WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Path Finding Algorithm Visualization for Acropolis Institute of Technology and Research

[1]Albin Thomas, [2]Ashutosh Singh Yadav, [3]Ayush Soni, [4]Chaman Mandloi, [5]Pir Mohammad

Department of Computer Science and Engineering, Acropolis Institute of Technology and Research, Indore, M.P, 453771

ABSTRACT:

In this paper we present you a Pathfinding Visualizer, aptly because it does what it says; it finds a path from a source to a destination. This project is based on graph theory. In this project we have tried to use graph theory and various graph algorithms which we have studied in the chapter of data structures and algorithm and that is all in the backend. The front end is designed using Pygame this project reflects the application of data structures.

Keywords: Visualization , Dijkstra's, A*, DFS, Python, e-Learning tool, shortest path algorithms

1. Introduction

Pathfinding or pathing is the plotting, by a computer application, of the shortest route between two points. It is a more practical variant on solving mazes. This field of research is based heavily on Dijkstra's algorithm for finding the shortest path on weighted graph.

Pathfinding is closely related to the shortest path problem, within graph theory, which examines how to identify the path that best meets some criteria (shortest, cheapest, fastest, etc) between two points in a large network.

At its core, a pathfinding method searches a graph by starting at one vertex and exploring adjacent nodes until the destination node is reached, generally with the intent of finding the cheapest route. Although graph searching methods such as a breadth-first search would find a route if given enough time, other methods, which "explore" the graph, would tend to reach the destination sooner. An analogy would be a person walking across a room; rather than examining every possible route in advance, the person would generally walk in the direction of the destination and only deviate from the path to avoid an obstruction, and make deviations as minor as possible.

Path-finding using BFS i.e. Breadth First Search

BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph thus exploring the nodes (nodes which are directly connected to source node). You must then move towards the next-level nodes.

Fig.1: Path-finding using BFS i.e. Breadth First Search

Path-finding using A* algorithm

A* algorithm is a searching algorithm that searches for the shortest path between the initial and the final state. It is used in various applications, such as maps. In maps the A* algorithm is used to calculate the shortest distance between the source (initial state) and the destination (final state).

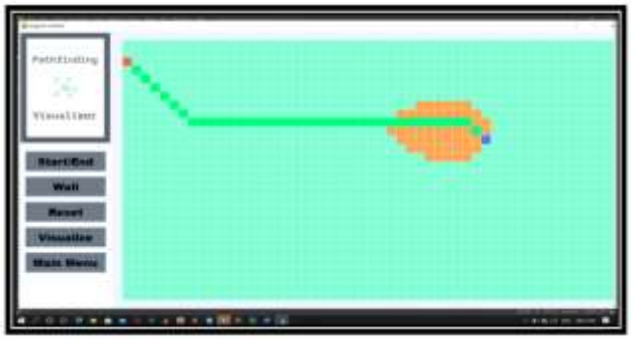

Fig.2: Path-finding using A* Algorithm

Path-finding using Dijkstra searching Algorithm

Dijkstra's algorithm published in 1959 and named after its creator Dutch computer scientist Edsger Dijkstra, can be applied on a weighted graph. The graph can either be directed or . One stipulation to using the algorithm is that the graph needs to have a weight on every edge.

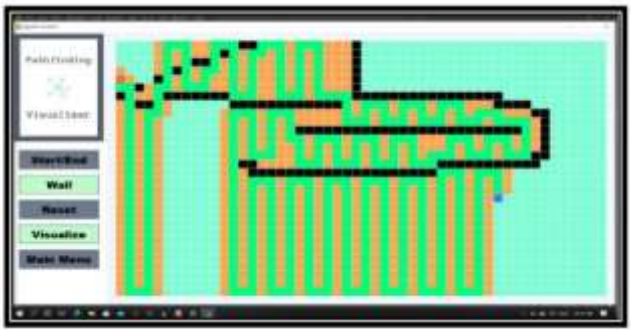

Fig.3: Path-finding using Dijkstra searching algorithm

Path-finding using DFS searching Algorithm

The data structure which is being used in DFS is stack. The process is similar to BFS algorithm. In DFS, the edges that leads to an node are called discovery edges while the edges that leads to an already visited node are called block edges.

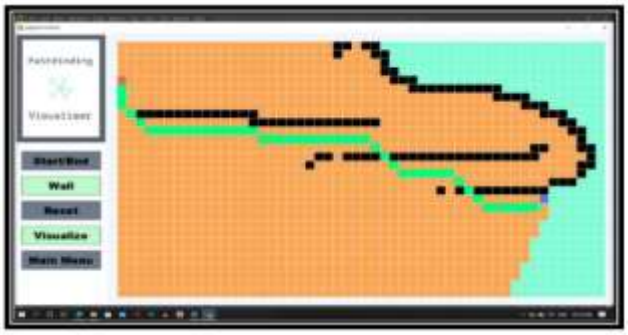

Fig.4: Path-finding using DFS searching algorithm

BACKGROUND

We have implemented the project using Pygame and python.:-

Pygame

• **Pygame** is a set of python modules designed for writing video games. adds functionality on top of the excellent SDL library. This allows you to create fully featured games and multimedia programs in the python language.

• **Pygame** is highly portable and runs on nearly every platform and operating system.

• **Pygame** itself has been downloaded millions of times.

• **Pygame** is free. Released under the LGPL license, you can create open source, freeware, shareware, and commercial games with it.

• Multi core can be used easily:-

With dual core common, and 8 core cheaply available on desktop systems, making use of multi core allows you to do more in your game. Selected functions release python GIL, which is something you can do from C code.

• Uses optimized C and Assembly code for core functions. C code is often 10-20 times faster than python code and assembly code can easily be 100x or more times faster than python code.

• Truly portable

Supports Linux (comes with most main stream distributions), Windows (95, 98, ME, 2000, XP, Vista, 64-bit Windows, etc), Windows CE, BeOS, Mac OS, Mac OS X, FreeBSD, Net BSD, Open BSD, BSD/OS, , IRIX, and QNX. The code contains support for Amiga OS, , Atari, AIX, OSF/Tru64, RISC OS, Symbian OS and OS/2, but these are not officially supported. You can use it on hand held devices, game consoles and the One Laptop Per Child (OLPC) computer.

• It's Simple and easy to use. Kids and adults make shooter games with . is used in the OLPC project and has been taught in essay courses to young kids and college students. It's also used by people who first programmed in z80 assembler or c64 basic.

• Comes with many operating systems. Just an apt-get, emerge, , or install away. No need to mess with installing it outside of your operating system's package manager. Comes with binary system installers (and uninstallers) for Windows or Mac OSX. Does not require setup tools with even to install.

• You control your main loop. You call functions, they don't call your functions. This gives you greater control when using other libraries, and for different types of programs.

• Does not require a GUI to use all functions. You can use from a command line if you want to use it just to process images, get joystick input, or play sounds.

• Small of code. It does not have hundreds of thousands of lines of code for things you won't use anyway. The core is kept simple, and extra things like GUI libraries, and effects are developed separately outside of .

• Modular. You can use pieces of separately. Want to use a different sound library? That's fine. Many of the core modules can be initialized and used separately.

PYTHON:

Python is an interpreted, high- level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and programming. Python is often described as a & quot ;batteries included & quot; language due to its comprehensive standard library

Python was created in the late 1980s, and first released in 1991, by Guido van as a successor to the ABC programming language. Python 2.0, released in 2000, introduced new features, such as list comprehensions, and a garbage collection system with reference counting, and was discontinued with version 2.7 in 2020. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains , a free and open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and development .It currently ties with Java as the second most popular programming language in the world.

METHODOLOGY:

Our Project, Pathfinder-visualizer is based on Python ; we have used basic tools of to develop the Skeleton of our project (like frames, grids, buttons, graphics etc.)

We have created a interface constructed on the account of the below shown buttons are features of the mentioned library.

We have used various path-finding algorithms (like BFS, DFS, A* And DIJKSTRA) to find the path between the starting and the ending node.

With the help of we, have also provided an option to the user for building the walls or maize in the presented grid!

Attributes-

• **RED NODE** is staring node.

• **BLUE NODE** is node.

• **ORANGE NODES** define the nodes we have to check to get the path

• **GREEN NODES** show the path between start node and ending node.

Simply,our method involves the implementation of various Path-finding Algorithms like BFS, DFS, A* And Dijkstra to find out the path between source and the destination node along with game development tools to build a simple but interactive visualizer.

CONCLUSION AND FUTURE WORK

In this work, we presented Pathfinder, a technique for the visual analysis of paths in large multivariate graphs .Our query-based approach allows users to search for paths between a specified start and end, using various path-finding algorithms like Dijkstra,A*, BFS and DFS. The immediate display of intermediate results allows for early query refinements and speeds up the analysis process. Paths can be judged and ranked holistically, taking topology, attributes, and grouping structure into account.

Our technique addresses all requirements save one: the exploration and detail analysis of paths. Aggregating similar paths and showing their basic structure could give analysts a better overview of the variation of paths. Combined with revealing details on demand, this could be an alternative approach to rank the alternative paths for tackling large lists of paths. Aggregations could also be driven by a user-defined combination of properties, such as topology, attributes, and sets. We believe this to be a fruitful direction for future research.

References

[1]https://clementmihailescu.github.io/Pathfinding-Visualizer.

[2] https://www,pygame.org/wiki/about.

[3] [1] “Map of Winnipeg, Manitoba,” February 2013. [Online]. Available:

[4] https://maps.google.ca/

[5] [2] R. Mahony, P. Pounds, and P. Croke. (2006, December) Modelling and

[6] control of a quad- rotor robot. In the Proceedings of the Australasian

[7] Conference on Robotics and Automation. Auckland, New Zealand. Accessed:

[8] September2012.[Online].Available:http://www.araa.asn.au/acra/acra2006/

[9] papers/paper 5 26.pdf

[10] [3] I. Sa and P. Croke. Vertical infrastructure inspection using a quadcopter

[11] and shared autonomy control. Brisbane, Australia. Accessed: September

[12] 2012.[Online].Available:https://wiki.qut.edu.au/download/attachments/10