What is a good strategy for constructing a directed graph for a game map (in Python)?

Asked 15 years, 9 months ago Modified 14 years, 2 months ago Viewed 1k times

9

I'm developing a procedurally-generated game world in Python. The structure of the world will be similar to the MUD/MUSH paradigm of rooms and exits arranged as a directed graph (rooms are nodes, exits are edges). (Note that this is *not* necessarily an acyclic graph, though I'm willing to consider acyclic solutions.)

To the world generation algorithm, rooms of different sorts will be distinguished by each room's "tags" attribute (a set of strings). Once they have been instantiated, rooms can be queried and selected by tags (single-tag, tag intersection, tag union, best-candidate).

I'll be creating specific sorts of rooms using a glorified system of template objects and factory methods--I don't think the details are important here, as the current implementation will probably change to match the chosen strategy. (For instance, it would be possible to add tags and tag-queries to the room template system.)

For an example, I will have rooms of these sorts:

```
side_street, main_street, plaza, bar, hotel, restaurant, shop, office
```

Finally, the question: what is a good strategy for instantiating and arranging these rooms to create a graph that might correspond to given rules?

Some rules might include: one plaza per 10,000 population; `main_street` connects to `plaza`; `side_street` connects to `main_street` or `side_street`; `hotel` favors `main_street` or `plaza` connections, and receives further tags accordingly; etc.

Bonus points if a suggested strategy would enable a data-driven implementation.

python **graph** **procedural-generation**

Share

Improve this question

Follow

edited Mar 4, 2009 at 19:00

asked Mar 4, 2009 at 14:48

David Eyk

12.5k ● 12 ● 68 ● 106

2 Answers

Sorted by: Highest score (default)

7

First, you need some sense of Location. Your various objects occupy some amount of coordinate space.

You have to decide how regular these various things are. In the trivial case, you can drop them into your coordinate space as simple rectangles (or rectangular solids) to make locations simpler to plan out.

If the things are irregular -- and densely packed -- life is somewhat more complex.

Define a Map to contain locations. Each location has a span of coordinates; if you work with simple rectangles, then each location can have a (left, top, right, bottom) tuple.

Your Map will need methods to determine who is residing in a given space, and what's adjacent to the space, etc.

You can then unit test this with a fixed set of locations you've worked out that can all be dropped into the map and pass some basic sanity checks for non-conflicting, adjacent, and the like.

Second, you need a kind of "maze generator". A simply-connected maze is easily generated as a tree structure that's folded up into the given space.

The maze/tree has a "root" node that will be the center of the maze. Not necessarily the physical center of your space, but the root node will be the middle of the maze structure.

Ideally, one branch from this node contains one "entrance" to the entire space.

The other branch from this node contains one "exit" from the entire space.

Someone can wander from entrance to exit, visiting a lot of "dead-end" locations along the way.

Pick a kind of space for the root node. Drop it into your Map space.

This will have 1 - *n* entrances, each of which is a sub-tree with a root node and 1 - *n* entrances. It's this multiple-entrance business that makes a tree a natural fit for this structure. Also a proper tree is always well-connected in that you never have isolated sections that can't be reached.

You'll -- recursively -- fan out from the root node, picking locations and dropping them into the available space.

Unit test this to be sure it fills space reasonably well.

The rest of your requirements are fine-tuning on the way the maze generator picks locations.

The easiest is to have a table of weights and random choices. Choose a random number, compare it with the weights to see which kind of location gets identified.

Your definition of space can be 2D or 3D -- both are pretty rational. For bonus credit, consider how you'd implement a 2D-space tiled with hexagons instead of squares.

This "geometry" can be a **Strategy** plug-in to the various algorithms. If you can replace square 2D with hexagonal 2D, you've done a good job of OO Design.

Share
Improve this answer
Follow

edited Mar 4, 2009 at 19:21

answered Mar 4, 2009 at 15:41

S.Lott
391k ● 82 ● 517 ● 788

-1: The OP is asking about a non-coordinate-space graph approach, which is used in many games quite successfully (and allows more irregular interconnections, which can be quite useful). You describe a totally different (not necessarily better) approach. – Carl Meyer Mar 4, 2009 at 17:59

Also, a tree is not the right data structure for this application. The OP said graph, and he meant graph. A tree is just an acyclic graph, and you don't want the acyclic restriction in this case. – Carl Meyer Mar 4, 2009 at 18:07

@Carl Meyer: "The OP is asking about a non-coordinate-space graph approach" Good to know. I wonder how you know that? Any words or phrases in the question that indicate this? I couldn't find any. What did I miss? – S.Lott Mar 4, 2009 at 18:17

@Carl Meyer: "you don't want the acyclic restriction in this case" Good to know. I wonder how you know that? Any words or phrases in the question that indicate this? – S.Lott Mar 4, 2009 at 18:18

It's true, I left the "acyclic" out of "directed graph" intentionally, though I suppose I'm willing to consider a DAG if it would simplify my life. – David Eyk Mar 4, 2009 at 18:57

2

Check out the discussions on The MUD Connector - there are some great discussions about world layout and generation, and different types of coordinate / navigation systems in the "Advanced Coding and Design" (or similar) forum.

Share Improve this answer Follow

answered Mar 4, 2009 at 22:20

Andy Mikula
16.8k ● 4 ● 33 ● 39

Good resource, but I'd appreciate some direct links. – David Eyk Mar 4, 2009 at 22:53