

---

# Introducción a la Programación Lúdica

Tema 8. Inteligencia Artificial en  
videojuegos

---

# Esquema

---

- Elementos de Inteligencia Artificial en videojuegos
- Técnicas de Inteligencia Artificial
  - Técnicas Clásicas
  - Técnicas Avanzadas

# Esquema

---

- Elementos de Inteligencia Artificial en videojuegos
- Técnicas de Inteligencia Artificial
  - Técnicas Clásicas
  - Técnicas Avanzadas

# Elementos de IA en videojuegos

---

- Inteligencia Artificial:
  - Disciplina que se ocupa de la creación de programas de ordenador que simulan la actuación y el pensamiento humano, así como el comportamiento racional [Russell, 1995]
- Objetivo en videojuegos. Simular automáticamente del **comportamiento** de un jugador:
  - Inteligente
    - Patrón óptimo para conseguir un objetivo
  - Humano
    - Predecible / No predecible
    - Cooperación
    - Sorpresa
- Gran importancia.

# Elementos de IA en videojuegos

---

- Restricciones:
  - Diversión:
    - Equilibrio entre victorias y derrotas
    - Uso de técnicas *poco sofisticadas*
    - Puede convenir “hacer trucos”
  - Desafío
- Propiedades:
  - Dependencia de contexto (Feigenbaum: “*knowledge is power*”)
  - Comportamiento emergente
  - Niveles de dificultad

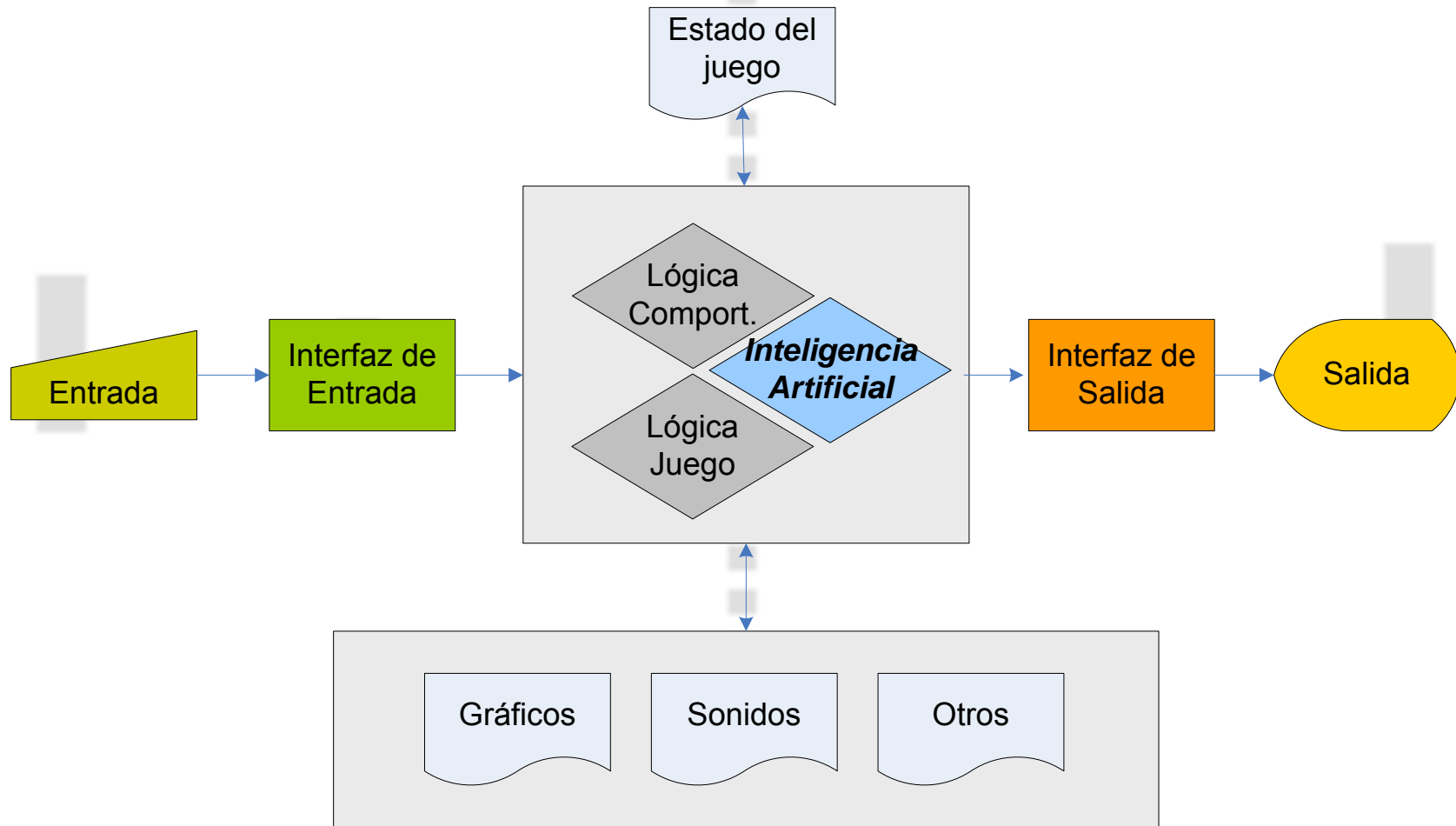
# Elementos de IA en videojuegos

---

- Tareas. Dependiendo de la complejidad del juego, se consideran desde tareas muy simples hasta otras muy complejas:
  - Atrapar enemigos (Pac-Man)
  - Detección de colisiones (Max Payne)
  - Búsqueda de caminos (Half Life)
  - Situación de cámaras (LoK Defiancé)
  - Toma de decisiones (Imperial Glory)
  - Aprendizaje (NHL 2K6)
  - Evolución de personajes (RPGs)
  - Comportamiento social (Sims)

# Elementos de IA en videojuegos

---



# Elementos de IA en videojuegos

---

- Áreas:
  - **Sistemas Expertos**
  - **Teoría de Autómatas**
  - Lógica Difusa
  - Vida Artificial
  - Sistemas Multi-Agente
  - Robótica
  - Sistemas de Planificación
  - Computación Evolutiva
  - Redes Neuronales
  - Modelos Gráficos Probabilísticos
  - Otras: Psicología Cognitiva, Simulación de Sistemas, etc.
- Técnicas:
  - **Métodos de búsqueda**
  - **Sistemas de producción**
  - **Árboles de decisión**
  - Optimización
  - Razonamiento basado en casos
  - Lógica de primer orden
  - Cálculo de predicados

# Elementos de IA en videojuegos

---

- Historia:
  - Primeros juegos (Pong, Pac-Man, Space Invaders, Donkey Kong)
  - Juegos de ajedrez (Chessmaster 2000)
  - Juegos de estrategia (Civilization)
  - Juegos de lucha (Street Fighter 2)
  - Juegos de estrategia en tiempo real (WarCraft 2, Age of Empires 2)
  - Juegos *God-alike* (Black & White)
  - Juegos FPS (Half Life, Unreal:Tournament)
  - Simuladores sociales (Sims)

# Esquema

---

- Elementos de Inteligencia Artificial en videojuegos
- **Técnicas de Inteligencia Artificial**
  - Técnicas Clásicas
  - Técnicas Avanzadas

# Técnicas clásicas

---

- Técnicas clásicas
  - **Chase and evade**
    - **Basics: Random motion, Patterns, Probabilistic Motion, Tracking**
    - **Advanced: Flocking, Movimiento basado en función potencial**
  - Búsqueda de caminos
    - Grafos
    - A\*
  - Planificación
    - Agentes dirigidos por metas
  - Simulación de sistemas
    - Patrones
    - Sistemas de reglas
    - Máquinas de estados finitas
  - Aprendizaje
    - Refuerzo
    - Deductivo
  - Inteligencia colectiva
    - Sistemas de agentes

# Técnicas clásicas

---

- *Chase and Evade* (Capturar y Huir):
  - Técnicas elementales:
    - Random motion: Movimiento aleatorio.
    - Patterns: Se sigue un patrón de movimiento (que puede incorporar aleatoriedad).
    - Probabilistic motion: Se selecciona el movimiento según cierta función de probabilidad.
    - Tracking: Encontrar caminos dentro del escenario que permitan a un objeto alcanzar su objetivo de forma adecuada:
      - Solución óptima: Kruskal, Dijkstra...
      - Solución satisfacible: Backtracking, Minimax, Poda  $\alpha$ - $\beta$ ...

# Técnicas clásicas

---

- *Chase and Evade* (Capturar y Huir):
  - Técnicas avanzadas:
    - Flocking: Movimiento de personajes agrupados.
      - Propiedades: cohesión, alineamiento, separación.
      - Implementación: patterns, probabilistic motion, tracking.
    - Basadas en función potencial: Una función potencial establece la atracción o repulsión entre dos elementos del juego según la distancia a la que se encuentren.
      - Propiedades: mecanismo genérico para diferentes tareas (chase & evade, evitar obstáculos, flocking, etc.).
      - Implementación: costoso definir y ejecutar función.

# Técnicas clásicas

---

- Técnicas clásicas
  - Chase and evade
    - Basics: Random motion, Patterns, Probabilistic Motion, Tracking
    - Advanced: Flocking, Movimiento basado en función potencial
  - **Búsqueda de caminos**
    - **Grafos**
    - **A\***
  - Simulación de sistemas
    - Patrones
    - Sistemas de reglas
    - Máquinas de estados finitas
  - Aprendizaje
    - Refuerzo
    - Deductivo
  - Inteligencia colectiva
    - Sistemas de agentes
  - Planificación
    - Agentes dirigidos por metas

# Técnicas clásicas

---

- Búsqueda de caminos:
  - Determinar la ruta que debe seguir un objeto desde una posición inicial hasta la posición deseada.
  - Problema habitual en la programación de videojuegos.
  - Uso de grafos (generalmente DAG)
  - <http://theory.stanford.edu/~amitp/GameProgramming/index.html>

# Técnicas clásicas

---

- Búsqueda de caminos:
  - Aproximaciones básicas:
    - Tracking sobre grafos:
      - Exploración: Búsqueda “Primero en Profundidad”, búsqueda “Primero en anchura”, Algoritmos “Greedy”...
      - Métodos basados en coste: Programación dinámica, Algoritmo de Dijkstra...
    - Evitar obstáculos:
      - Movimiento aleatorio
      - Seguimiento de paredes
      - Memorizado de mapa (“*migas de pan*”)
    - Seguimiento de caminos predefinidos
    - Movimiento a través de caminos pre-calculados

# Técnicas clásicas

---

- Búsqueda de caminos:
  - Algoritmo A\*:
    - Solución general para problemas de búsqueda en espacios de soluciones representados como grafos.
    - Ampliamente aplicado al problema de la búsqueda de caminos.
    - Uso de función de distancia acumulada  $g(n)$  (distancia desde el nodo inicial hasta  $n$ ).
    - Uso de función de distancia esperada  $h(n)$  (valora el coste estimado desde  $n$  hasta el nodo final).
    - $h(n)$  puede ser una función de distancia: Euclídea, Manhattan, etc.
    - Propiedades:
      - Si  $h(n) = 0$ , A\*  $\leftrightarrow$  Dijkstra
      - Si  $h(n) < \text{distancia}$ , A\* es óptimo (explora más nodos).
      - Si  $h(n) = \text{distancia}$ , A\* es óptimo y no expande nodos extra.
      - Si  $h(n) > \text{distancia}$ , A\* no es óptimo (explora menos nodos).
      - Si  $h(n) \gg g(n)$ , A\* se comporta como una búsqueda “primero en anchura”.

# Técnicas clásicas

---

```
O = {s} // Abiertos = {nodo de partida}
C = ∅ // Cerrados = vacío

mientras O ≠ ∅ // Mientras queden nodos en Abiertos

    n = mini∈O (f(i)) // n: nodo de mínimo coste en Abiertos

    si (n == objetivo)
        FIN: Camino completo
    si no
        C = C ∪ n // Añadir n a la lista de Cerrados
        N = adyacentes(n) // Obtener nodos adyacentes a n

        para cada i ∈ N // Para cada nodo i adyacente a n
            si i ∉ O ∪ C // Si i no ha sido visitado aún
                y i ≠ obstáculo // Si i no es un obstáculo
                    O = O ∪ i // Añadir i a la lista de abiertos
                    g(i) = g(n) + d(n, i) // Calcular distancia de i
                    f(i) = g(i) + h(i) // Calcular coste de i
```

Ejemplo: <http://www.gamedev.net/reference/articles/article2003.asp>

# Técnicas clásicas

---

- Técnicas clásicas
  - Chase and evade
    - Basics: Random motion, Patterns, Probabilistic Motion, Tracking
    - Advanced: Flocking, Movimiento basado en función potencial
  - Búsqueda de caminos
    - Grafos
    - A\*
  - **Simulación de sistemas**
    - **Patrones**
    - **Sistemas de reglas**
    - **Máquinas de estados finitas**
  - Aprendizaje
    - Refuerzo
    - Deductivo
  - Inteligencia colectiva
    - Sistemas de agentes
  - Planificación
    - Agentes dirigidos por metas

# Técnicas clásicas

---

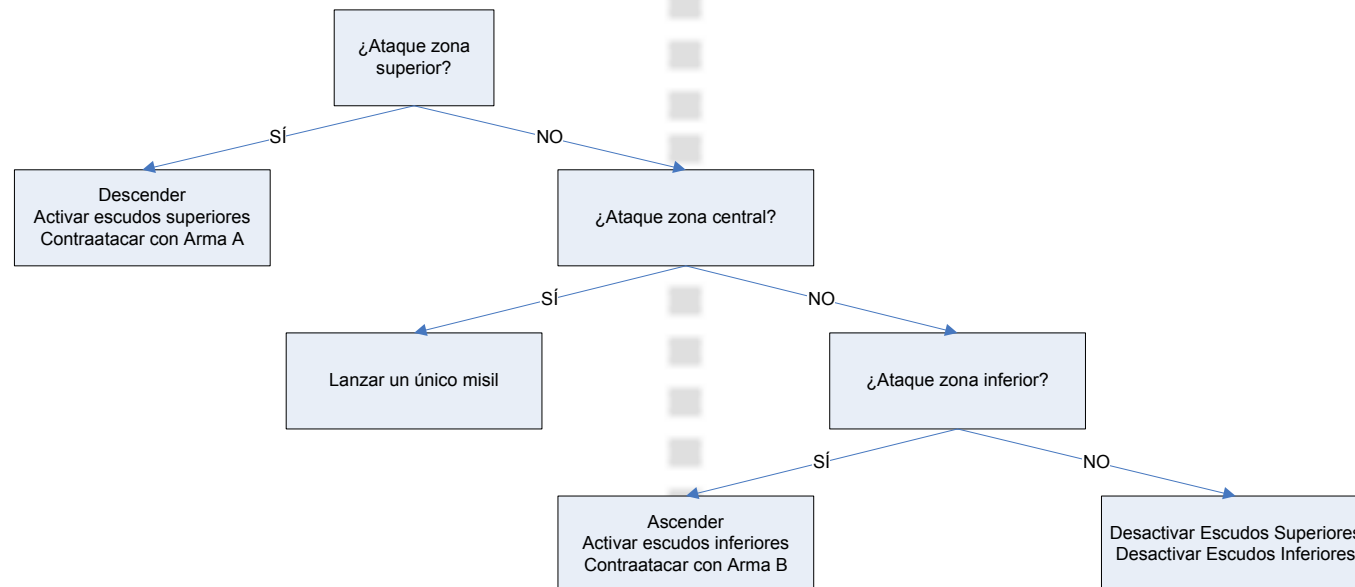
- Simulación de Sistemas: Emular el comportamiento general de un objeto del juego.
  - Patrones: Pautas predefinidas (insertadas en el código). Ejemplo:

```
ControlData {
    double turnRight;
    double turnLeft;
    double stepForward;
    double stepBackward;
};
...
Pattern[0].turnRight = 0;
Pattern[0].turnLeft = 0;
Pattern[0].stepForward = 2;
Pattern[0].stepBackward = 0;
...
Object.orientation += Pattern[CurrentPattern].turnRight;
Object.orientation -= Pattern[CurrentPattern].turnLeft;
Object.x += Pattern[CurrentPattern].stepForward;
Object.x -= Pattern[CurrentPattern].stepBackward;
```

# Técnicas clásicas

---

- Simulación de Sistemas.
  - Árboles de decisión: Pautas predefinidas mediante estructuras en forma de árbol con condiciones.
    - Pueden construirse automáticamente a partir de tablas.
    - Pueden utilizarse para construir reglas de decisión (ID3, C4.5).



# Técnicas clásicas

---

- Simulación de Sistemas.
  - Sistemas de reglas: Pautas predefinidas mediante estructuras tipo if-then.
    - Implementación en código o mediante motor de inferencia.
    - Ejemplo:

```
IF Energia < 20% AND #Enemigos > 2 THEN Alejar
IF Energia > 70% || Arma = Rocket THEN Perseguir
...
acciones = triggerRules(estadoActual);
```

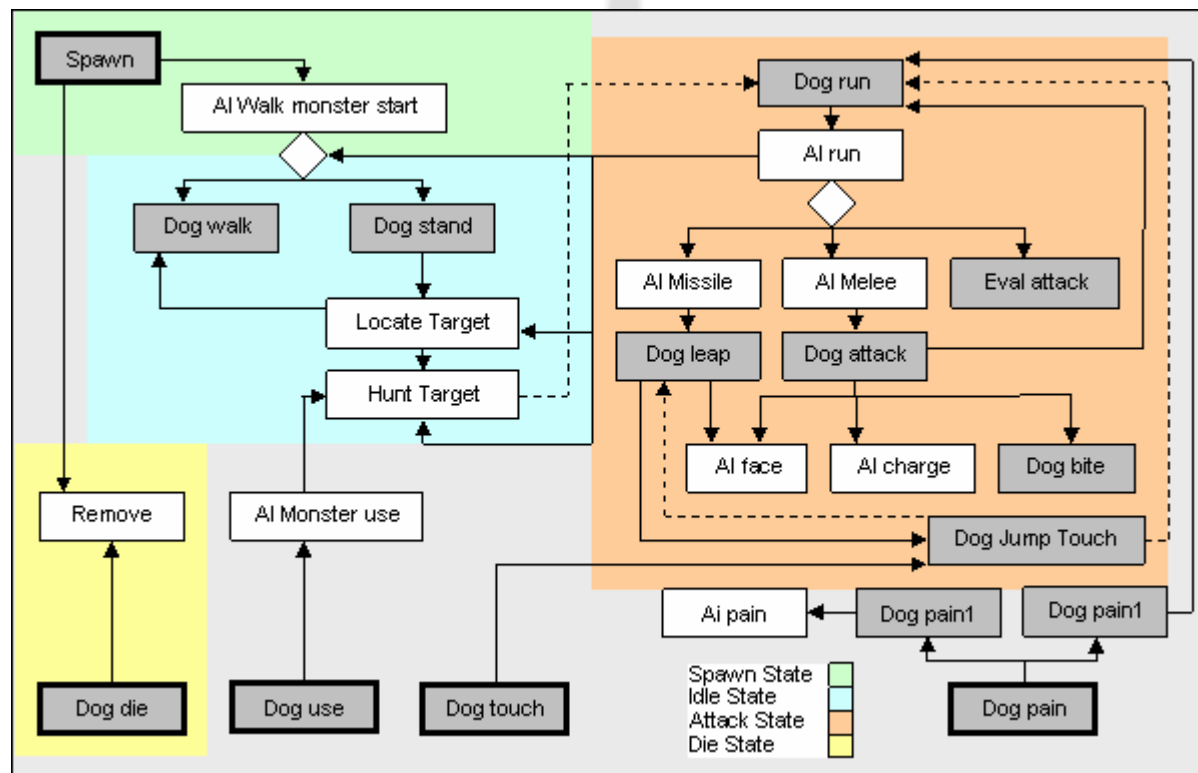
# Técnicas clásicas

---

- Simulación de Sistemas.
  - Autómatas Finitos Deterministas: Diagramas de estados y transiciones que codifican el comportamiento de los elementos.
    - Ventajas:
      - Fáciles de implementar
      - Bajo coste computacional
      - Intuitivos
      - Pueden transformarse en sistemas basados en reglas
    - Ejemplos: Fantasmas Pac-Man, Quake Bots, Jugadores Fifa 2002.

# Técnicas clásicas

- Simulación de Sistemas.
  - Autómatas Finitos Deterministas: Quake Dog (<http://ai-depot.com/FiniteStateMachines/FSM-Framework.html>)



# Técnicas clásicas

---

- Técnicas clásicas
  - Chase and evade
    - Basics: Random motion, Patterns, Probabilistic Motion, Tracking
    - Advanced: Flocking, Movimiento basado en función potencial
  - Búsqueda de caminos
    - Grafos
    - A\*
  - Simulación de sistemas
    - Patrones
    - Sistemas de reglas
    - Máquinas de estados finitas
  - **Aprendizaje**
    - **Refuerzo**
    - **Deductivo**
  - Inteligencia colectiva
    - Sistemas de agentes
  - Planificación
    - Agentes dirigidos por metas

# Técnicas clásicas

---

- Aprendizaje: Conjunto de técnicas que permiten a un objeto modificar su comportamiento a partir de la experiencia pasada.
  - Aprendizaje Supervisado:
    - Existe un conjunto de patrones para los que se conoce la salida que debe dar el sistema.
    - El sistema premia el comportamiento que interpola dichos patrones.
    - Ejemplos: Razonamiento basado en casos, Perceptrón Multicapa.
  - Aprendizaje No Supervisado:
    - No existe un conjunto de patrones de entrenamiento.
    - El sistema clasifica las entradas según su similitud, correlación, distancia, etc.
    - Ejemplos: Mapas auto-organizativos, *clustering*.

# Técnicas clásicas

---

- Aprendizaje:
  - Aprendizaje por refuerzo: Método de aprendizaje no supervisado basado en el modelo conductista.
    - No existe un conjunto de patrones de entrenamiento predefinidos.
    - Generalmente, el agente se relaciona con un mundo donde varían las condiciones ambientales.
    - Cada acción produce resultados que se valoran positiva o negativamente.
    - En situaciones futuras se preferirán las acciones que han dado mejores resultados.
    - Es el más apropiado para la programación de videojuegos.
    - Ejemplos: Funciones Q de aprendizaje, Computación Evolutiva.

# Técnicas clásicas

---

- Aprendizaje:
  - Aprendizaje deductivo: Capacidad de deducir nuevo conocimiento a partir del existente.
    - Creación de bases de conocimiento estructuradas.
    - Uso de lógicas formales y razonadores.
    - Similitud con sistemas expertos.
    - Obtención de comportamiento emergente.
    - Mayor esfuerzo de modelado.

# Técnicas clásicas

---

- Técnicas clásicas
  - Chase and evade
    - Basics: Random motion, Patterns, Probabilistic Motion, Tracking
    - Advanced: Flocking, Movimiento basado en función potencial
  - Búsqueda de caminos
    - Grafos
    - A\*
  - Simulación de sistemas
    - Patrones
    - Sistemas de reglas
    - Máquinas de estados finitas
  - Aprendizaje
    - Refuerzo
    - Deductivo
  - **Inteligencia colectiva**
    - **Sistemas de agentes**
  - Planificación
    - Agentes dirigidos por metas

# Técnicas clásicas

---

- Sistemas de Agentes: Un agente es una entidad autónoma que recoge información del ambiente (perceptivo) y actúa sobre él (proactivo) con el propósito de alcanzar un objetivo (guiado por metas).
  - Sistemas Multiagente:
    - Sistema complejo donde diferentes agentes independientes se coordinan para realizar una tarea.
    - Comportamiento de grupo.
    - Plataformas, lenguajes y herramientas para el desarrollo de agentes.
    - Diferentes arquitecturas: jerárquica, p2p, pizarra, etc.
    - Ejemplos: RoboCup, Quake II (<http://www.cs.rochester.edu/~ferguson/papers/brown-et-al-quagents-tr853.pdf>), etc.

# Técnicas clásicas

---

- Técnicas clásicas
  - Chase and evade
    - Basics: Random motion, Patterns, Probabilistic Motion, Tracking
    - Advanced: Flocking, Movimiento basado en función potencial
  - Búsqueda de caminos
    - Grafos
    - A\*
  - Simulación de sistemas
    - Patrones
    - Sistemas de reglas
    - Máquinas de estados finitas
  - Aprendizaje
    - Refuerzo
    - Deductivo
  - Inteligencia colectiva
    - Sistemas de agentes
  - **Planificación**
    - **Agentes dirigidos por metas**

# Técnicas clásicas

---

- Planificación: Cálculo de una composición de acciones cuya ejecución provoca la consecución de un objetivo.
  - Agentes guiados por metas: Agentes que realizan tareas de planificación:
    - Poseen una representación del mundo y de sus propias habilidades
    - Son capaces de construir un plan para modificar el estado actual hacia una situación objetivo.
    - Descripción jerárquica de tareas y subtareas.
    - Cada acción puede tener asociado un coste y/o una preferencia.
    - Implementación: grafos, cálculo situacional, HTN-planning, etc.
    - Ejemplo: Hunt the Wumpus.

# Esquema

---

- Elementos de Inteligencia Artificial en videojuegos
- **Técnicas de Inteligencia Artificial**
  - Técnicas Clásicas
  - **Técnicas Avanzadas**

# Técnicas avanzadas

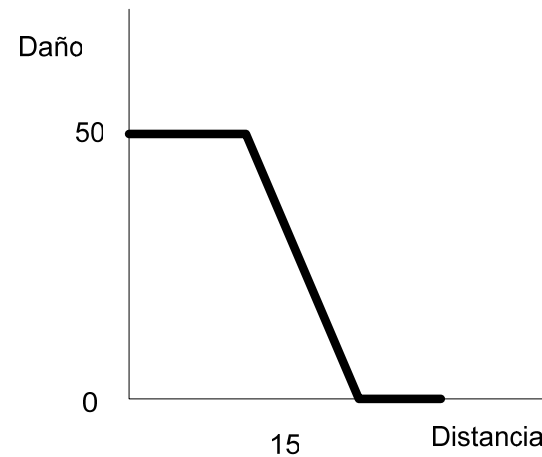
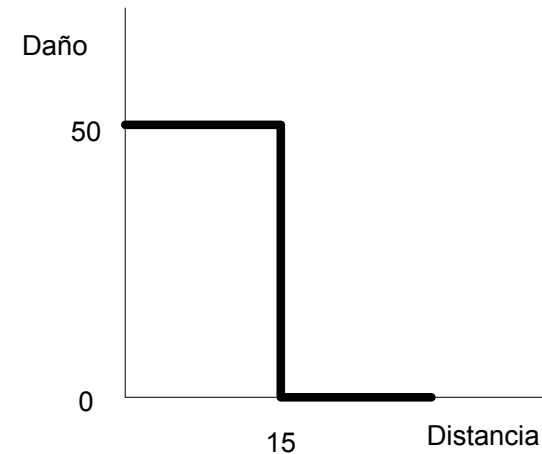
---

- Técnicas Avanzadas
  - Lógica Difusa
  - Computación Evolutiva
  - Redes Neuronales
  - Vida Artificial
  - Modelos Probabilísticos

# Técnicas Avanzadas

---

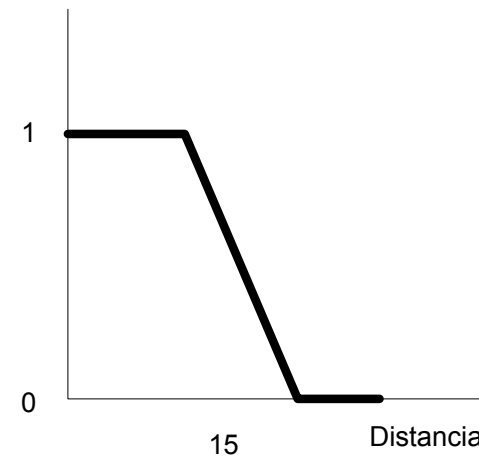
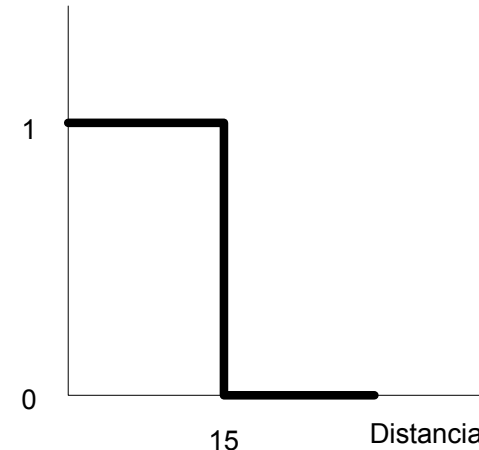
- Lógica Difusa:
  - Ampliación de la lógica clásica en la que cada proposición tiene asociado un grado de verdad o cumplimiento.
    - Lógica clásica: conceptos *crisp*.
    - Lógica difusa: conceptos *fuzzy*.
  - Ejemplo: Daño inflingido por un impacto.



# Técnicas Avanzadas

---

- Lógica difusa:
  - Conjunto difuso sobre el dominio  $D$ :
    - Se asocia una función de pertenencia  $f: D \rightarrow [0, 1]$  que determina el grado de pertenencia de un individuo  $d \in F$ .
    - Ejemplo: Conjunto “distancias peligrosas”
  - Operaciones sobre conjuntos difusos:
    - Unión: t-conorma (max, + , etc.)
    - Intersección: t-norma (min,  $\cdot$  , etc.)



# Técnicas Avanzadas

---

- Lógica Difusa:
  - Utilidad:
    - Sistemas de reglas difusas.
    - Sistemas de control difuso.
    - Puede utilizarse como parte de otras técnicas: Máquinas de Estado Difusas (f-FSM), Redes Neuronales Híbridas, etc.
  - Ejemplos:
    - Close Combat 2, The Sims, Battlecruiser: 3000AD, Civilization: Call to Power, S.W.A.T. 2, etc.

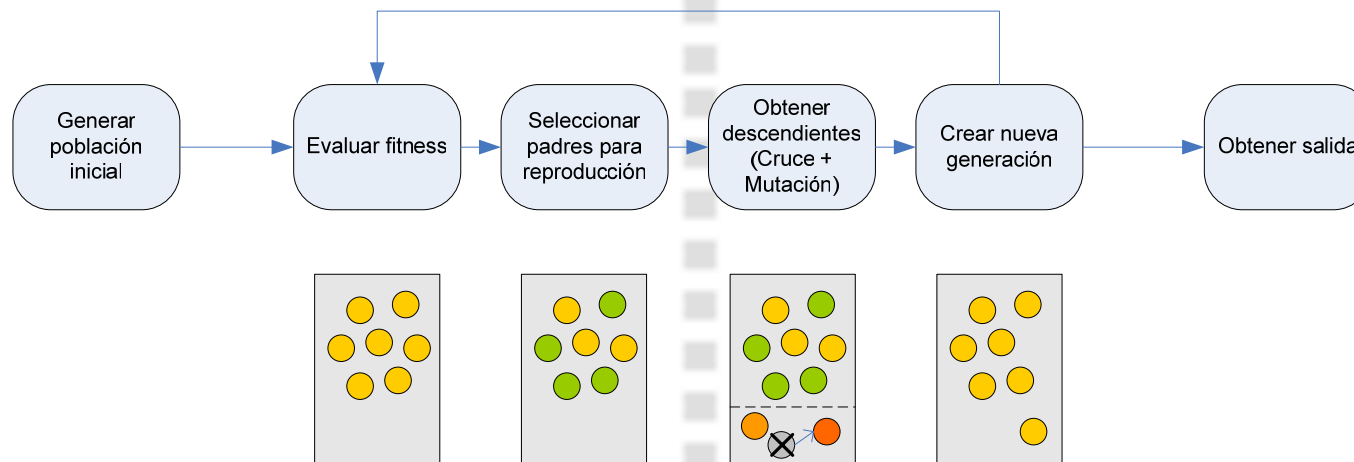
# Técnicas Avanzadas

---

- Computación Evolutiva:
  - Algoritmos de optimización basados en ideas neodarwinistas: calcular una población de *buenos* individuos mediante repetición de selección, cruce y mutación.
  - Terminología:
    - Individuo o espécimen: representación numérica de una solución para un problema.
    - Población: conjunto de soluciones.
    - Generación *i-ésima*: iteración número *i* del algoritmo.
    - Selección: toma en consideración de un conjunto de individuos de una población según un criterio de optimalidad.
    - Cruce: combinación de individuos para generar nuevas soluciones.
    - Mutación: modificación, generalmente aleatoria, de una solución.
    - Función de *fitness*: función que valora numéricamente la *bondad* de una solución.

# Técnicas Avanzadas

- Computación Evolutiva:
  - Equilibrio:
    - Convergencia (*selección*):
      - ↑ El algoritmo obtiene buenas soluciones en pocas generaciones.
      - ↓ El algoritmo se estanca en óptimos locales.
    - Diversidad (*mutación*):
      - ↑ El algoritmo explora todo el espacio de búsqueda.
      - ↓ El algoritmo tarda mucho en obtener buenas soluciones.



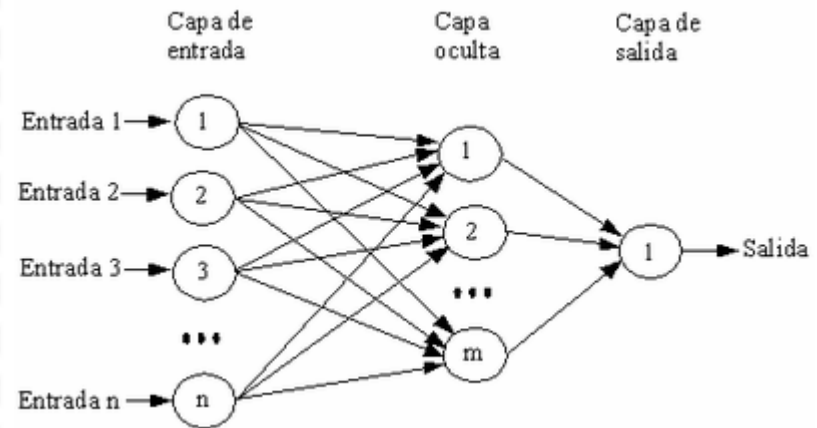
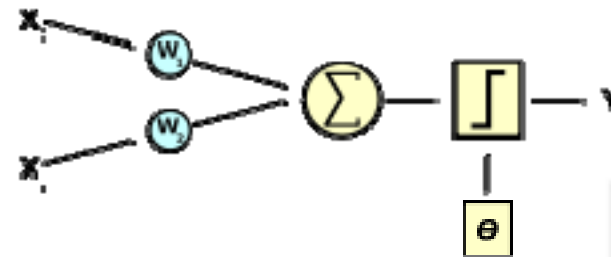
# Técnicas Avanzadas

---

- Utilidad:
  - Aplicables a problemas de optimización que no tienen solución numérica sencilla, ya que encuentran soluciones aceptables en un tiempo razonable.
  - Apropriados en juegos donde se puede codificar el comportamiento de un personaje y valorar el beneficio obtenido.
  - Hibridación con otros algoritmos: búsqueda clásica.
- Ejemplos:
  - *Cloak, Dagger, and DNA*, Creatures Series, Return Fire 2, Sigma.

# Técnicas Avanzadas

- Redes Neuronales:
  - Modelo matemático que simula el comportamiento de una neurona humana.
  - Elementos:
    - Datos de entrada
    - Datos de salida
    - Función de cómputo
    - Función de activación (umbral)



# Técnicas Avanzadas

---

- Redes Neuronales:
  - Ejemplos: Perceptrón (multicapa), Redes Recurrentes
  - Aprendizaje con RNAs (aproximadores universales):
    - Modificar los pesos de las conexiones para reproducir ciertos patrones de entrada.
    - Uso de algoritmos de entrenamiento para obtener dichos pesos automáticamente:
      - Supervisado (uso de ejemplos de entrenamiento para los que se conoce la salida): perceptrón multicapa (backpropagation), redes neuronales recurrentes (algoritmos genéticos), etc. (Aproximación de funciones)
      - No supervisado (no se conoce la salida para un patrón de entrada): mapas auto-organizativos (clustering), redes de Hopfield, etc. (Reconocimiento de patrones indefinidos)

# Técnicas Avanzadas

---

- Redes Neuronales:
  - Utilidad:
    - Permiten “aprender” comportamientos a partir del conjunto de patrones de entrenamiento.
    - Facilitan la reproducción de comportamientos difíciles de modelar matemáticamente.
    - Las salidas pueden modificarse variando los pesos de la red, con la cual el comportamiento puede actualizarse automáticamente.
  - Ejemplos:
    - Creatures Series.

# Técnicas Avanzadas

---

- Vida Artificial:
  - Conjunto de técnicas para simulación de sistemas que recrean artificialmente fenómenos biológicos.
  - El objetivo es observar a los organismos vivos y construir sistemas que se comporten como éstos para resolver algún problema.
  - Técnicas:
    - Algoritmos de colonias de hormigas.
    - Inteligencia de enjambre.
    - Sistemas de partículas.
  - En realidad, prácticamente todos los videojuegos intentan reproducir el comportamiento de alguna entidad biológica.

# Técnicas Avanzadas

---

- Vida Artificial:
  - Colonias de hormigas:
    - Técnica de optimización para problemas que pueden solucionarse como búsquedas sobre grafos (TSP, enrutamiento, caminos mínimos, etc.).
    - Simula el comportamiento de las hormigas:
      - Inicialmente, movimiento aleatorio.
      - Si se encuentra comida, se marca con feromona el camino seguido hasta encontrarla.
      - Una hormiga tenderá a seguir la feromona depositada por otras hormigas.
    - Ventaja: Tolerante a cambios en el grafo del problema.

# Técnicas Avanzadas

---

- Vida Artificial:
  - Utilidad:
    - En general, es conveniente simular realísticamente el comportamiento de los personajes del juego.
    - Permiten implementar de forma natural comportamientos.
    - Suelen apoyarse en otras técnicas de Inteligencia Artificial.
  - Ejemplos:
    - The Sims, Nintendogs, etc.

# Recursos

---

- *AI Game Programming Wisdom*. Ed. Steve Rabin. Charles River Media, 2002.
- *AI Techniques for Game Programming*. Mat Buckland. Premier Press, 2003.
- *Programming Game AI by example*. Mat Buckland. Wordware Publishing, 2005.
- *AI for Game Developers*. D. M. Bourg, G. Seeman. O'Reilly Media, 2004.
- *Game Programming Gems*. Charles River Media.
- <http://www.gameai.com/>
- <http://www.aiguru.com>
- <http://aaaipress.org/AITopics/html/video.html>