lógica e input
- Los *cheats* son especialmente peligrosos en juegos multijugador

Tiempo fuera del juego

- Importante en juegos de gestión de tiempo
  - Farmville, Clash of Clans, Tiny Tower, etc
  - Completar tareas y re/generar recursos
- También presente en MMOs:
  - Acumulación de tiempo de descanso
  - Eliminación de penalizaciones
  - Subastas, eventos
- Contar tiempo transcurrido al iniciar el juego
  - O mejor, guardar tiempos de inicio y comparar con tiempo actual

Vuestro turno!

Preguntas?