

Navegación de Peatones en Bosque de Obstáculos

Trabajo Práctico Final

Badi Leonel, Buchhalter Nicolás Demián y Meola Franco
Román

19 de julio de 2016

Grupo 3

Introducción

- Vamos a simular el trayecto de un peatón que viajará desde un punto inicial a un punto final
- Este agente está equipado con un **mecanismo de navegación** para evitar colisiones con los obstáculos fijos
- Utilizaremos el *Contractile Particle Model*¹
- Implementación propia (No se utilizaron plataformas de simulación peatonal)

¹ Gabriel Baglietto y Daniel R Parisi. "Continuous-space automaton model for pedestrian dynamics". En: *Physical Review E* 83.5 (2011), pág. 056117.

Objetivos

Definición de una heurística

Crear una heurística que permita ajustar **dinámicamente** el ángulo del vedor de la velocidad deseada en función de las posiciones de los obstáculos.

- Se implementó una versión modificada del **método de búsqueda informado A*** que utiliza una heurística definida
- La búsqueda de la ruta se realiza antes de la simulación ya que todos los obstáculos a evitar son fijos
- Se modificará el *Contractile Particle Model* para evitar colisiones en el transcurso la simulación

Estado del Arte

- *Steering Behavior*²
- *Contractile Particle Model*
- Métodos de búsqueda informados de Sistemas de Inteligencia Artificial (ITBA)

²Mat Buckland. "Chapter 3: How to Create Autonomously Moving Game Agents". En: *Programming Game AI by Example*, Wordware Publishing (2005).

Sistema

Parámetros del Agente

- $r_0 = 0,25m$
- $r_{min} = 0,15m$
- $r_{max} = 0,32m$
- $v_d^{max} = 1,55 \frac{m}{s}$

Sistema

Parámetros del *Contractile Particle Model*

- $\tau = 0,5s$
- $\beta = 0,9$

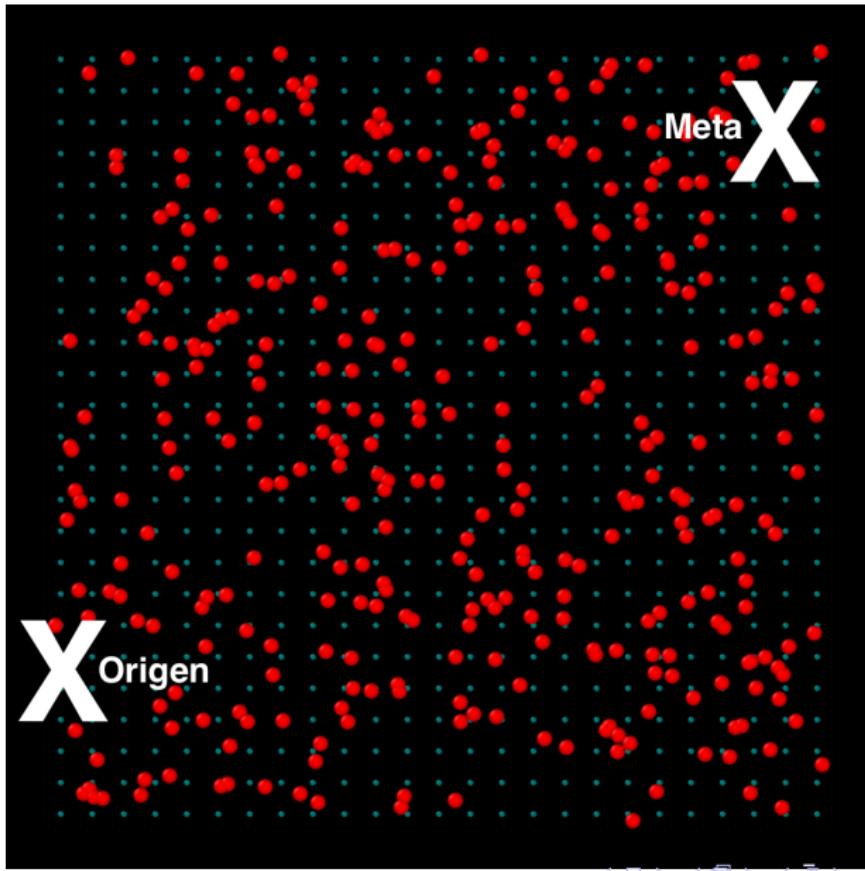
Sistema

Parámetros de la escena

- Habitación cuadrada de $25m$ de lado
- **300 obstáculos fijos** distribuidos aleatoriamente
- $r_{obstaculo} = r_0$
- La posición inicial del agente está en la esquina inferior izquierda
- La posición destino del agente está en la esquina superior derecha

Sistema

Ejemplo de Escena



Métodos de Búsqueda Informado

Introducción

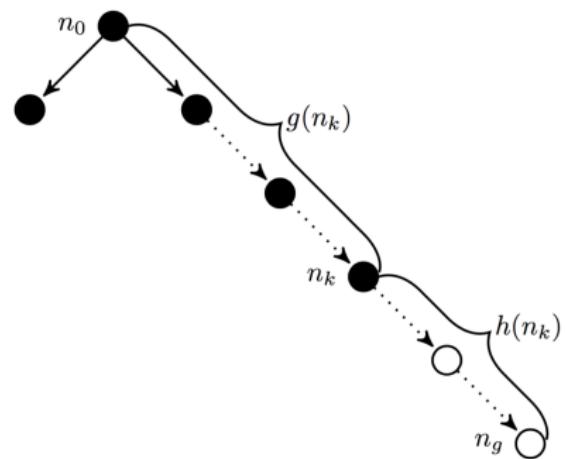
- Sólo utilizando el costo $g(n)$, la búsqueda a ciegas resulta muy ineficiente
- Es necesario agregar valores que estimen cuánto falta para alcanzar el objetivo
- Definimos a una **heurística** $h(n) =$ costo estimado de la ruta más barata que une el estado del nodo n con un estado meta
- Familia de algoritmos *Best First*

Métodos de Búsqueda Informado

Mínimo Costo de Ruta

$$f(n) = g(n) + h(n)$$

- $g(n)$: es el costo de la ruta que va del nodo inicial hasta n
- $h(n)$: estimación del mínimo costo de ruta que va de n al nodo objetivo
- $f(n)$: costo estimado de la solución más barata que pasa por n



Método de Búsqueda Informado

Algoritmo A*

- Actúa sobre un **grafo**
- El objetivo es encontrar el camino de **costo mínimo**
- Elige para expandir el nodo de menor f

Adaptación necesaria

Al actuar sobre un grafo el espacio de búsqueda debe ser **discreto**. Se requiere hacer una adaptación de la escena para poder discretizar posiciones de la habitación.

Algoritmo A*

- ① Crear un árbol de búsqueda, T_r , que sólo esté formado por el nodo inicial n_0 . Insertar n_0 en la estructura FRONTERA.
- ② Crear el conjunto EXPLORADOS, inicialmente vacío.
- ③ Si FRONTERA está vacía, el algoritmo termina con falla.
- ④ Extraer el primer nodo de FRONTERA e insertarlo en EXPLORADOS. Llamar a este nodo n .
- ⑤ Si n es el nodo objetivo, salir del algoritmo y devolver la solución, que está formada por el camino definido por los arcos que llevan de n a n_0 en el árbol T_r .
- ⑥ Expandir el nodo n y generar el conjunto de nodos sucesores M . Incorporar los elementos de M como sucesores de n en el árbol T_r , creando los arcos correspondientes entre n y cada uno de los nodos de M .
- ⑦ **Reordenar FRONTERA de forma que primero estén los nodos de menor f .**
- ⑧ Ir al paso 3.

Algoritmo A*

Comportamiento

- Una heurística es **admisible** si nunca sobreestima el costo real
- Si h es admisible $\Rightarrow f(n)$ nunca sobreestima el costo real de la mejor solución que pasa por n
- Con h admisible A* es **completo** y **admisible**: Garantiza encontrar el camino óptimo al objetivo

Adaptación

Utilización de *Waypoints*

Discretización de la escena

Waypoints: puntos (x, y) por donde el agente podrá desplazarse. Estos serán los **nodos** por donde iterará el método de búsqueda.

- Se decidió armar una **grilla** de *waypoints* sobre la habitación
- Los *waypoints* están separados unos de otros por una distancia W_d

Adaptación

Utilización de *Waypoints*

- Cada *waypoint* cuenta con una **ventana cuadrada de vecinos**
- W_v : longitud del lado de la ventana
 - $W_v = 3$: la ventana contiene a los 8 vecinos
 - $W_v = 5$: la ventana contiene a los 24 vecinos
- El conjunto de vecinos de la ventana del nodo n constituye a los nodos sucesores del mismo

Adaptación

Utilización de *Waypoints*

- Se utiliza A^* para obtener una lista de *waypoints* por donde el agente tiene que transitar hasta alcanzar el nodo meta
- La lista de *waypoints* obtenida se define como un conjunto de *sub-objetivos* que tiene que ir alcanzando el agente
- Así, para cierto momento dado, el versor velocidad del agente tiene como dirección a la línea que se forma entre la posición actual del agente y el primer sub-objetivo

Heurística

Distancia al objetivo

Heurística propuesta

$h(n)$ es la distancia real que se tiene al objetivo (distancia en línea recta al nodo meta)

- $h(n)$ es admisible
- $g(n)$ es la distancia recorrida desde el nodo raíz hasta n

Búsqueda limitada en profundidad

Para el algoritmo A^*

- El árbol de búsqueda tiene un **máximo de profundidad L**
- En el borde del límite se encuentra un nodo meta sub-óptimo
- Una vez alcanzado el sub-óptimo, **se reinicia la búsqueda desde allí**

Detección de colisiones

Poda del árbol de búsqueda del algoritmo A*

- Se genera un *raycast* desde un waypoint origen hacia otro waypoint destino
- Si el trayecto del *raycast* impacta sobre un obstáculo, entonces ese camino es **descartado** del árbol de búsqueda
- Así se evitan muchas colisiones innecesarias y caminos sin salida

Prevención de colisiones

Modificación del *Contractile Particle Model*

- El objetivo es evitar los choques del agente con obstáculos que tiene en sus inmediaciones
- Con una partícula de radio $r_p = 1,1r$ se calculan todos los posibles **solapamientos con obstáculos**
- Para cada solapamiento detectado se genera un versor director en dirección contraria a la del agente, para así evitar este obstáculo
- Todos los versores se suman y se obtiene un **versor summarizado** que representa la dirección a transitar para no colisionar con los obstáculos

Simulación

Variables relevantes de la simulación

- t [s]: Tiempo (en segundos) a visualizar.
- dt [s]: Tiempo (en segundos) del paso de la simulación.
- k : Relación entre cantidad de pasos simulados y escritos para visualización.

Simulación

Algoritmo de simulación

```
public void simulate(double t, double dt, int k){  
    writeFrame(0);  
    int framesWritten = 1;  
    double totalTimeSimulated = 0;  
    moveSystem(dt);  
    totalTimeSimulated += dt;  
    while(totalTimeSimulated < t){  
        for(int i = 0; i < k; i++){  
            moveSystem(dt);  
            totalTimeSimulated += dt;  
        }  
        writeFrame(framesWritten++);  
    }  
}
```

Código 1: Algoritmo de simulación

Simulación

Parámetros de la simulación

- $t < 100s$ donde la simulación termina cuando el agente alcanza el nodo meta
- $3 < k < 7$
- $dt = 0,01s$

Simulación

Visualización

- La simulación y la visualización son independientes
- El algoritmo de simulación escribe un archivo .tsv con los siguientes datos:
 - (x, y)
 - r
 - Color RGB para indicar las velocidades, donde R es la componente en el eje Y y G es la componente en eje X
- Por último, se carga en Ovito el archivo de salida .tsv para realizar la visualización

Gráfico de la distancia recorrida promedio

Para distintos valores de obstáculos fijos

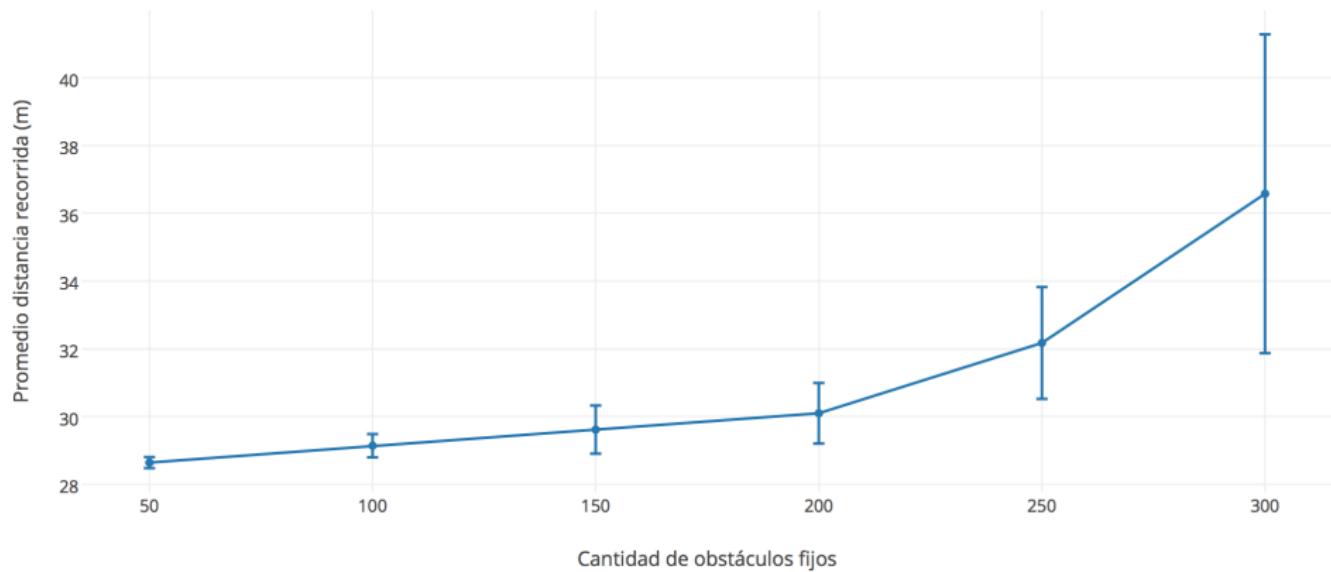
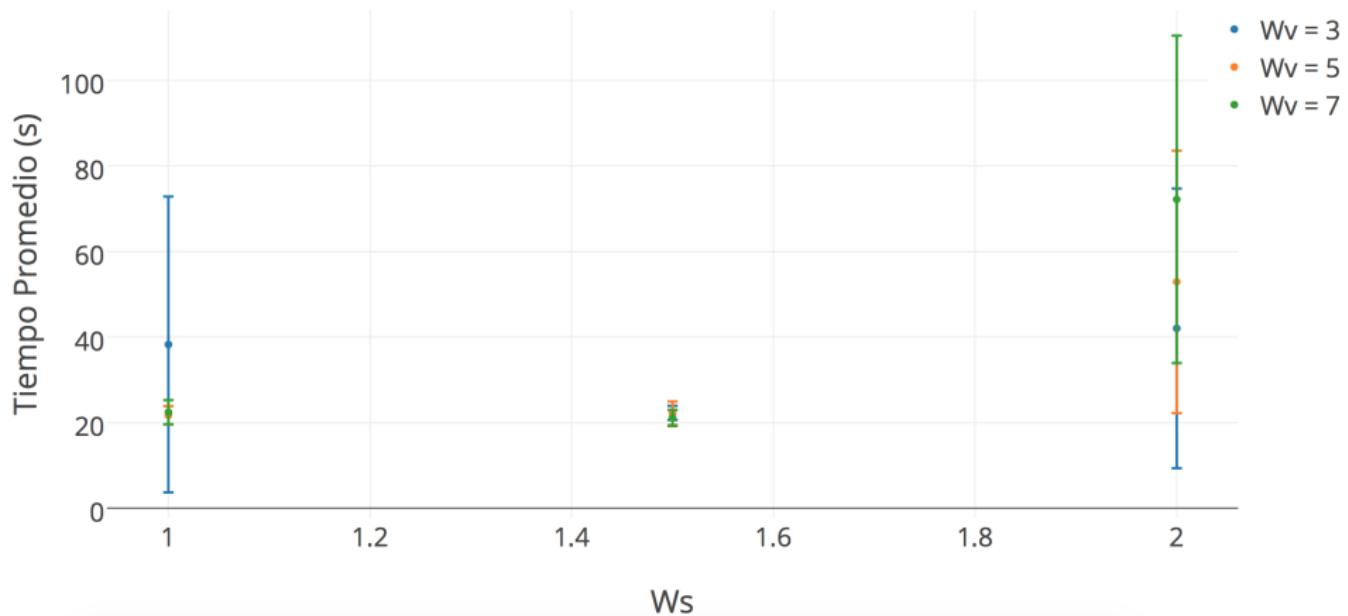


Gráfico del tiempo promedio de simulación

Para distintos valores de W_s y W_v



Animación de la simulación con 250 obstáculos fijos

$W_d = 1$, $W_v = 5$

Animación de la simulación con 25 obstáculos móviles

Conclusiones

- A valores mayores de W_v , se pueden encontrar caminos diagonales que, claramente, tendrán una distancia menor y por lo tanto serán considerados con mayor prioridad en el método de búsqueda informado.
- Se logró obtener un modelo capaz de navegar por una escena con una gran densidad de obstáculos obteniendo más de un 80 % de efectividad (cantidad de veces que el agente alcanza el goal)
- Este modelo de navegación es altamente personalizable, permitiendo modificar parámetros como la cantidad y posición de los waypoint así como también permitiendo implementar nuevos *steering behaviors*

Futuras extensiones

- **Definición de la heurística:** Los posibles choques con obstáculos fijos no sean causa de descarte del camino sino que solamente se pondere a ese trayecto con un costo mayor en el árbol de búsqueda.
- **Detección de colisiones:** Proyectando varios *raycast* con distintos radios. Así se aprovecharía mejor la variación del radio del modelo.
- **Otros steering behaviours:** Para que el agente tenga un movimiento más complejo. Por ejemplo, si se quiere que un grupo de agentes siga a un líder, se puede implementar el comportamiento *Follow the leader*.

Gracias