UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Algoritmos de búsqueda 3D

Presentado por Víctor Pérez Esteban
en Universidad de Burgos — 22 de mayo de 2017
Tutor: José Francisco Diez Pastos

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. José Francisco Diez Pastos, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Víctor Pérez Esteban, con DNI 71279579A, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 22 de mayo de 2017

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Conceptos teóricos

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Algunos conceptos teóricos de LaTeX[1].

3.1. Algoritmo A*

A* es un algoritmo de búsqueda informada del tipo primero el mejor, que usa una función de evaluación para elegir hacia que nodo expandirse desde un nodo inicial hacia un nodo final.

Para representar el espacio de búsqueda del algoritmo, se usa una cuadrícula donde cada nodo puede representar un espacio donde es posible desplazarse o un obstaculo que es inalcanzable.

La función F() de evaluación calcula el coste de cada nodo, con lo que se eligen los nodos con menor coste para llegar al destino a través del camino óptimo. Esta función está formada por dos funciones a su vez. Una función G() que calcula el coste del camino seguido desde el nodo inicial a ese nodo concreto, y una función H() que hace un estimación del coste del camino desde ese nodo al nodo final o meta.

Al algoritmo comienza desde el nodo inicial explorando los nodos adyacentes o sucesores, cual es de menor coste. Un nodo ya explorado, es decir del que ya se ha buscado sus sucesores, se manda a la lista de nodos cerrados. Con los sucesores se forma una lista de nodos abiertos o por explorar, de la cual se elige el de menor coste para ser el siguiente en ser explorado, hasta

[1]Créditos a los proyectos de Álvaro López Cantero: Configurador de Presupuestos y Roberto Izquierdo Amo: PLQuiz

que se alcance el nodo meta o no queden más nodos por ser explorados. Si al explorar un nodo esta ya se encontraba en alguna de las listas de nodos abiertos o cerrados, se actualizarán los valores de los nodos al de menor coste encontrado.

Heurística

Al algoritmo A* es completo, lo que significa que encontrará un camino hasta la meta siempre que este exista. Además, para que sea admisible, que significa que siempre encontrará un camino óptimo, su función H() tambíen debe ser admisible.

Una función H() es admisible siempre y cuando no sobreestime el coste del camino desde un nodo hasta la meta. Por ejemplo se puede considerar que el camino desde un nodo hasta la meta será la línea recta.

La admisibilidad del A* trae consigo un gran coste computacional debido al gran número de nodos explorados. Para mejorar la eficiencia podemos dar pesos a las funciones G() y H(), de tal forma que si damos mas valor a G() la búsqueda se expandirá en anchura buscando el camino, mientras que si damos más valor a H() se expandirá más rápido acercandose a la meta.

Pseudocódigo

Además de secciones tenemos subsecciones.

Subsubsecciones

Y subsecciones.

3.2. Referencias

Las referencias se incluyen en el texto usando cite [**?**]. Para citar webs, artículos o libros [**?**].

3.3. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

Figura 3.1: Autómata para una expresión vacía

3.4. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

3.5. Tablas

Igualmente se pueden usar los comandos específicos de LATEXo bien usar alguno de los comandos de la plantilla.

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas. No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.

4.1. Programas usados

Unity

Unity un motor para juegos usado principalmente para desarrollar viejuegos y simulaciones. Que sea un motor para juegos quiere decior que está formado por varios motores a su vez, como el motor gráfico y el motor de físicas.

Unity proporciona un editor donde se puede crear escenas junto con un gran número de herramientas, ejemplos y modelos para crear escenarios tanto 2D y principalment 3D.

Tiene una licencia de uso por pasos de ingresos obtenidos por el contenido creado con el. En la versión Personal, su puede usar libremente siempre que los ingresos generados por el contenido usado no supere los 100 dólares anuales, y es la versión de la licencia usada.

Para el proyecto probamos de forma descente las distintas versiones de Unity para linux en un xUbuntu 16.04, debido a que las últimas versiones producían errores que cerraban el editor al iniciarlo o al poco tiempo despues

de abrirse. La primera versión estable que funcionaba en el equipo de desarrollo fue la versión 5.3.6f1, y ha sido la usada para su realización.

Texmaker

Para la realización de la memoria se ha usado latex con el editor texmaker. Texmaker es un editor libre con licencia GPL.

Texmaker incluye varias herramientas como el corrector ortográfico o el visor de pdf que facilitan la creación de los documentos.

La versión usada ha sido la 4.4.1.

SmartGit

SmartGit es un gestor gráfico para git que soporta github y que facilita la realización de commits, push y pull de los repositorios del proyecto.

Tiene una licencia no comercial que permite el uso de forma gratuita a desarrolladores de código abierto, profesores, estudiantes, desarrollos no comerciales y también organizaciones sin ánimo de lucro.

La versión que hemos usado ha sido la 17.0.3

GitHub y ZenHub

La plataforma elegida para los repositorios ha sido github, principalmente por su compatibilidad con Zenhub que es la herramienta de gestión de proyectos que hemos usado.

Zenhub es una herramienta que permite la planificación y control del proyecto. Es un complemento de firefox que se integra con github y añade funciones como las boards para manejar el control de las issues y las milestones del proyecto, y que permite ver los gráficos del progreso y burnouts que se ha realizado.

La versión utilizada ha sido la 2.34.2

Aspectos relevantes del desarrollo del proyecto

5.1. Nav Mesh

Una nav mesh es un conjunto de polígonos, triángulos normalmente, que se usan en los motores gráficos 3D para representar la zona o camino recorrible de un agente. Se crean a partir de los objetos estáticos de una escena, que son los obstaculos, donde no se puede desplazarse. El resto de la zona de la escena será zona recorrible donde nos podemos desplazar.

En relación por ejemplo al A*, sería el equivalente a la matriz que reprsenta el mapa donde se bucará los nodos a explorar.

Está formada por los vértices de los polígonos, y si un punto esta dentro del área que forman entonces ese punto es recorrible. Esto es mas realista en un escenario en tres dimensiones debido a que permite un movimiento menos discretizado, al contrario que las parrillas de casillas que se adecuan más a una representación en dos dimensiones. Y si consideramas los vértices los nodos sucesores, resulta más óptimo computacionalmente, ya que el número de nodos (vértices) a explorar es menor, debido a que no tenemos que tener en cuenta todos los puntos intermedios hasta llegar al vértice, si no unicamente si el vértice es visible o alcanzable desde el nodo actual.

Hemos usado una nav mesh para generar de forma automática el mapa para el A*. En Unity esto se puede hacer de dos formas. En versiones anteriores a la versión 5.6 solo es posible generarlo desde el editor antes del tiempo de ejecución. A partir de la versión 5.6, que permite acceder a las herramientas del editor en tiempo de ejecución, se puede construir a través de NavMeshBuilder, dando mucha más libertad a la hora de crear mapas de forma dinámica o interactiva. Lamentablemnte no se pudo usar la versión 5.6 y la versión mñas actual que funcionba en el equipo de desarrollo fue la 5.3.6.

La clase NavMesh de Unity tiene métodos que podemos usar como RayTrace, que lanza un rayo desde un vector a otro y devuelve si son visibles dentro de la NavMesh, es decir, si existe una línea recta entre ellos que los una dentro del camino recorrible. Este métodol lo hemos usado para decidir si un nodo sucesor es válido o no. Al usarlo encontramos un bug que producía que a veces no se encontrará el camino o bucles infinitos a recorrer las listas. Esto es debido a que puede ocurrir que no devuelva lo mismo dependiendo del orden o sentido en el que se le indiquen los vectores entre los que lanzar el rayo. Por ejemplo RayTrace(vector1, vector2) puede ser visible pero RayTrace(vector2, vector1) no. Para solucionarlo decidimos que para que fuera un sucesor valido debie ser visible en los dos sentidos.

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.