Fundamentos de Entretenimiento Digital

Diego C. Martínez

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

John McCarthy

Inteligencia Artificial

una ciencia

objetivo científico

comprender los principios del comportamiento inteligente.

una disciplina ingenieril

objetivo ingenieril

especificar métodos para el diseño de artefactos inteligentes.

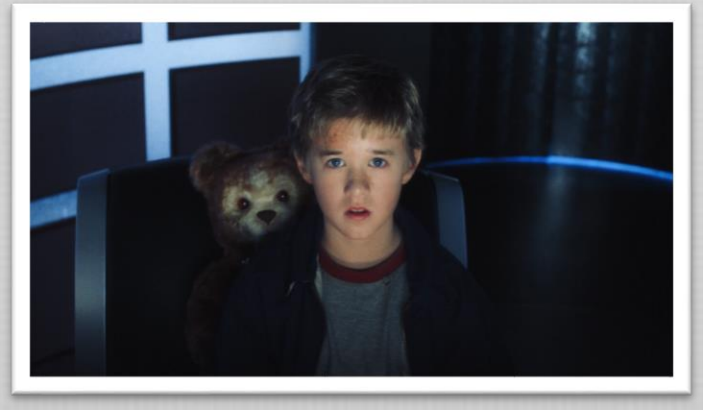
Dos visiones diferentes de la Inteligencia Artificial

Inteligencia Artificial *Fuerte*

Las máquinas que muestren inteligencia tendrán mentes reales y serán conscientes

Inteligencia Artificial *Débil*

Es posible lograr que las máquinas actúen como si fueran inteligentes.



David
¿débil o fuerte?

IA Académica vs IA en videojuegos

IA Académica

No se busca un efecto particular, sino comportamiento general

A veces el efecto justifica el costo

Algunas áreas de investigación buscan superar al humano

Sin limitaciones de ideas y desarrollo

IA videojuegos

Usualmente se buscan efectos puntuales

A veces el costo no justifica el efecto

Superar *siempre* al humano no es negocio

Ideas y desarrollo acotado por el mercado

IA en videojuegos

¿en qué se aplica
la inteligencia artificial
en los videojuegos?

enemigos

amigos

grupos

navegación
decisiones
diálogos
reactividad y proactividad
comportamiento creíble

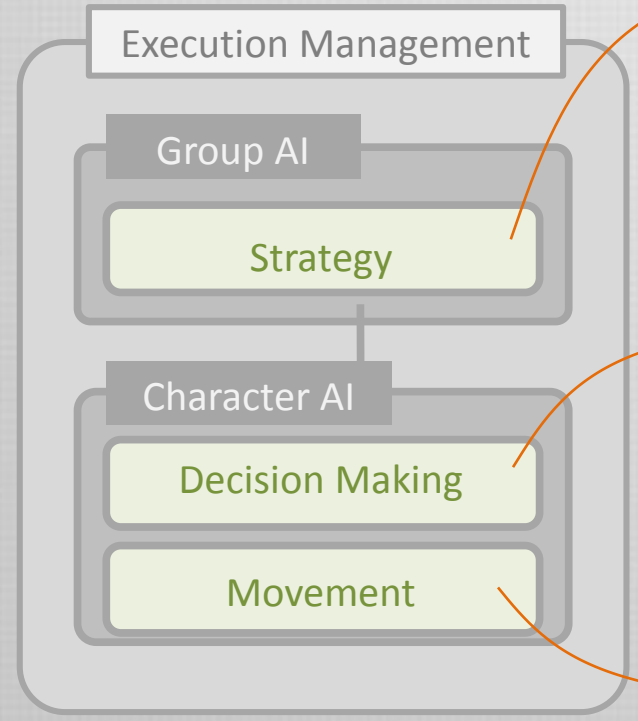
IA en videojuegos

¿en qué se aplica
la inteligencia artificial
en los videojuegos?

En la generación de diálogos en Role Playing Games
En la generación de enemigos que evolucionan
En la simulación completa de un jugador virtual (bot)
En la generación de mapas y terrenos
En el comportamiento emocional de los NPC

en crear mejores oponentes, en ayudar al jugador,
en extender el tiempo de vida del juego,
en ser un producto diferente a los demás.

IA en videojuegos



Decision Strategy

Algoritmos y técnicas enfocados en el comportamiento grupal
(actitud agresiva, defensiva, toma de control del terreno, etc)

Decision Making

Algoritmos y técnicas para la toma de decisiones de los caracteres sintéticos
(atacar, defender, construir, destruir, selección de acciones encadenadas, planificación, etc)

Movement

Algoritmos y técnicas para el movimiento de los caracteres sintéticos
(moverse, buscar, perseguir, esquivar, escapar, pasear, etc)

Movimiento

Character

- posición
- velocidad

Algoritmo de movimiento

Movement request

- nueva posición
- nueva velocidad

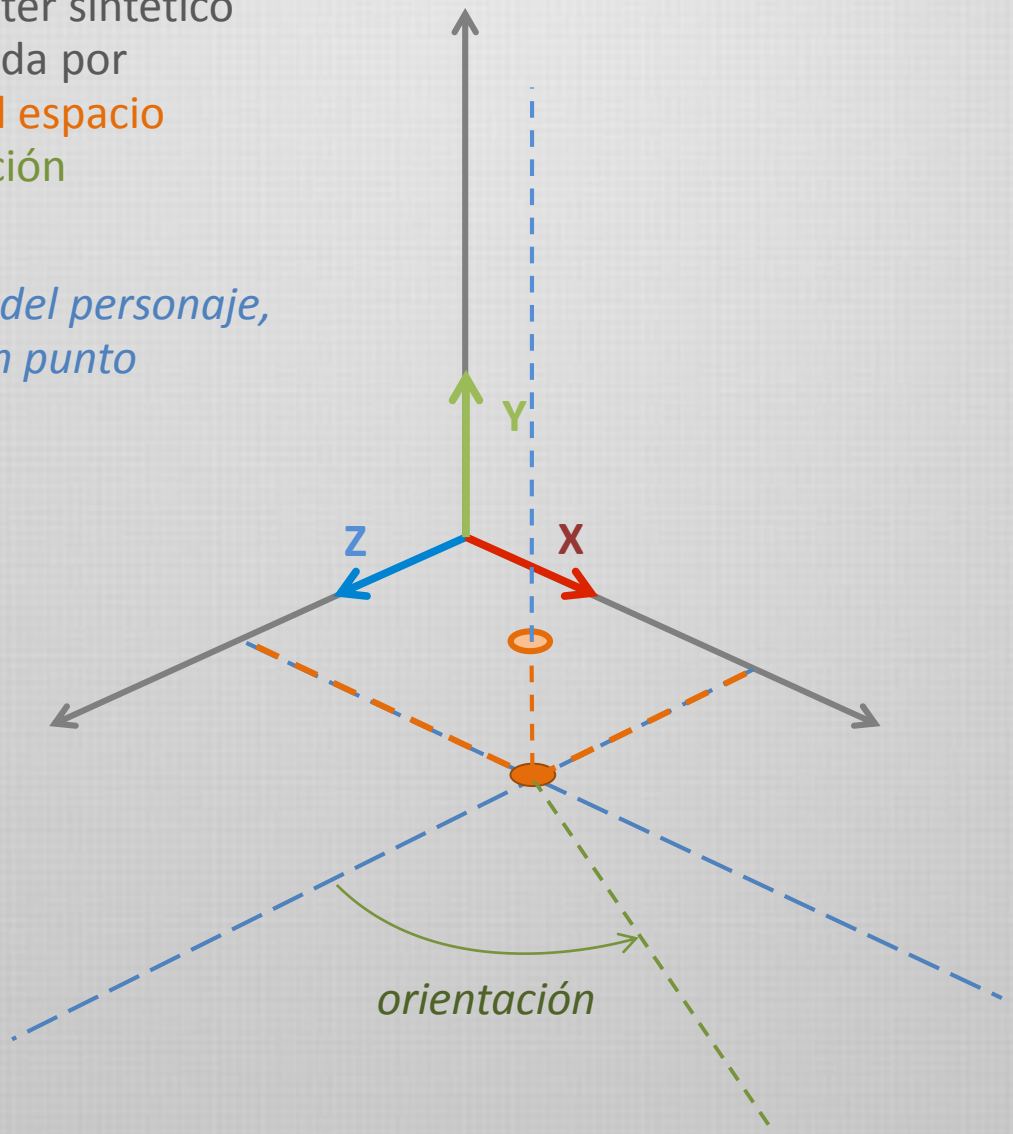
Game

- otros caracteres
- geometria del nivel
- paths
- estado del juego

Movimiento

La posición del caracter sintético
está determinada por
la ubicación en el espacio
y la orientación

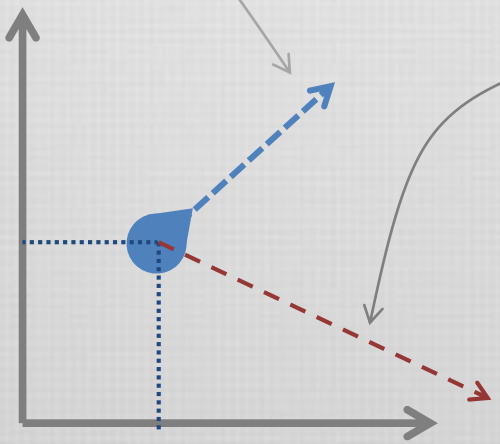
No importa el tamaño del personaje,
la posición es un punto



Movimiento

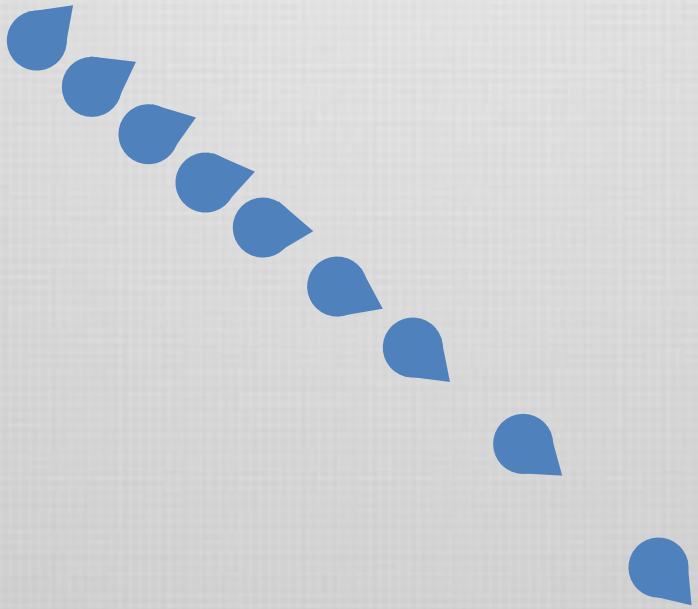
velocidad lineal
(un vector con magnitud)
Representa la tasa de cambio de posición con respecto al tiempo

aceleración
(un vector con magnitud)
tasa de cambio de la velocidad con respecto al tiempo

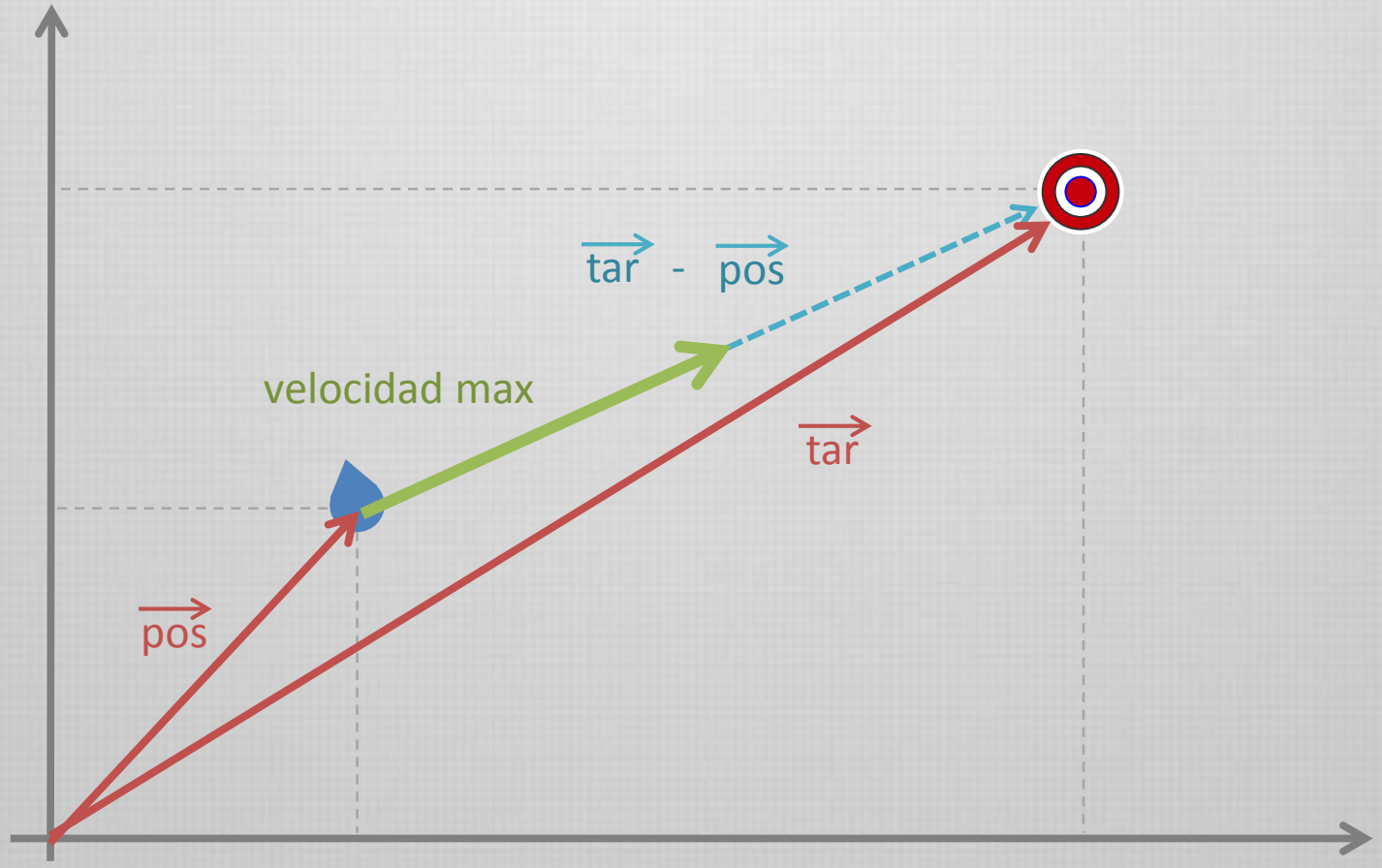


velocidad angular
(un valor en radianes)
#radianes por segundo para cambiar la orientación

Movimiento



Movimiento



seek

Seek

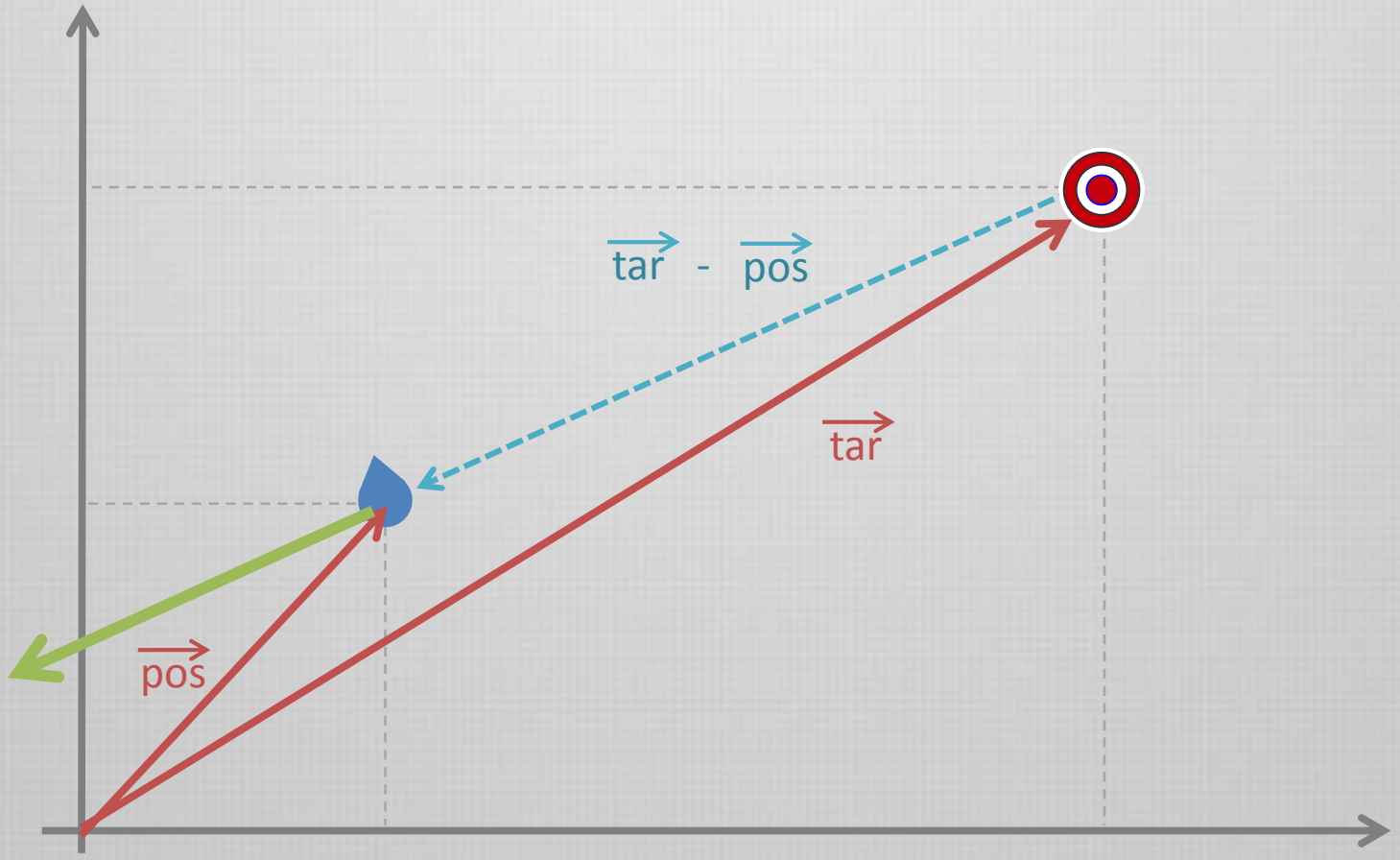
Artificial Intelligence for Games.
Millington - Funge

```
def getSteering():

        #Create the structure for output
        steering = new KinematicSteeringOutput()

        # Get the direction to the target
        steering.velocity =   target.position - character.position

        # The velocity is along this direction, at full speed
        steering.velocity.normalize()
        steering.velocity *= maxSpeed

        ...
        return steering
```

Programming game AI by Example
Mat Buckland

```
Vector2D SeekSteering(Vector2D TargetPos)
{
    Vector2D DesiredVelocity = Vec2DNormalize(TargetPos - m_pVehicle->Pos()) * m_pVehicle->MaxSpeed();
    return (DesiredVelocity - m_pVehicle->Velocity());
}
```

¿cuál es el problema de utilizar la velocidad máxima?
¿cual es la solución?

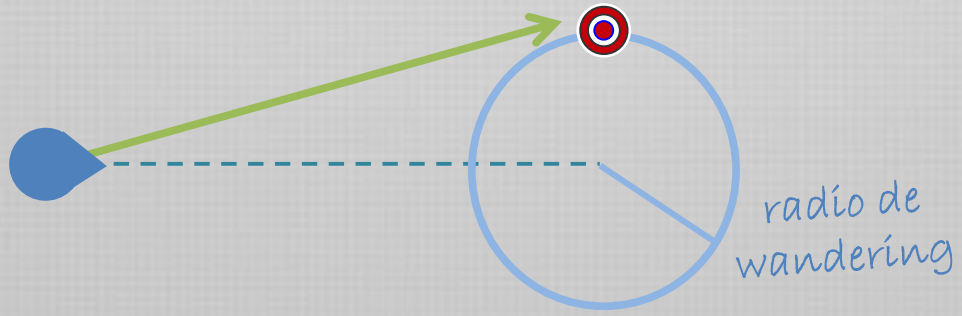


flee

Wander

Wander es un movimiento cinemático frecuente.
Representa un andar aleatorio con una orientación general de poca variación.

La orientación está dada por un ángulo de rotación aleatorio, tendiente a 0
Puede implementarse en base a *Seek* generando puntos al azar



Steering Behaviors

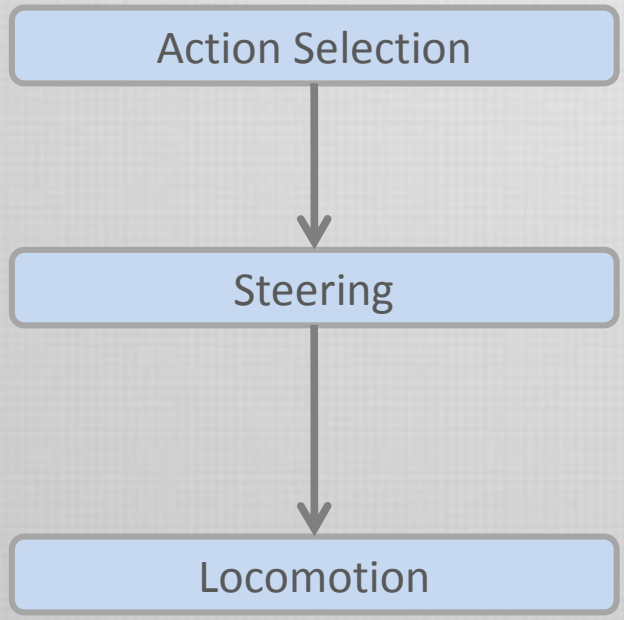
Crayg Reynolds presentó en 1999 el concepto de *steering behaviors* para caracteres autónomos

"situated, embodied, reactive, virtual agents"

- situado en un mundo virtual junto con otras entidades
- con una manifestación física concreta
- sus acciones son inducidas por eventos del entorno
- agentes reales en un mundo virtual

Steering Behaviors

Sugiere que el comportamiento de un agente autónomo puede descomponerse en tres capas



Decide las acciones de acuerdo a las metas seleccionadas.

Calcula las trajectorias deseadas para cumplir lo decidido en la capa anterior

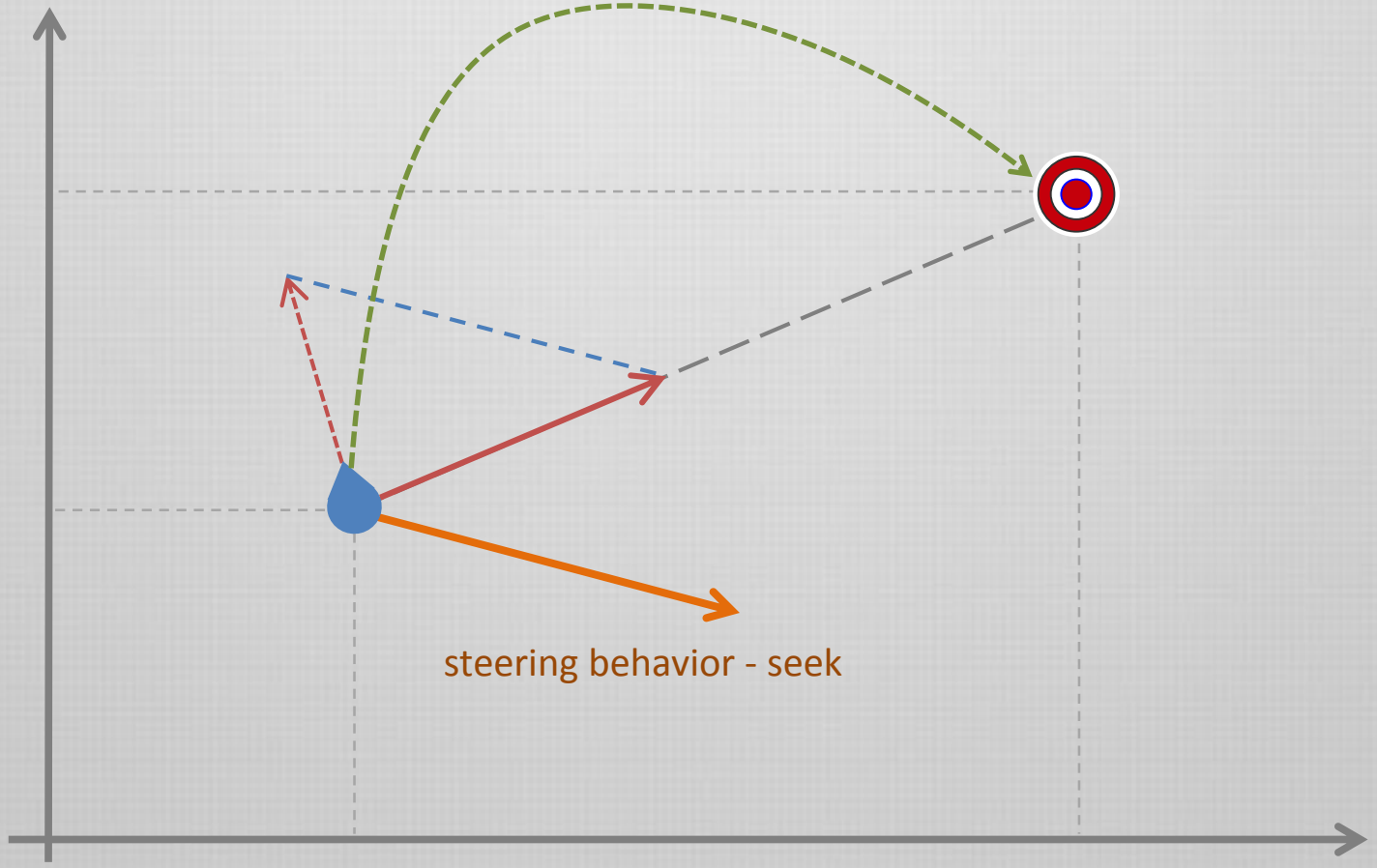
Aspectos mecánicos del movimiento del agente

Steering Behaviors

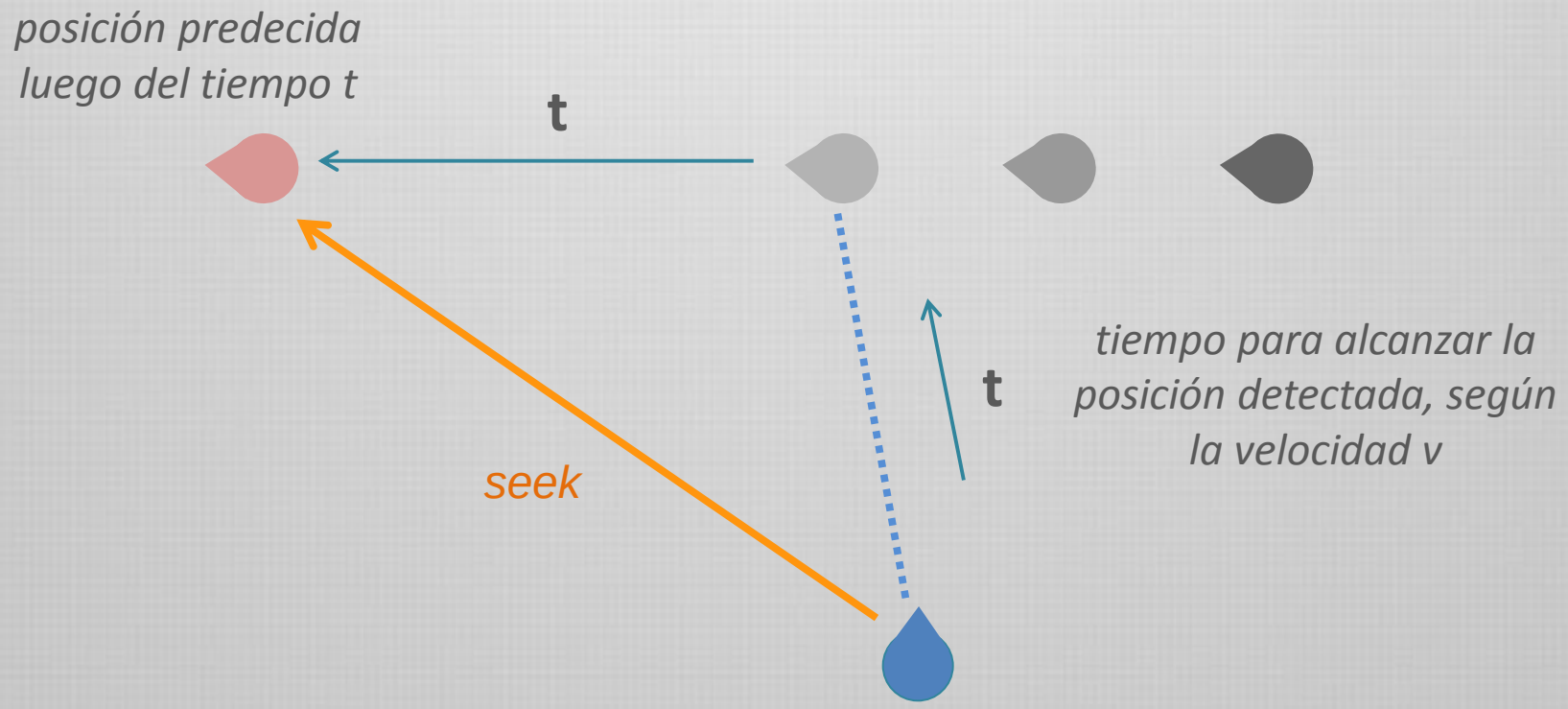
seek
flee
pursuit
evasion
offset pursuit
arrival
obstacle avoidance
wander
path following
wall following
containment
flow field following
unaligned collision avoidance
separation
cohesion
alignment
flocking
leader following

Los steering behaviors pueden combinarse / modificarse para formar nuevos steering behaviors

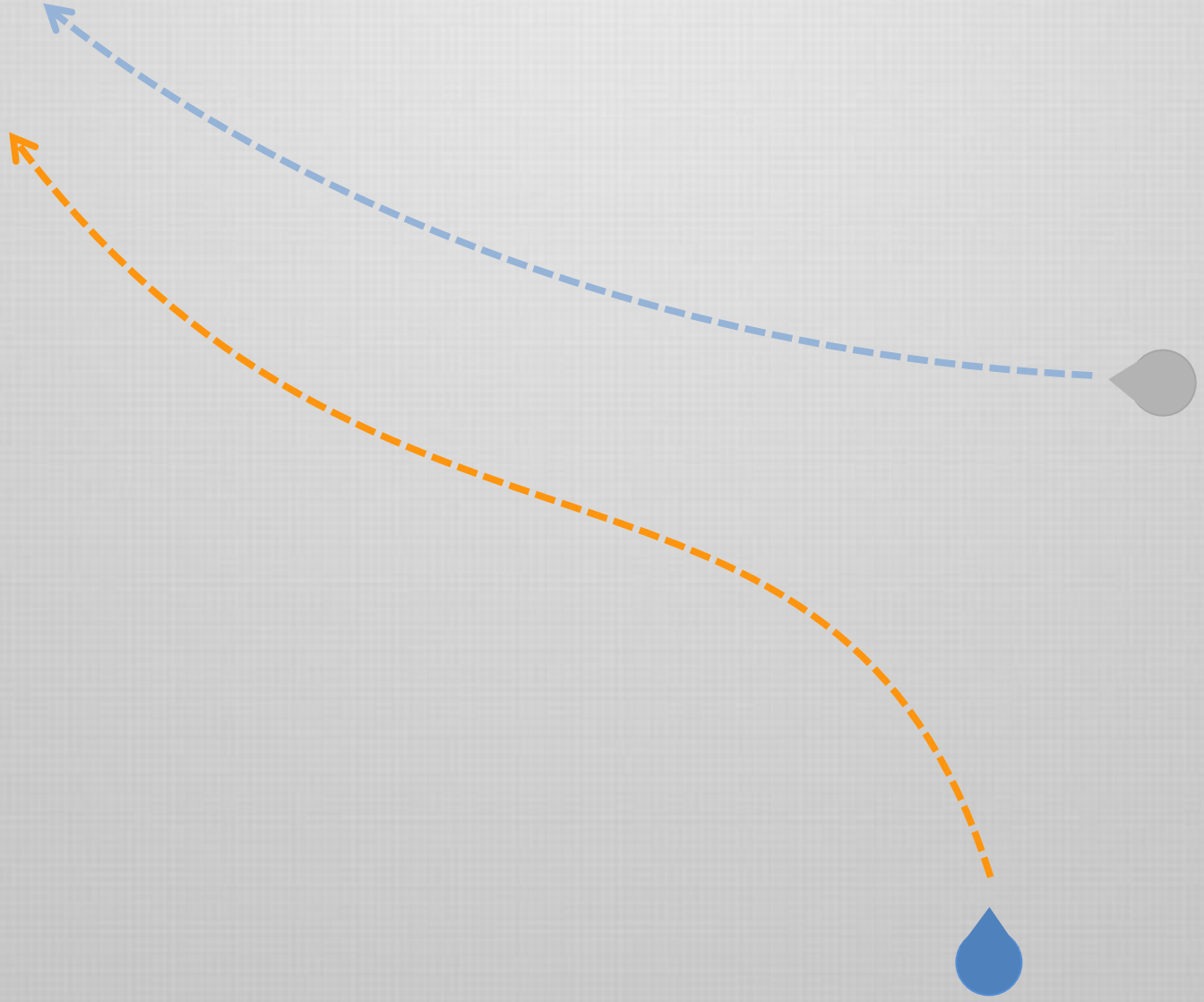
Steering Behaviors - seek



Steering Behavior - pursuit



Steering Behavior - pursuit

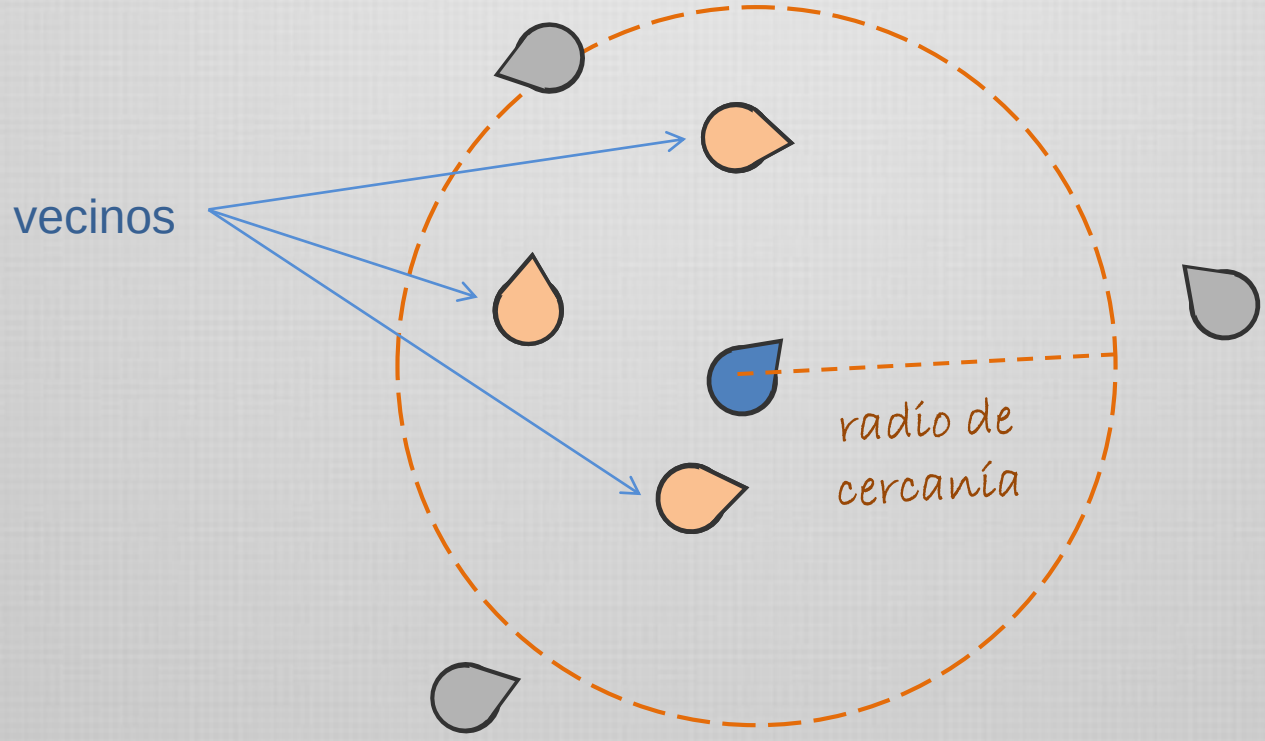


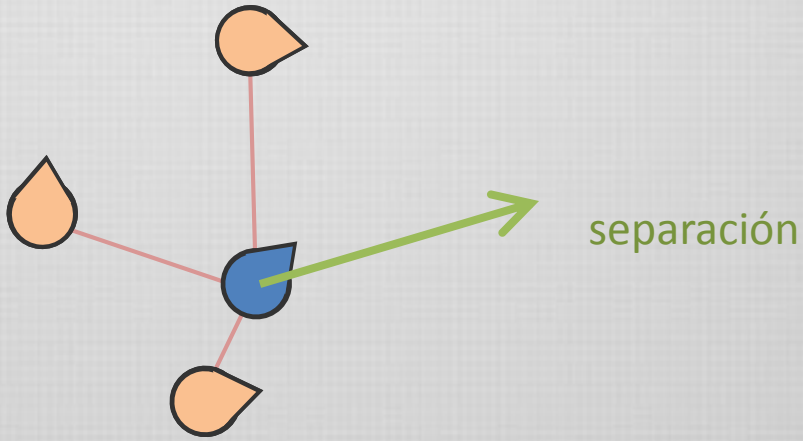
Steering Behaviors

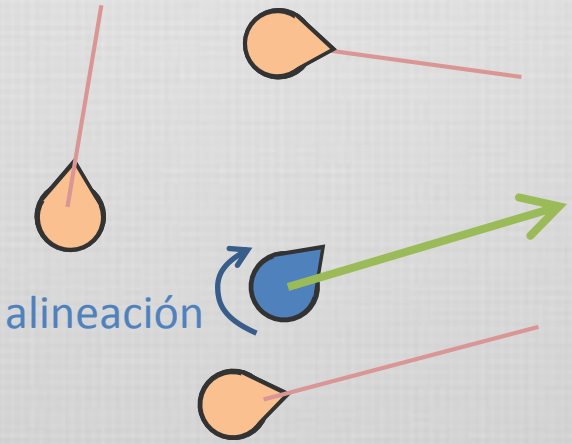
Los *steering behaviors* pueden combinarse y ponderarse para construir comportamientos complejos y variados

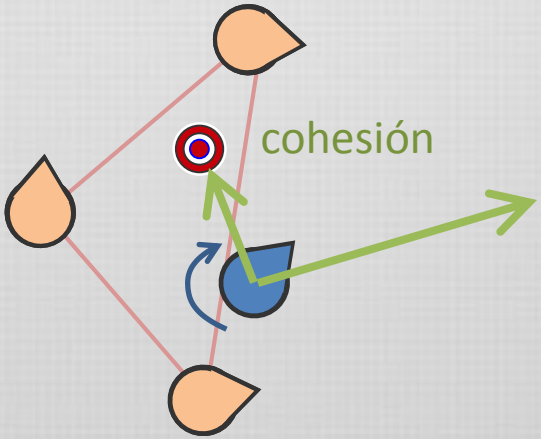
Un ejemplo muy popular es flocking

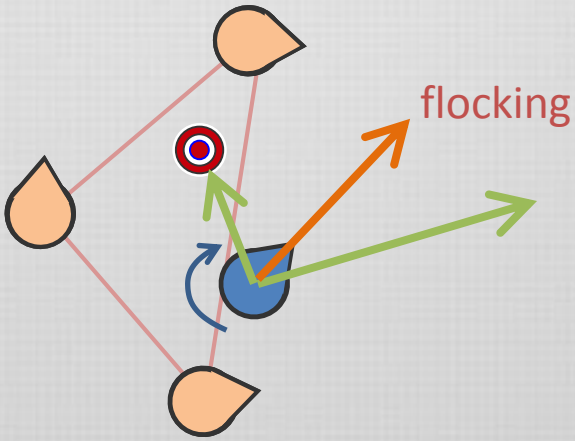
Flocking es un comportamiento emergente
que se logra fácilmente combinando tres steering behaviors:
separación + cohesion + alineación


vecinos
radio de
cercanía


separación

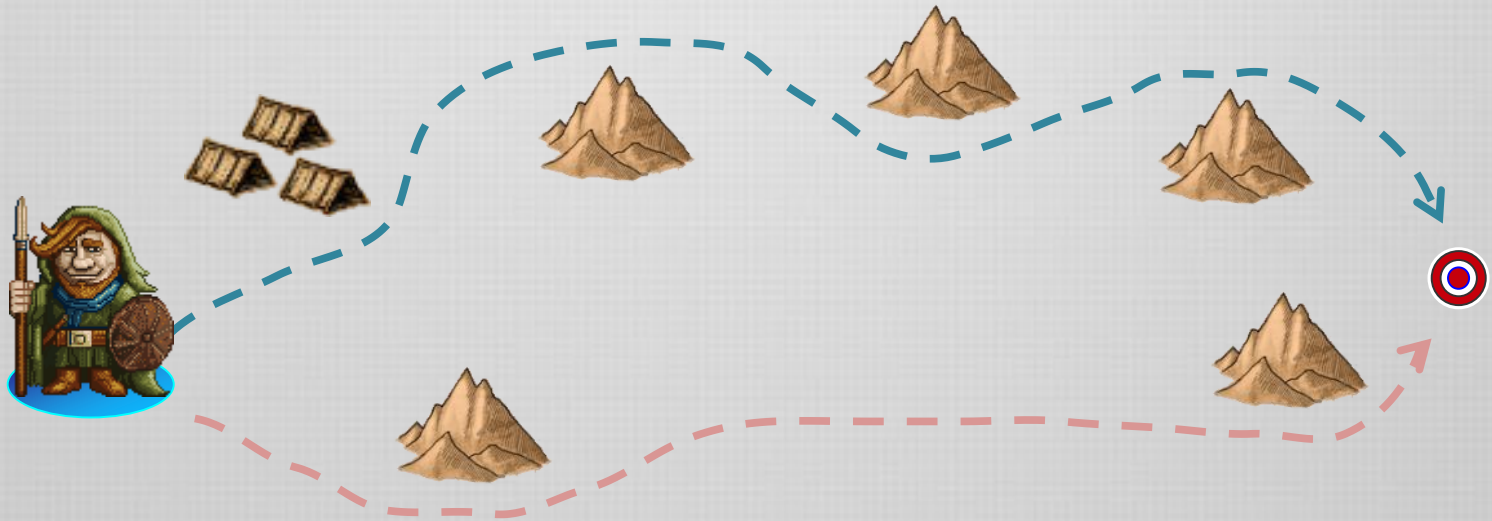

alineación


cohesión


flocking

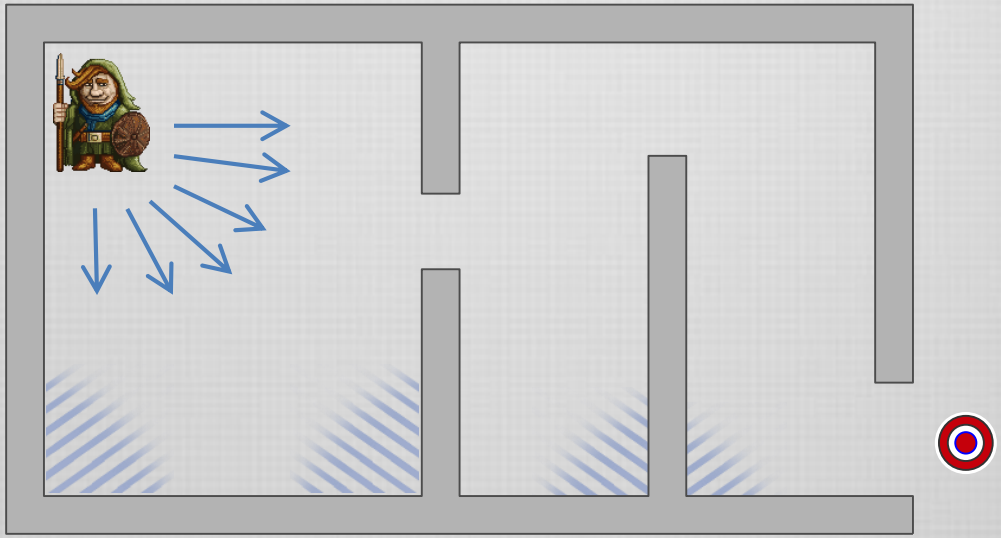
Pathfinding

En la mayoría de los juegos, es necesario además decidir *por dónde* moverse



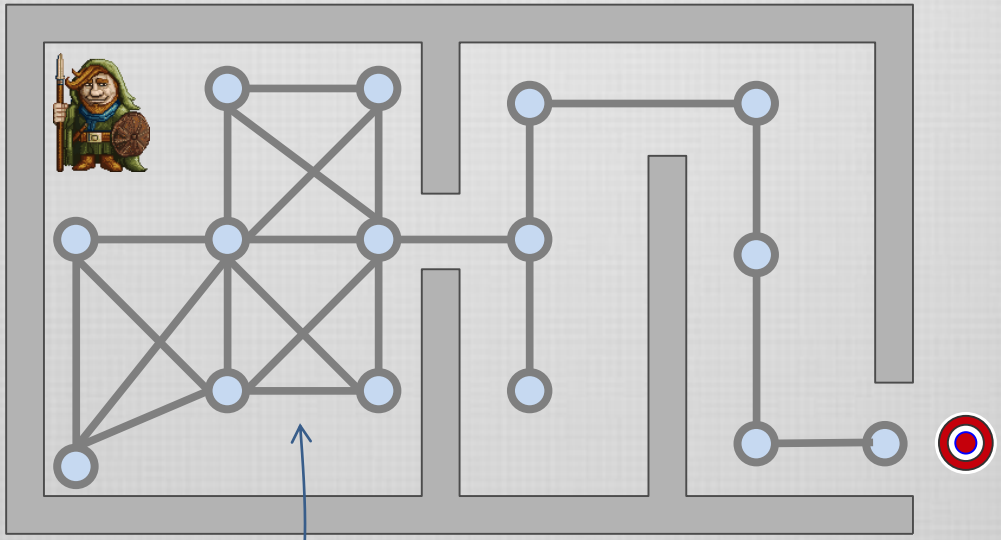
La tarea de
encontrar un camino desde una posición a otra en el escenario
se denomina
pathfinding o *path planning*

Pathfinding



¿cuales son las posiciones que deben explorarse?

Pathfinding



navigation graph

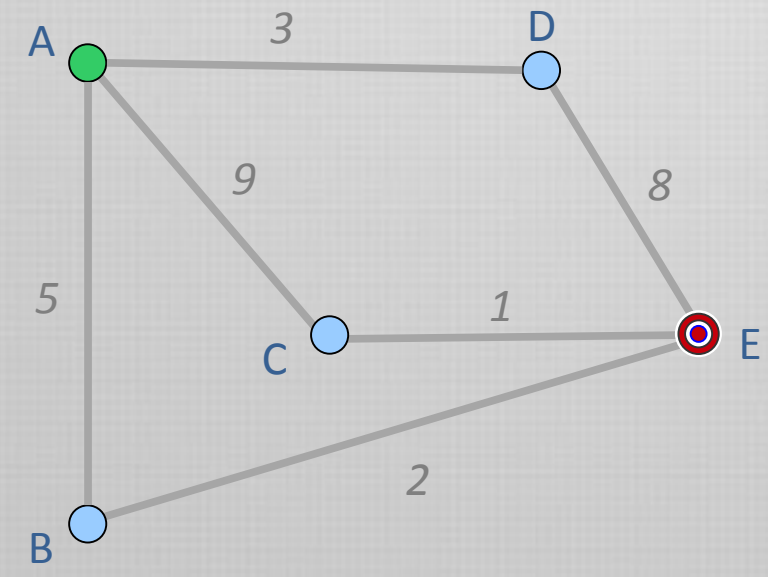
Los nodos y los arcos pueden tener "pesos" diferentes

Pathfinding

Los algoritmos más utilizados para la búsqueda de caminos son *Dijsktra* y *A**
Existen además muchas variaciones y optimizaciones.

La idea general es explorar nodos del grafo según el costo acumulado.

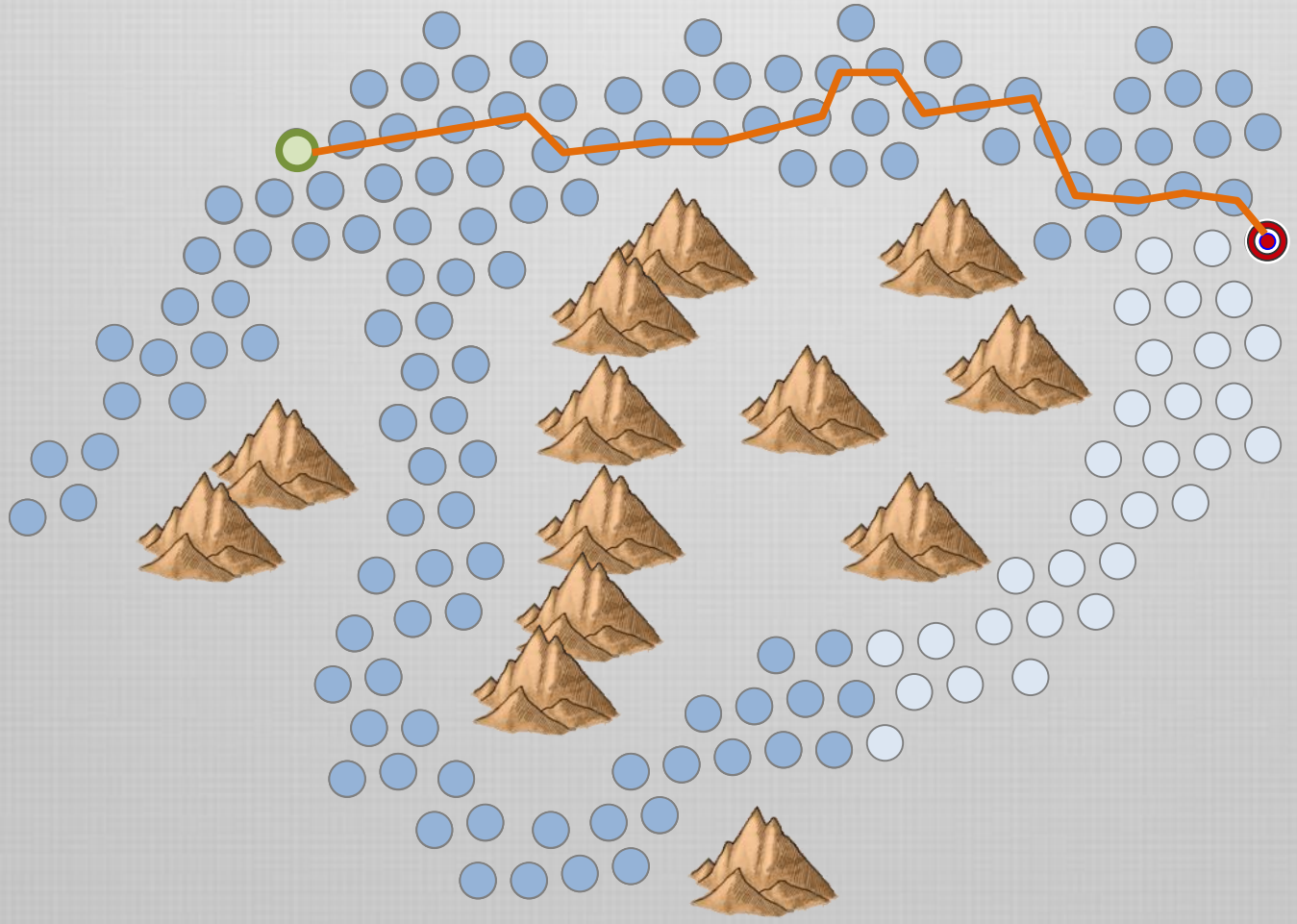
Dijkstra



Nodos elegidos	
A	
A	*AB – costo 5* *AC – costo 9* *AD – costo 3*
AD	*AB – costo 5* *AC – costo 9* *ADE – costo 11*
AB	*AC – costo 9* *ADE – costo 11* *ABE – costo 7*
ABE	

Pathfinding

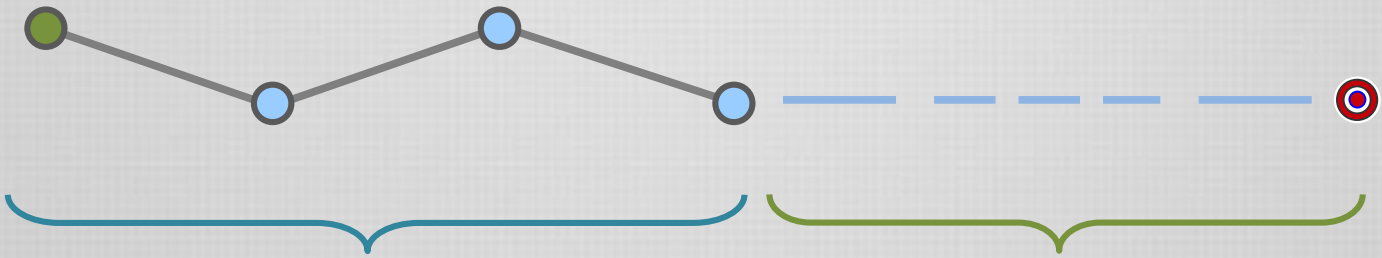
Dijkstra hace una exploración exhaustiva, sin ninguna orientación especial hacia la meta



Pathfinding

El algoritmo A* es una viariación de Dijkstra que agrega estimación de distancia a la meta

siempre de acuerdo a la noción de costo



costo real del camino parcial

estimación del resto (heurística)

g(s) + h(s) = f(s)

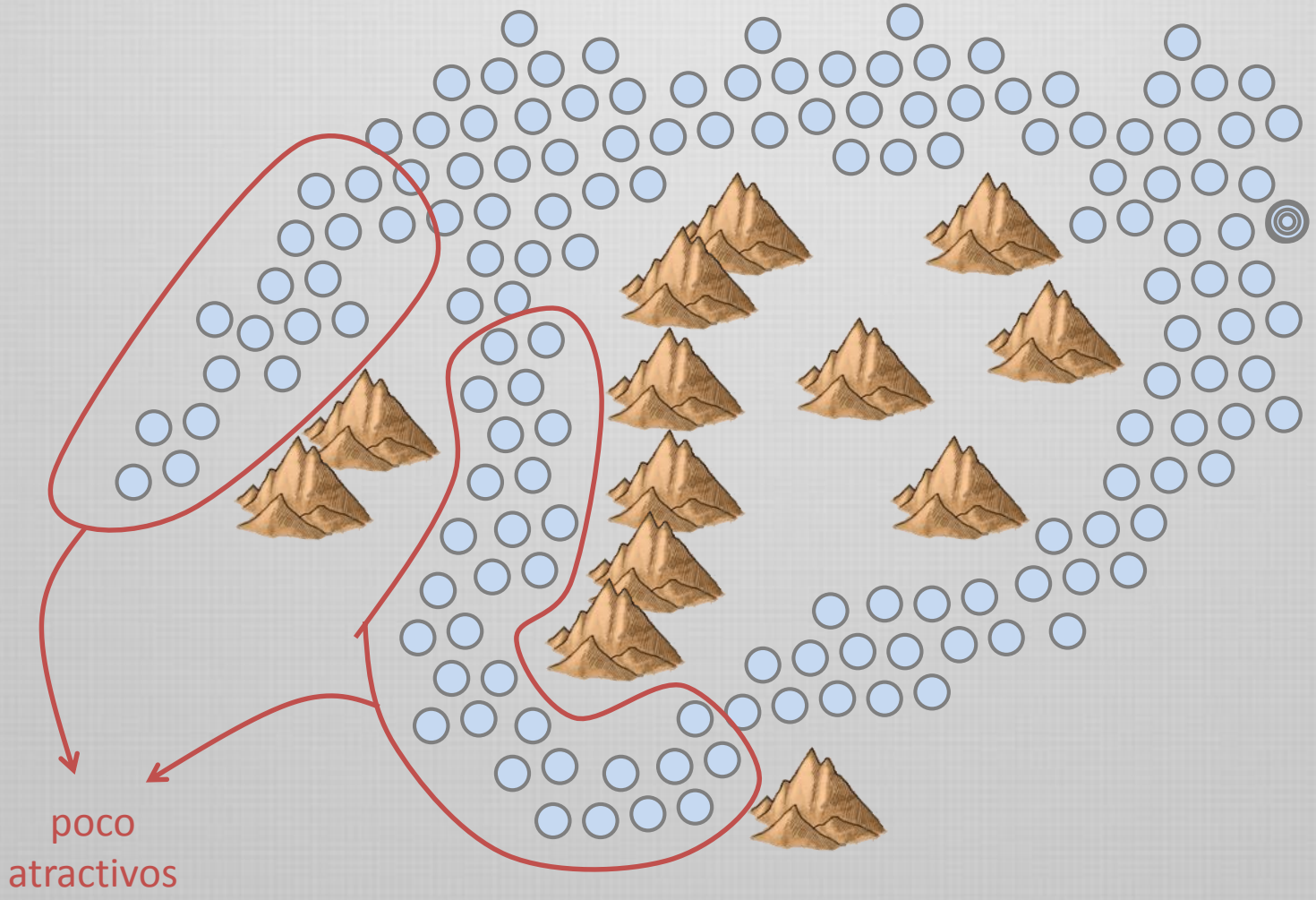
El algoritmo A* elige y expande nodos de *mejor* valor heurístico f()

Le dará preferencia a los nodos que

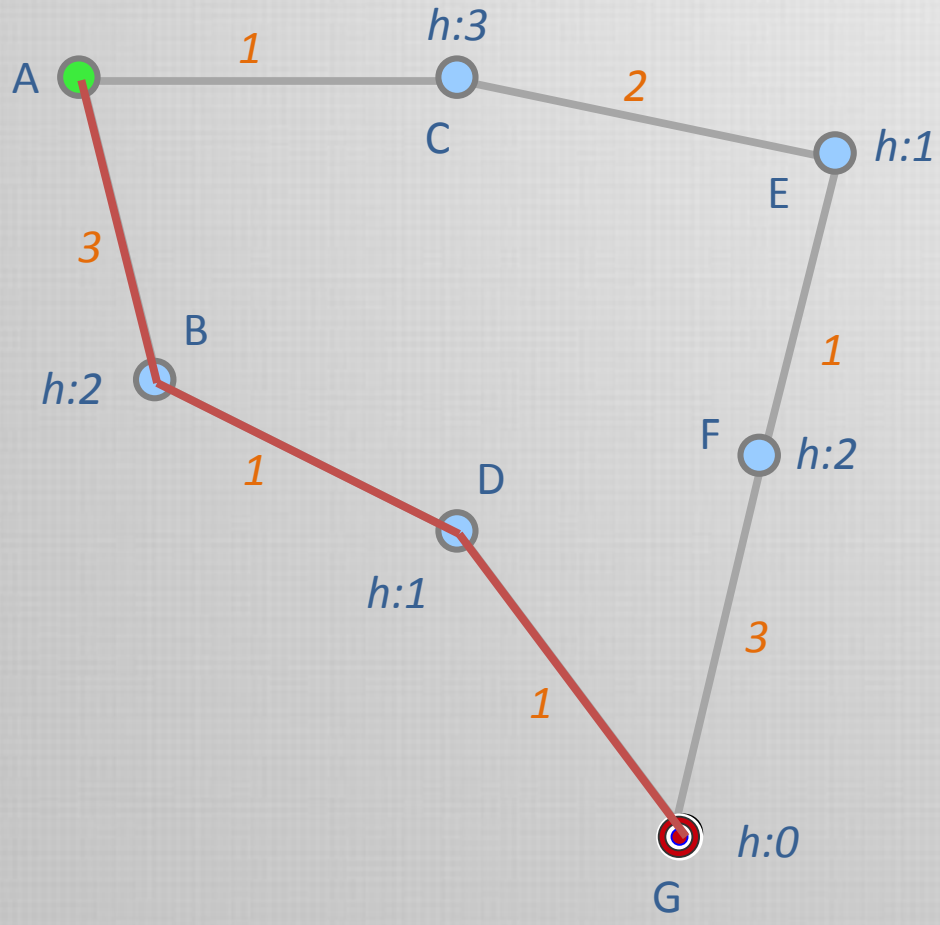
conforman un camino de bajo costo

son prometedores de acuerdo a la estimación hacia la meta

Pathfinding



Pathfinding



A	B [f = 3+2 = 5]
	C [f = 1+3 = 4]
AC	B [f = 3+2 = 5]
	E [f = 3+1 = 4]
ACE	B [f = 3+2 = 5]
	F [f = 4+2 = 6]
AB	F [f = 4+2 = 6]
	D [f = 4+1 = 5]
ABD	F [f = 4+2 = 6]
	G [f = 5+0 = 5]

ABDG