



---

---

---

---

---

---

---

### Terreno

- Cómo representar el entorno
- Necesario para razonar sobre él
- Muchos comportamientos dependen de él
  - Puntos de cobertura
  - Pathfinding
  - Flanqueos
  - Emboscadas

© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

### Necesidad

- Hay juegos en que es muy sencillo
  - Deportes: Fútbol, Baloncesto, Hockey
    - Siempre el mismo escenario
    - Simple de generar
    - Posibilidad de anotar
- Hay juegos en que es muy complicado
  - Mundo abierto

© Diego Garcés

20/04/2016

---

---

---

---

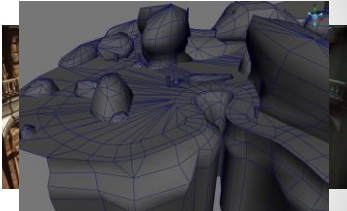
---

---

---

## Diferentes representaciones

- Representación visual
  - Millones de polígonos
  - Shaders geométricos, pixel
- Representación física
  - Generalmente más simplificada
  - Demasiado detalle en visual



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

## Física

- Complicado hacer la física estable
- Costoso hacer la física estable
- Complicado hacer consultas
  - Visibilidad (Line of sight): ¿Estoy viendo al jugador?
  - Disparo (Line of fire): ¿Puedo disparar al jugador?
  - Desde donde estoy y desde cualquier posición a la que puedo ir
  - Necesario para tomar decisiones
  - Ocupación
    - ¿Qué enemigos están en esta zona?
    - ¿Qué obstáculos me voy a encontrar en este área?

© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

## Diferentes representaciones

- Representación Lógica
  - Más simplificada que la física
  - Quizás con anotaciones
  - Relaciones entre los distintos elementos
    - Diferentes a la física y visual
  - Quizás con varios niveles

© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

# Grid



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

# Grid

- Problemas
  - Trade-off entre resolución y tamaño
    - Poca resolución (casilla grande)
      - No puedo representar detalles
      - Pasillos estrechos
      - Zonas angulosas o diagonales
    - Mucha resolución (casilla pequeña)
      - Muchas casillas
      - Muchas casillas con la misma información
  - Problemas en 3D
    - Muy costoso usar un grid 3D directamente

© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

# Grid



© Diego Garcés

20/04/2016

---

---

---


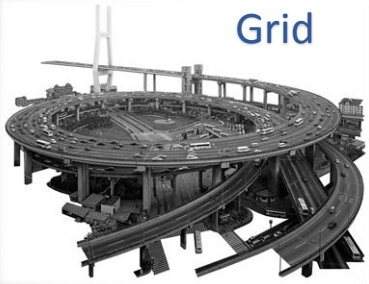
---

---

---

---

# Grid



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

# Waypoints

- Compuesto por puntos unidos entre sí



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

# Waypoints

- Traducción directa de un grafo
- Datos
  - Puntos
  - Conexiones a otros puntos
- Dos puntos estan conectados
  - Si puedo moverme de uno a otro
  - Tienen visibilidad entre ellos
- Muy sencillo
- No representa todo el terreno
  - Sólo se que puedo moverme de este punto a otros

© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

## Waypoints: Datos

- Pseudo – Código para representar los waypoints

```
struct Waypoint
{
    struct Connection
    {
        Waypoint* m_pTarget;
    }
    TerrainType m_terreno;
    Vect2 m_position;
    std::vector<Connection> m_connections;
};

std::vector<Waypoint> m_Waypoints;
```

---

---

---

---

---

---

---

## Waypoints: Uso

- El NPC se mueve de un punto a otro
- Útil para rutas de patrulla
- Busco el punto más cercano a mi posición
- Me muevo de un punto a otro
- Hasta que llego al punto más cercano al destino
- Posibilidad de variar las rutas
  - Diferentes conexiones
  - Diferentes puntos

---

---

---

---

---

---

---

## Waypoints: Ventajas

- Concepto muy sencillo
- Directamente traducido a datos para el pathfinding

---

---

---

---


---

---

---

### Waypoints: Problemas

- Sólo tengo información en los puntos
- Fuera de los puntos estoy ciego



© Diego Garcés 20/04/2016

---

---

---

---

---

---

---

### Waypoints: Problemas

- En zonas abiertas muchísimos puntos



- Muchas conexiones entre puntos
  - En un escenario típico veré muchos puntos desde muchos otros

© Diego Garcés 20/04/2016

---

---

---

---

---

---

---

### Waypoints: Problemas

- Normalmente editados a mano
- O al menos ajustados a mano
  - Limpieza de conexiones
  - Ajuste de posiciones
  - Separación de esquinas
  - Puntos estratégicos

© Diego Garcés 20/04/2016

---

---

---

---

---

---

---

# Waypoints: Problemas

- Difícil para NPCs de diferentes tamaños
  - Sólo tengo información de los puntos
  - Cuánto separar el waypoint?
  - Dejar todo a obstacle avoidance?



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

# Waypoints: Ejemplo



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

# Navigation meshes

- Representar espacio mediante polígonos conectados



© Diego Garcés

20/04/2016

---

---

---

---

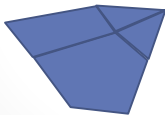
---

---

---

# Navigation meshes

- Generalización del grid
- Ya no son casillas cuadradas iguales
- Cada "casilla" es un polígono de n lados
- Conectados con otros mediante aristas



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

# Navigation meshes

- Convierten el escenario en un mapa Pseudo-2D



© Diego Garcés

20/04/2016

---

---

---

---

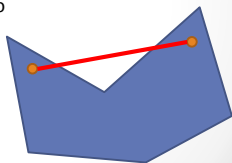
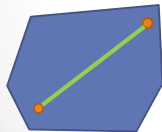
---

---

---

# Navigation meshes

- Tienen que ser convexos
- Cualquier punto dentro del polígono esta conectado con cualquier otro dentro del polígono
- Si estoy dentro del polígono puedo ir en línea recta a cualquier otro sitio del polígono



© Diego Garcés

20/04/2016

---

---

---

---

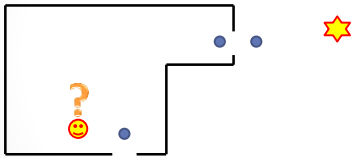
---

---

---



# Navigation meshes



© Diego Garcés

20/04/2016

---

---

---

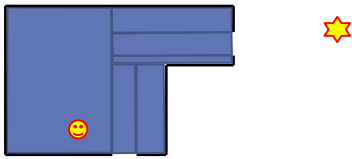
---

---

---

---

# Navigation meshes



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

# Navigation meshes

- Cada arista conecta con un polígono o ninguno
- Si es ninguno → No se puede pasar por esa arista
  - Sólo hay polígonos en las zonas pasables
- Límites del espacio navegable

© Diego Garcés

20/04/2016

---

---

---

---

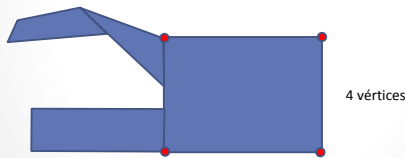
---

---

---

## Navigation meshes

- Mas sencillo cumplir:
- Una arista → Un polígono
- Si tenemos un caso como este:



© Diego Garcés

20/04/2016

---

---

---

---

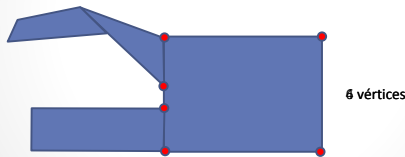
---

---

---

## Navigation meshes

- Se añaden vértices adicionales en intersecciones



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

## Navmesh: Estructura

- Pseudo Código de la estructura de datos

```
struct NavPolygon
{
    struct Edge
    {
        int m_verts[2]; // Indices de m_verts
        NavPolygon* m_pNeighbour;
    }
    TerrainType m_terreno;
    std::vector<Vect2> m_verts;
    std::vector<Edge> m_Edges;
    // Otros: centro, área, plano, etc
};

std::vector<NavPolygon> m_Navmesh;
```

© Diego Garcés

20/04/2016

---

---

---

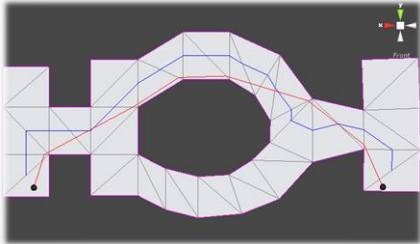
---

---

---

---

## Navmesh: Ejemplo



© Diego Garcés 20/04/2016

---

---

---

---

---

---

---

## Navigation meshes

- Etiquetar los polígonos
  - Pasable (si esta definido es que puedo pasar)
  - Tipo de terreno
    - Diferentes costes de pathfinding
    - Diferentes sonidos al caminar
    - Diferentes velocidades de movimiento

© Diego Garcés 20/04/2016

---

---

---

---

---

---

---

## Navmesh: Ventajas

- Representa todo el espacio navegable
  - Si un punto esta fuera del navmesh → No se puede pasar por allí
- Facilita ciertos cálculos
  - No es necesario recurrir a representación física
- Trabajo en 2D
- Fácil representación de varios niveles de altura
  - Puentes
  - Edificios de varias plantas

© Diego Garcés 20/04/2016

---

---

---

---

---

---

---

## Navmesh: Problemas

- Mapas grandes → Muchos polígonos
  - Segmentar el navmesh en zonas
  - Hacer streaming del navmesh
- Movimiento en 3D
  - Difífil extenderlo a movimiento 3D: volar, nadar
  - AAS Quake 3

© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

## Navmesh: Problemas

- Problemas cuando NPC deja el suelo
- Por ejemplo salto:



© Diego Garcés

20/04/2016

---

---

---

---

---

---

---

## Navmesh: Construcción

- Generalmente manual
  - Editor específico
  - Editor in-game
  - Exportador paquete 3D
- Procesos automáticos
  - Recast

© Diego Garcés

20/04/2016

---

---

---

---

---

---

---