Inteligencia Artificial para Videojuegos

Cuestiones avanzadas

Gestión de la ejecución

Motivación

- Hablemos de lo mucho que se puede aprender de desarrolladores veteranos
  - Estudios que mantienen el mismo núcleo del equipo (los *senior* no se queman), buenos programadores que se ocupan de las *herramientas* y buenos técnicos que se ocupan de soportar los *procesos*
  - Ej. Guerrilla Games con su motor *Decima* y su tecnología de IA bien integrada (percepción, grupos de agentes, planificación HTN, etc.)

Motivación

- Cuando llevamos la *teoría* al mundo real debemos estar alerta ante la falacia de la complejidad
  - Lo *simple* muchas veces... *¡sale bien!* (Ej. Pac-Man)
  - Y lo *complicado* a veces... sale mal ☹ (Ej. Herdy Gerdy)

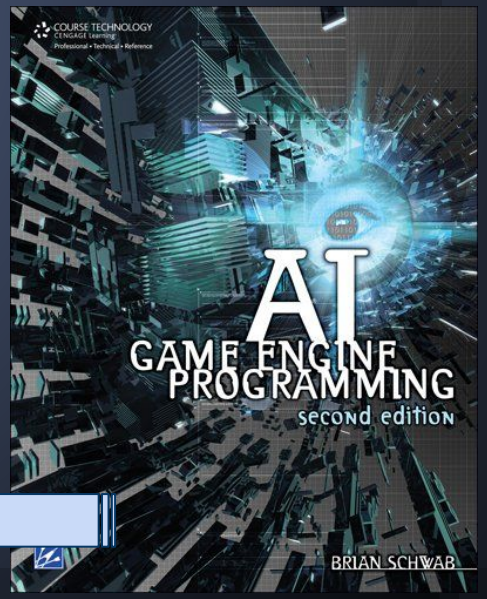
"Tenemos suficiente experiencia como para saber que [estas técnicas] son a menudo innecesarias: habitualmente el mismo efecto se puede conseguir *mejor*, *más rápido* y *con más control* utilizando un enfoque más simple"
"Saber cuando complicarse y cuando mantenerlo simple es lo más difícil del arte del programador de la IA de un juego"
Ian Millington & John Funge (2009)

Motivación

Ian Millington

- Como *cuestiones avanzadas* veremos consejos *de implementación* para las técnicas de IA para videojuegos que vimos
- Concretamente empezaremos hablando de las tecnologías de soporte* para estos sistemas (≃ *motores de IA*)

* Que muchas veces damos por supuestas por estar incluidas en juegos e incluso IDEs actuales



Puntos clave

- Hitos históricos
- Sistema multi-agente
- Arquitectura de agentes
- Velocidad y capacidad
- Motor de IA
- Gestión de la ejecución
  - Programaciones
  - Algoritmos siempre listos
  - Niveles de detalle

Hitos históricos

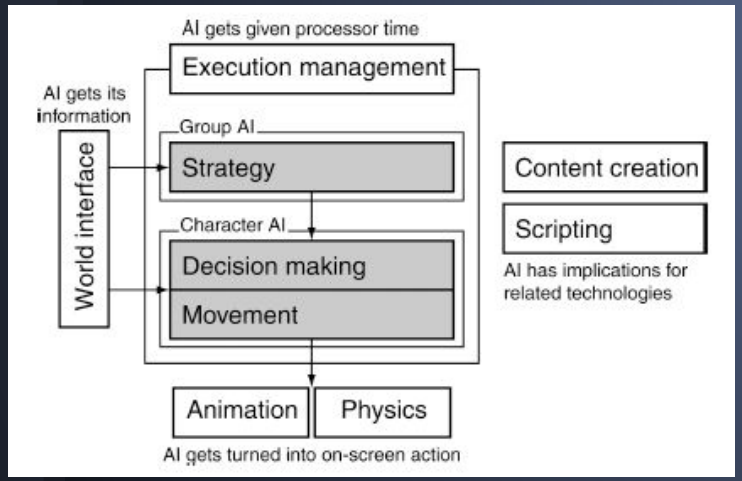
- En los últimos años van surgiendo *nuevas propuestas* de IA para videojuegos
  - Bot Colony (2014)
    - Se basa íntegramente en NPCs que reconocen órdenes por voz, lo están "rehaciendo" en UE4
  - Alien Isolation (2014)
    - Los comportamientos de la IA se van desbloqueando poco a poco, para dar sensación de aprendizaje
  - The Last Guardian (2016)
    - Busca credibilidad en Trico, un animal como compañero

Hitos históricos

- Hay mucha más investigación en juegos FPS y RTS (Ej. Competición anual de IAs para StarCraft y muchos otros títulos)
  - Técnicas de simulación militar en RTS (Ej. Full Spectrum Warrior comenzó como simulador...)
  - Análisis de datos en juegos deportivos y de conducción (Ej. Calcular en tiempo real la manera más rápida de recorrer un circuito es útil para los equipos real de Fórmula 1 también)
  - Sistemas de gestión distintos a los árboles de conversación en RPGs (Ej. Façade o Versu), incluso *directores de juego* (Ej. Left 4 Dead o Earthfall)

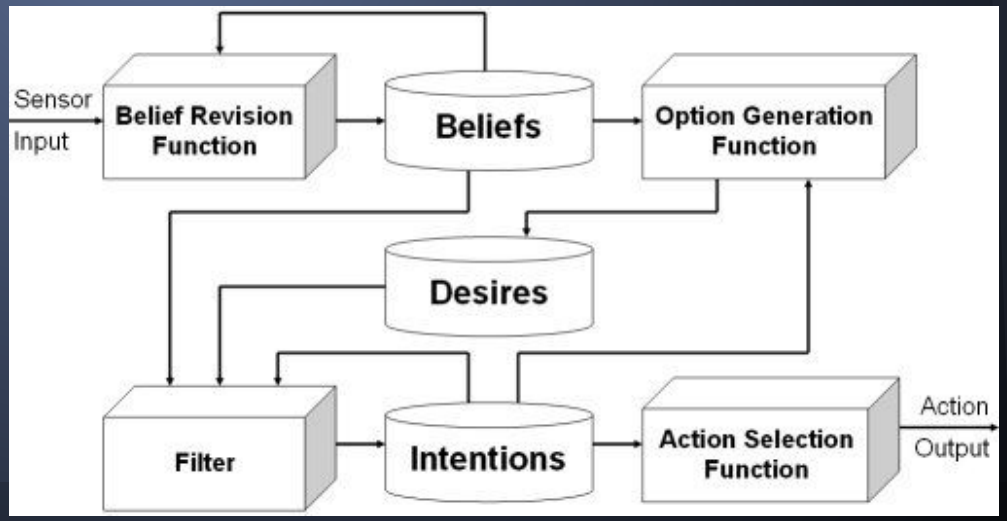
Sistemas multi-agente

- El original de Millington y Funge (2009) es un modelo… pero hay otros como las arquitectura cognitivas BDI (1987)



BELIEF-DESIRE-INTENTION (BDI)

* Michael Bratman propuso modelar la mente del agente con *creencias*, *deseos* e *intenciones*



Sistemas multi-agente

- *Casi* todo NPC sigue el enfoque de Agente
  - Pero la industria evita esta *terminología académica*
- La industria se centra en *implementar* sistemas multi-agente
  - Vimos que la solución general son los algoritmos: *entender el problema*, esbozar una *solución* (*diagramas* y *pseudocódigo*), diseñar y tomar decisiones de implementación, medir el *rendimiento* y reconocer las *debilidades* que pueda tener cada técnica

Sistemas multi-agente

- Pero los algoritmos son sólo parte de la IA, hace mucha falta buena infraestructura
  - El movimiento hay que convertirlo en *acciones del juego* (con sus animaciones y simulación física)
  - Hace falta la *percepción*, tomar datos del juego con los que la IA podrá decidir… no sólo lo que el personaje ve u oye, sino *todo lo que hace el programador de IA por conectar con el mundo* (y tener que depurar esta interfaz)
  - Y por supuesto, controlar cuánto *consumo del procesador y la memoria* tienen los sistemas de IA

Velocidad y capacidad

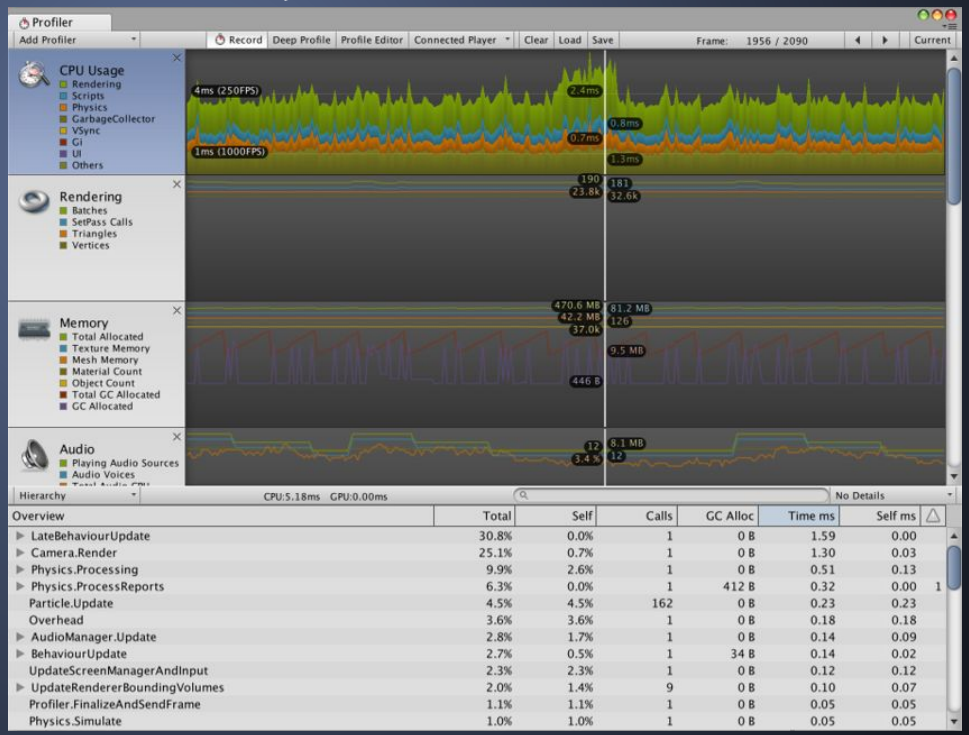
- El desafío radica en *aprovechar el hardware*
  - Podemos tener gráficos distintos en cada máquina, ¡pero no *IA distinta* en cada máquina!
    - Ej. Quejas según *fps* en PayDay 2
  - Presupuesto en IA de videojuegos = velocidad de procesamiento & capacidad de almacenamiento
- Procesamiento, repartir los *ms* de CPU
  - Con los gráficos en GPU, ahora la IA dispone del 5-25% del *frame* (a veces llega a usar más del 50%)
- Memoria, concentrar toda la *info* en un sitio
  - Los algoritmos son más eficaces si todos los datos necesarios los tienen disponibles a la vez en *caché*

Velocidad y capacidad

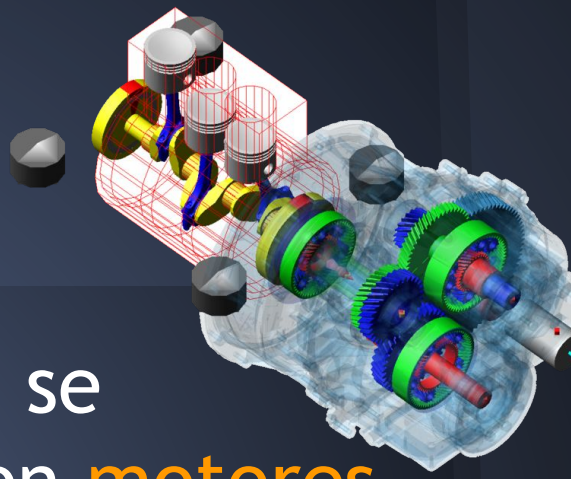
PROFILER

- Optimizar el código usando el optimizador para ver en qué áreas o *frames* concretos aparecen picos
- PC
  - La IA se diseña pensando en la *gama baja*, salvo decoración

- Consolas
  - Presupuesto fijo (suele ser *bajo*), pero peor depuración por retardo de la plataforma



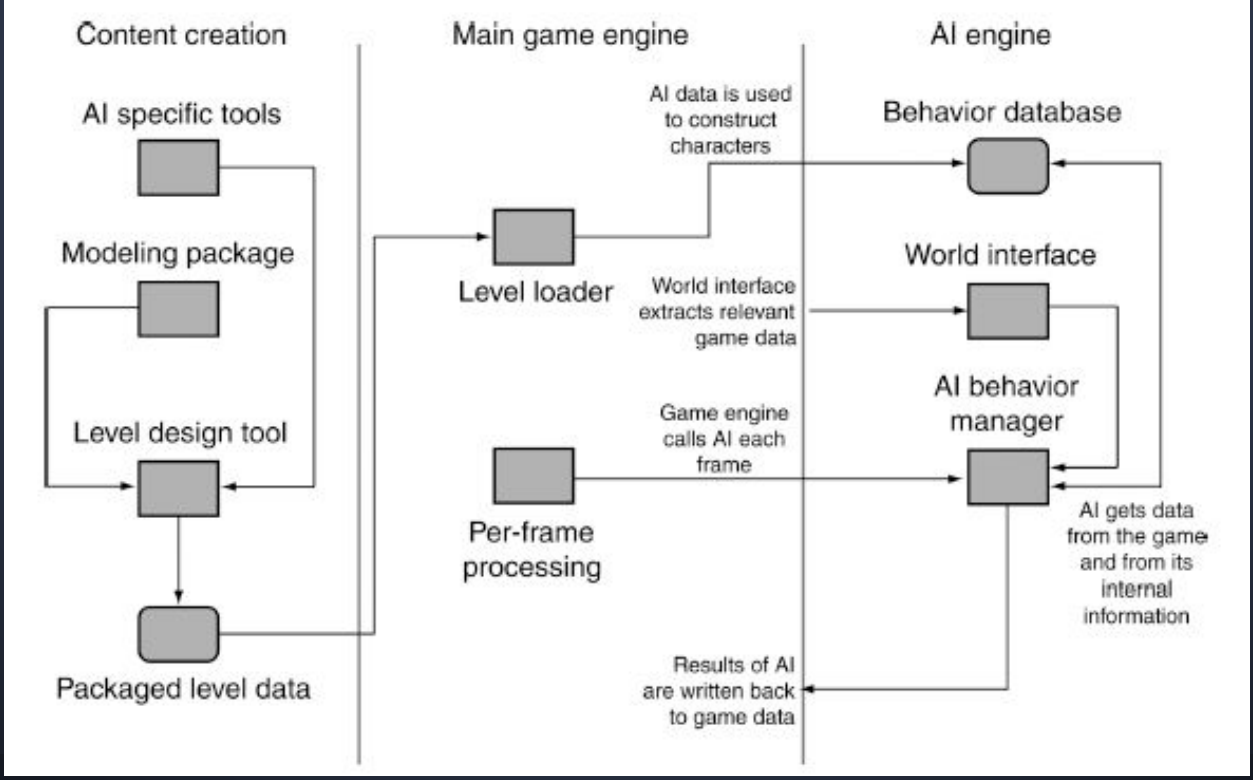
Motor de IA



- En los 90 el videojuego (y su IA) se construía desde cero... luego con motores
  - Un conjunto de herramientas reutilizables, combinables/encadenables y aplicables a varios géneros (según *dinámicas* de más alto nivel)
- Ofrecen una infraestructura básica
  - Mecanismo general para gestionar comportamientos (movimiento, navegación, decisión, etc.) con estructura estándar (cierto interfaz)
  - Interfaz para recibir datos del entorno
  - Interfaz para actuar sobre el entorno y los NPCs (controladores de *movimiento* y *animación*)

Motor de IA

- Este es el esquema que *integra* el motor en el proceso de desarrollo de un videojuego



Motor de IA

- En un desarrollo profesional primero se *diseña e implementa* el motor de IA y luego, sobre él, van las técnicas concretas
  - Su diseño permite desarrollar más eficazmente, reutilizando y depurando comportamientos mejor
- El motor incorpora varias características

  - Gestión de la ejecución, se puede controlar el *tiempo* dedicado a cada comportamiento

  - Interfaz del mundo, un sistema central de *paso de mensajes* y acciones según un *formato estándar*

  - Herramientas de autoría, para que el diseñador pueda configurar y combinar los comportamientos

Participación

tiny.cc/IAV

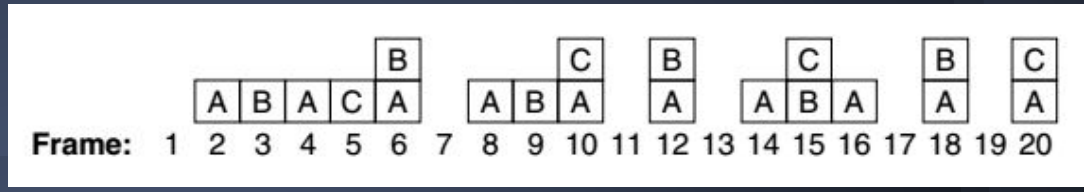
- ¿Qué ofrece un motor de IA?
  A. Optimización de renderizado de *frames*
  B. Algoritmos y tipos de datos básicos
  C. Interfaz entre el mundo y las herramientas
  D. Control sobre el tiempo dedicado a cada técnica
- Desarrolla tu respuesta (en texto libre)

Gestión de la ejecución

- Es la primera de las características del motor de IA que da soporte a los agentes
- Saca provecho del tiempo de procesamiento usando tres técnicas fundamentales
  - Programaciones, repartir el tiempo disponible entre todos los comportamientos activos ese momento
  - Algoritmos siempre listos, dar soluciones enseguida y utilizar los *fotogramas* sobrantes para pulirlas
  - Niveles de detalle, emplear más esfuerzo en los comportamientos que *ahora* son más relevantes

Programaciones

SCHEDULING

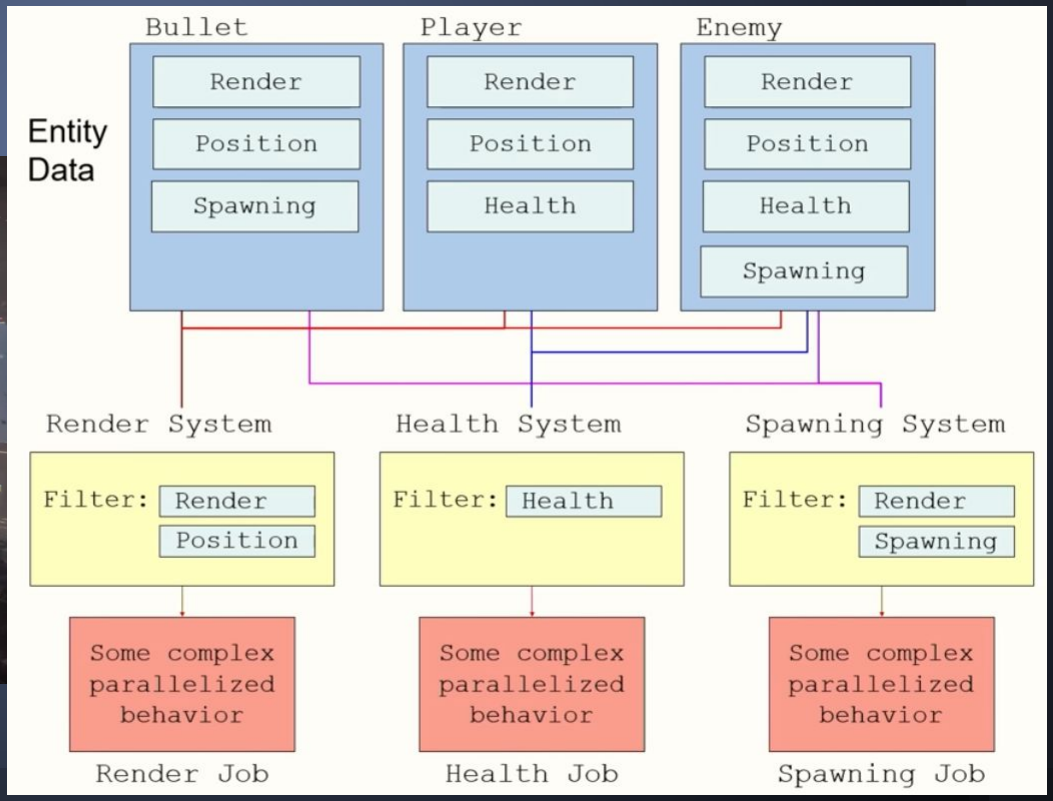


- El gestor de la ejecución *automatiza* las programaciones del procesador (decide qué *frames* puede usar cada comportamiento)
  - Organiza un conjunto de comportamientos activos, cada uno con su duración y su frecuencia
- Se utilizan diversas técnicas para ello
  - Se pueden reducir frecuencias para que sean *primas relativas*, aplicarlas cierto desplazamiento (o *fase*)...o usar algoritmos interrumpibles usando hilos, corrutinas y similares
  - Las técnicas más avanzadas incluyen jerarquizar y priorizar los comportamientos

Programaciones

- Para optimizar el uso de la memoria cobra fuerza el paradigma ECS ENTITY-COMPONENT-SYSTEM

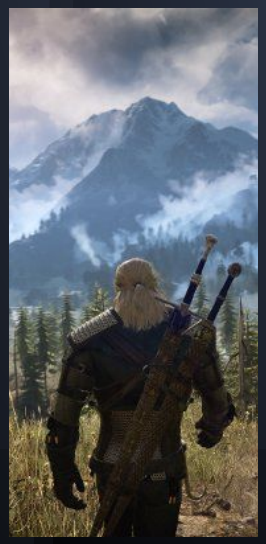
* MegaCity demo (Unity, 2019)



Algoritmos siempre listos

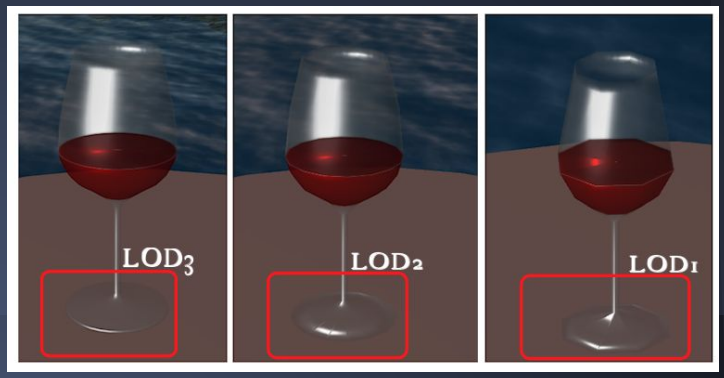
ANYTIME ALGORITHMS

- A pesar de usar *programaciones*, si un algoritmo tarda demasiado en dar una solución, *el NPC permanecerá paralizado*
  - Existen algoritmos que *reciben la cantidad de tiempo de que disponen* y se ajustan a ella
  - Aún mejor: existen los algoritmos siempre listos, algoritmos *interrumpibles* que garantizan tener siempre una *solución válida* (aunque sea una aproximación burda) y *pulirla* si les das más tiempo
    - Ej. Un personaje navega hacia otro punto del mundo, aunque su IA *no ha terminado* de elaborar *el camino óptimo* (por ahora vale con unos pocos nodos, ya seguirá añadiendo...)



Niveles de detalle

LEVELS OF DETAIL (LOD)



- Los algoritmos que admiten distintos niveles de detalle son habituales en *gráficos*
  - La distancia suele ser el principal motivo para ajustar el nivel de detalle, aunque *no* el único
- En IA puede usarse en las programaciones
  - Dar más tiempo a *los NPCs o comportamientos más relevantes* o con los que *el jugador es más sensible*
    - Depende de la jugabilidad, de la narrativa…
- …en los propios comportamientos
  - Algunos serán versiones "comprimidas" de otros
- … y hasta en grupos enteros de NPCs
  - Ej. Simulando población de una región

Resumen

- La IA para videojuegos básicamente consiste en crear sistemas multi-agente
- Exprime la velocidad de procesamiento y la capacidad de almacenamiento del HW
- El motor de IA unifica la gestión de la ejecución, la interfaz con el mundo y las herramientas de autoría
- La gestión de la ejecución optimiza el tiempo con programaciones, algoritmos siempre listos y usando niveles de detalle

Más información

- Millington, I.: Artificial Intelligence for Games. CRC Press, 3rd Edition (2019)

Críticas, dudas, sugerencias...

Federico Peinado (2019-2021)

www.federicopeinado.es