# Unit 3 – Lesson 6. Introduction to Pathfinding in Game Development

**Aim:**

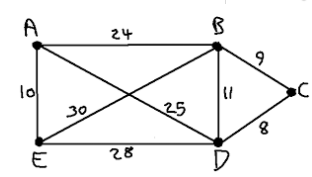
* How do we design path finding in a game?

**Objectives:** After the lesson, students should be able to:

* Obtain understanding of graphs, edges, nodes
* Obtain understanding of costs in path finding
* Comparing way points, grids, and navigation mesh (Reference: <http://jceipek.com/Olin-Coding-Tutorials/pathing.html>)

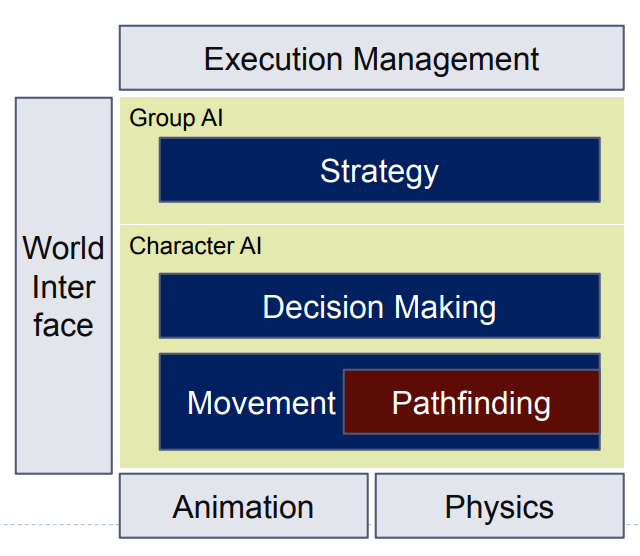
**CLASS PROCEDURE:**

***Do Now:*** Consider the famous traveling sales man problem: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?



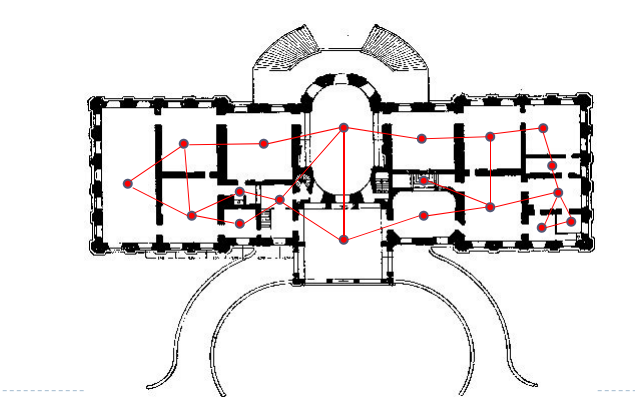
***Discussion / Presentation:***

1. Why do we need to study the pathfinding algorithms?



1. What is pathfinding?

* Pathfinding does not work directly on Geometry
* Simplification: Abstraction of movement possibilities in a graph



* **Nodes: Important places ! Sometimes just rooms, sometimes different places in a single room**
* **Edges: Connections through which we can travel**
* **Weights: Costs of traveling through a certain connection**
* **General assumption for developing algorithms: Weights are positive numbers**

1. What is the definition of graphs in mathematics and in game AI?

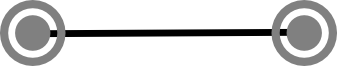
* In game AI (and in mathematics), graphs are a series of *nodes* connected by *edges*.

1. What is a node? What is an edge?

* The simplest way to think of these terms in the context of pathfinding AI is that each node in a graph represents a possible place a character could be and each edge indicates that it is possible for the character to move between the nodes it connects.
* Each edge has an associated *cost*, which represents the effort it takes for a character to move from one node to the next. When we talk about pathfinding, we're really talking about finding a path between two nodes in a graph, ideally minimizing the cost of the path.



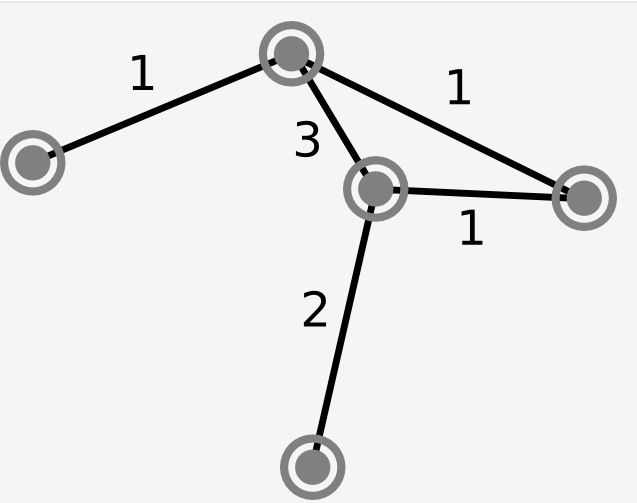
This is a node. It represents a place a character can be.



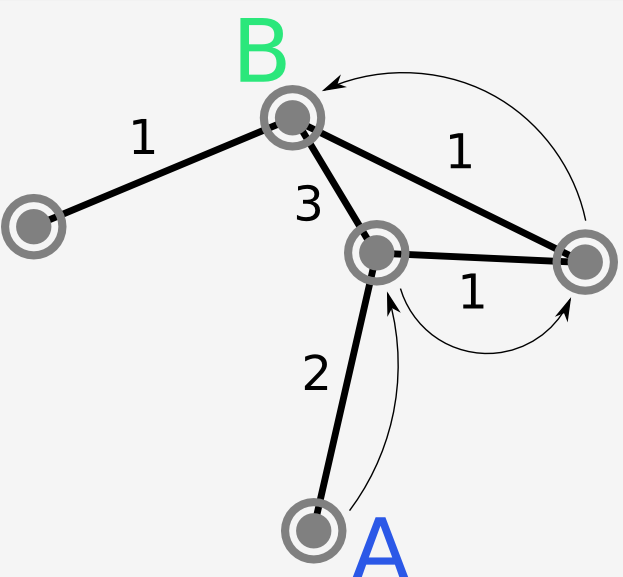
**Characters can only move between nodes via edges.**

1. A graph is a network of nodes. Moving between nodes is costly.

The edges we'll be looking at are bi-directional; characters can walk between them at will. More complicated edges might be one way only. For example, a character might jump down from a platform but be unable to jump back up. Don't worry, though, if you ever run into that kind of situation, the techniques we'll talk about can be applied just as easily.



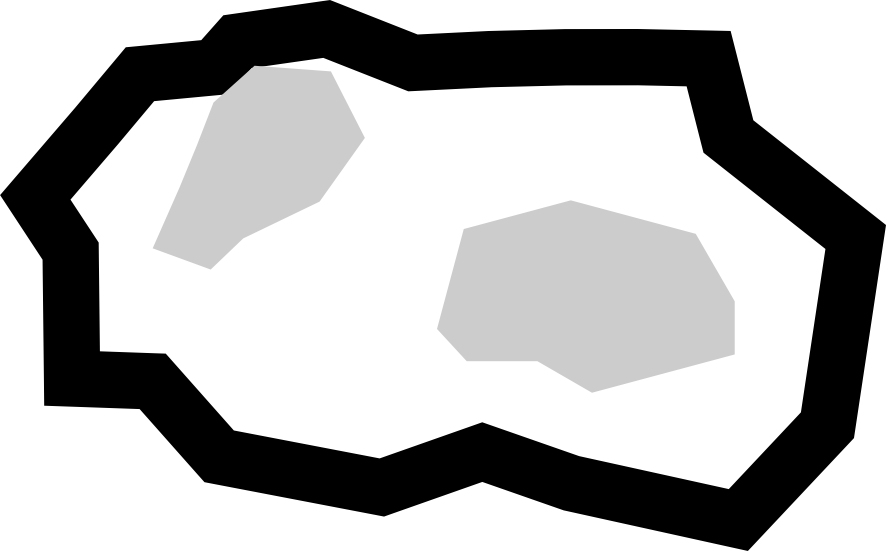
The cheapest path from A to B takes 3 steps and has a total cost of 4 = 2 + 1 + 1.



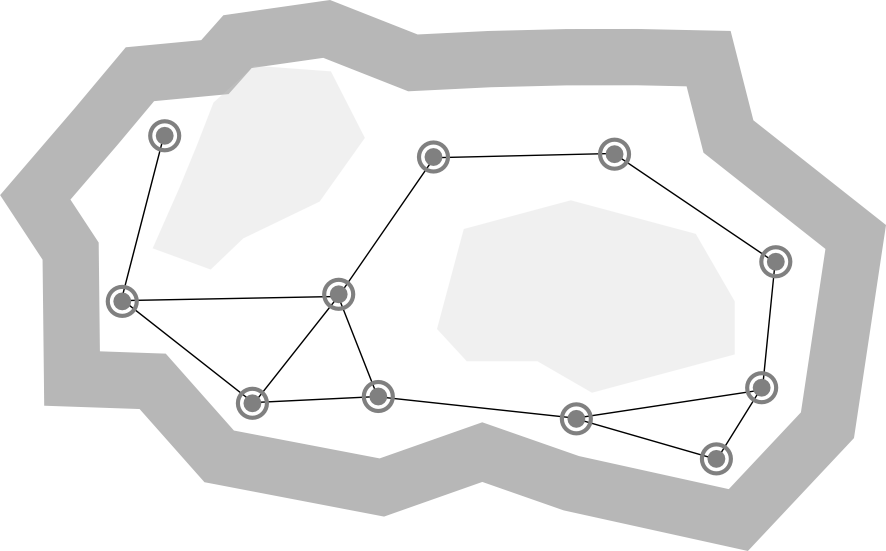
1. Some of the common video game path finding models:

4.1) Way points:

* Waypoint representations of space are easy to create and think about, but they [are not very good for pathfinding](http://www.ai-blog.net/archives/000152.html). Although they use very little memory, they result in inefficient, unrealistic paths.
* The main idea behind waypoint systems is to place a small number of nodes and edges in the game world so that characters can find their way around static obstacles. Unfortunately, if characters always stay on paths between waypoints, their behavior will be unrealistically constrained. However, if they leave the waypoint network, they can get stuck easily.



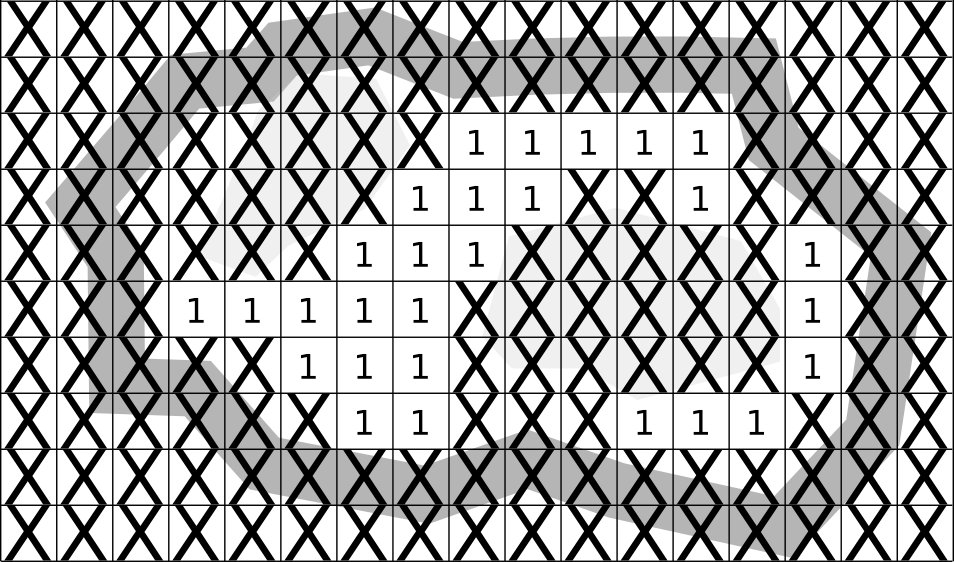
A rough island world.



Waypoints around an island. Each waypoint corresponds directly to a node. The costs would just be the length of the connections between waypoints. Note that some regions are unreachable.

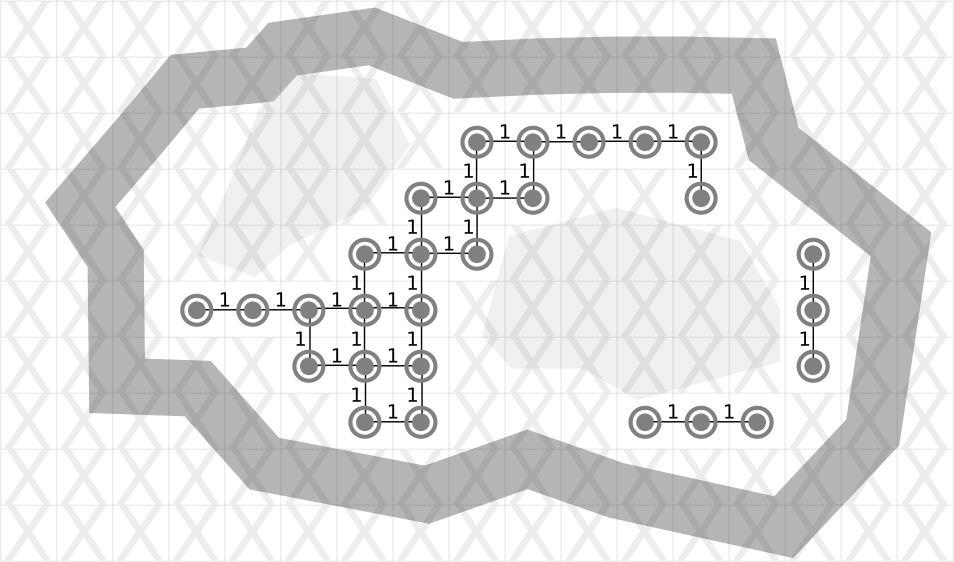
4.2) Grids:

* If you look for an introductory tutorial on path planning, chances are you'll find one that uses a grid. Grids work really well for small environments, especially for games in which characters move on a grid already. More complex environments require finer grids, which can take up an infeasible amount of memory for large worlds.
* You might have seen a representation of a grid like this:



* The numbers in each grid square represent the difficulty or "cost" of going to that square from an adjacent square. The rough grid makes many areas unreachable. A higher resolution grid would use more memory but would represent the world more effectively.

What you might not have realized is that a grid like this is just a more compact way to represent a very dense graph:

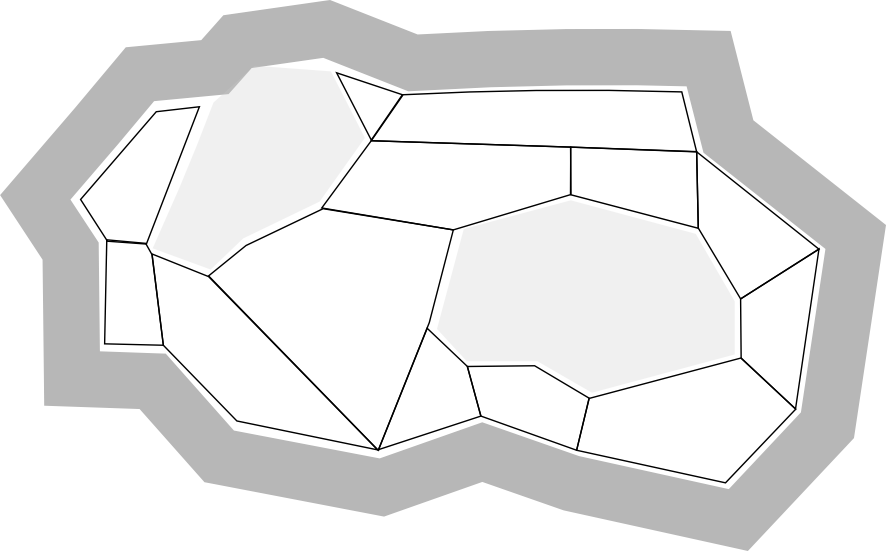


A grid is just a graph!

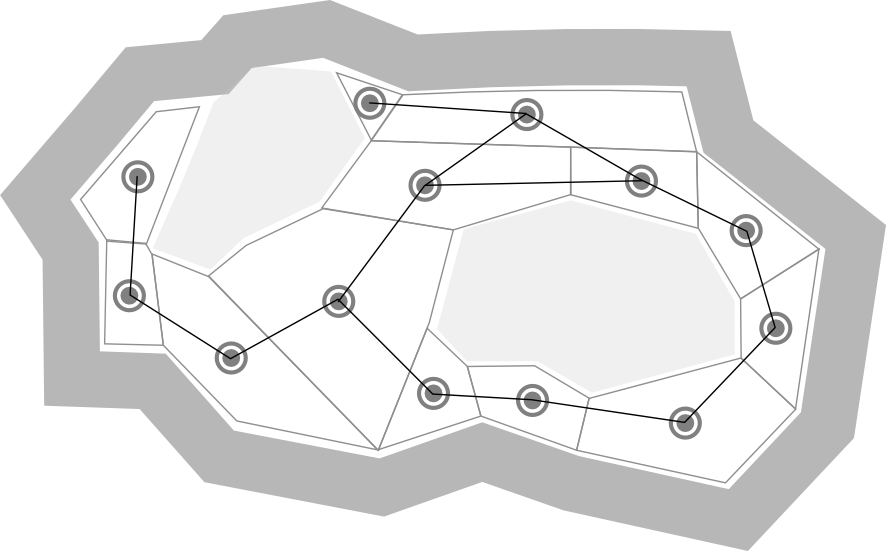
* Grids don't seem to be used much for large, modern games. However, cutting-edge AI researchers are [finding uses for them](http://digestingduck.blogspot.com/2010/03/local-navigation-grids.html) for very localized navigation in conjunction with other techniques such as navigation meshes.

4.3) Mesh Navigation:

* Navigation meshes are an efficient way to represent walkable regions in a game world. A common implementation relies on connected convex polygons.



* Navigation meshes efficiently describe walkable regions in the world.
* Just like in the grid representation, each polygon is a node and each edge shared by two polygons represents a connection between nodes.



* A Navigation Mesh is also just a graph!

***Pair – sharing Activity:***

Explore the sample AI pathfinding games and projects.

***Pair – sharing Activity:***

Continue working on your Maze Runner