Planificador STRIPS

Planificador STRIPS

INTRODUCCIÓN

Anteriormente se realizaron algoritmos de búsqueda para resolver problemas de ruta en laberintos. Estos algoritmos respondían bien ante estos problemas, pero no respondían ante una nueva implementación en el laberinto: las condiciones.

La única condición en los laberintos previos era llegar hasta la meta, pero ahora activar la meta tiene una serie de requisitos dependientes de otros items que se encuentren en el tablero.

Para ello se propone usar STRIPS, un algoritmo de resolución de problemas que nos permite atender a condiciones y propiedades sin dejar de hacer uso de los algoritmos que ya usábamos de búsqueda para posibilitar el movimiento.

En Madrid, a mayo de 2019

Jesús Fermín Villar Ramírez
Diseño y Desarrollo de Videojuegos, ESNE

Título
Planificador STRIPS
Autor
Jesús Fermín Villar Ramírez
Asignatura
Ingeniería del conocimiento: IA
Dicente
Luis Peña Sánchez
Grado
Diseño y Desarrollo de Videojuegos, ESNE
Fecha
Mayo 2019
Fotografía de portada
https://www.pexels.com/photo/white-multi-story-building-928208/

ALGORITMO USADO

El algoritmo que se ha usado para resolver el problema ha sido STRIPS.
Mediante una estructura de operadores y propiedades representamos la información del mundo de la siguiente forma:

Operador "GoTo"
El operador principal de este problema consistirá en ir hacia una celda en concreto.
Al aplicar este operador, el controlador usará algoritmos de búsqueda (en este caso BFS) para encontrar la ruta idónea para llegar a ese punto.

Propiedades "Has"
La única propiedad existente en este problema es sobre si el personaje tiene cada uno de los items del tablero.

Algunos operadores poseen precondiciones. Esto es, propiedades que debe tener el personaje para poder ejecutar ese operador.
A su vez, todos los operadores dotan de nuevas propiedades al personaje; añadiendo o eliminando propiedades.

Con estos datos se genera el algoritmo. Un algoritmo que recorra todos los operadores para identificar aquel necesario para la victoria y; desde dicho algoritmo, hacer una expansión regresiva analizando las precondiciones de dicho operador. Por cada precondición, el algoritmo evalúa si es una propiedad a conseguir o no (si el personaje ya cumple esa precondición o no). Si necesita conseguirla, volverá a comprobar los operadores para identificar cuál dota al personaje de esas propiedades.

Así, el controlador hará este ciclo tantas veces como sea necesario hasta llegar a su estado inicil. El resultado de esta operación es una lista de operadores que ejecutar en un determinado orden.

Cada vez que se completa la aplicación de un operador, el controlador vuelve a realizar la búsqueda por si hubiese adquirido, de alguna otra forma, una nueva propiedad o si hubiese cambiado el mundo.

```
private void Analize()
{
    currentPlan.Clear();

    while (desired_tags.Count != 0)
    {
        foreach (var _operator in all_operators)
        {
            var owned_tags = new List<string>();
            foreach (var property in own_properties) owned_tags.Add(property.GetTag());

            var iterator = 0;
            var added_conditions = _operator.GetAddedProperties();

            for (iterator = 0; iterator < _operator.GetAddedProperties().Count; iterator++)
            {
                if (desired_tags.Contains(added_conditions[iterator].GetTag()) && !currentPlan.Contains(_operator))
                {
                    currentPlan.Add(_operator);
                    foreach (var condition in _operator.GetPreconditions()) if (!owned_tags.Contains(condition.GetTag())) desired_tags.Add(condition.GetTag());
                    foreach (var added in _operator.GetAddedProperties()) if (desired_tags.Contains(added.GetTag())) desired_tags.Remove(added.GetTag());
                }
            }
        }

    }
    currentPlan.Reverse();
}
```

Figura 1: Implementación algoritmo de STRIPS

Para facilitar el visionado del funcionamiento del algoritmo en el editor de Unity, se ha incorporado, en el algoritmo de búsqueda de pathfinding, un sistema de tintado de celdas que pinta las celdas expandidas cada vez que realiza una consulta al algoritmo de búsqueda.

Al haber incorporado el algoritmo de búsqueda BFS se observa que para celdas lejanas expande casi todo el tablero. Como este es un entorno pequeño no supone un cambio importante, pero sería recomendable usar otros algoritmos de búsqueda como A*, por ejemplo.

El controlador funciona correctamente y solo bloquea Unity en aquellos casos en los que no hay una solución aparente. En el resto de casos funciona bien y adecuada.

Para evitar falsos positivos se ha quitado la condición que havía que al pisar la casilla de meta se saliese del laberinto y esta se ha incorporado a solo en aqueloos casos que la lista de propiedades por obtener esté vacía; es decir, que ya haya cumplido todos sus objetivos.

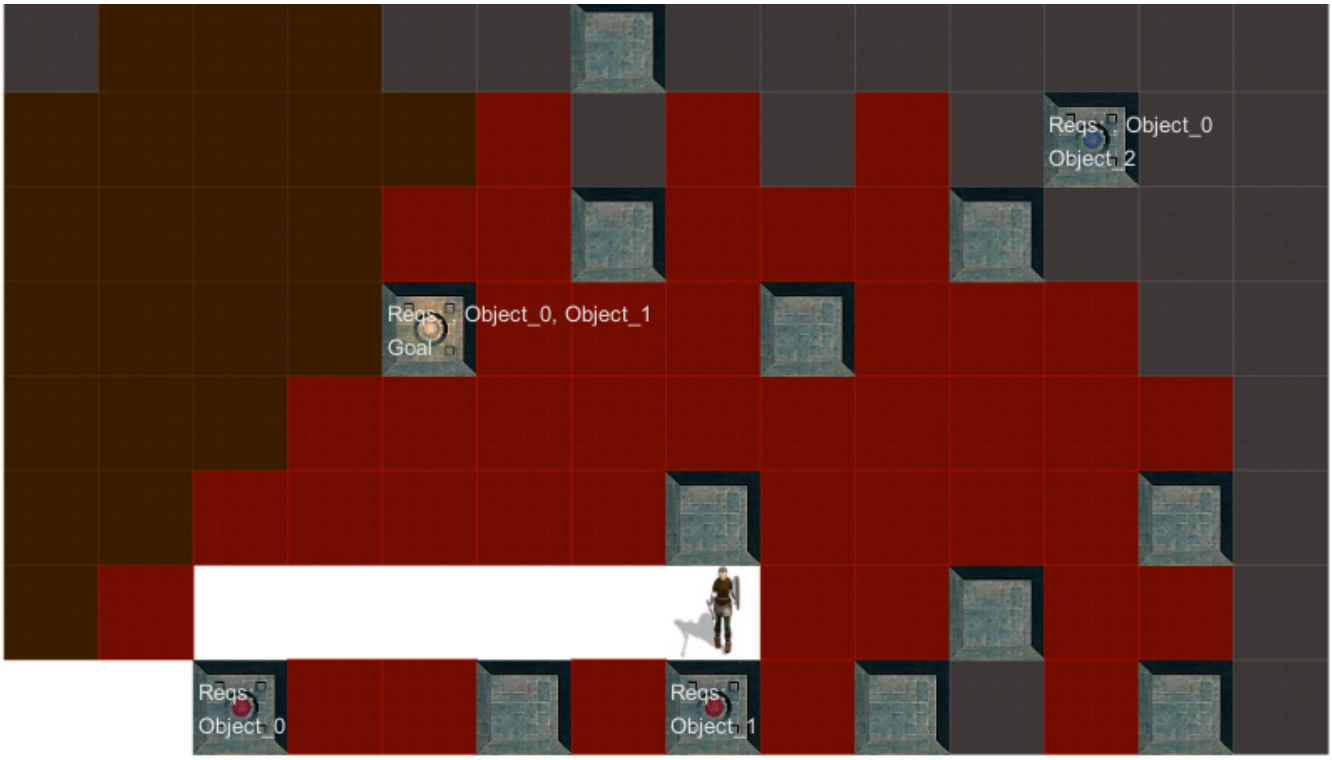

Figura 2: Implementación algoritmo de STRIPS en el editor