Truly, the best way to build a high-level pathfinding system is to make it as generic as possible with known and isolated customization points. This is a perfect time to either buy, or dust off those design pattern books. Be ready to utilize patterns such as factories, strategies, prototypes, and policies. Isolate the customization points, because those will be frequently modified, and we don't want tons of special case code floating around the engine.

4.3 Why High-Level Pathfinding?

Imagine we have a world that contains over a million tiles. In this world, there is a battle raging in the North at the walls of an enemy city as seen in Figure 4.1. Our heroic army is trying to take the walled in city by force, but needs reinforcements. Let's help our army by moving a siege ram, a man-at-arms, and naval galley north to the battle.

Figure 4.1. A world map, with tile markings.

A million or more tiles is a great distance to navigate, especially with obstacles such as forests and towering enemy walls. Odds are that during a battle, CPUs are busy with animations, projectiles, physics, AI, and graphics. The effort of calculating detailed paths for an army across a huge map will adversely affect game performance; however, if our world was only a few thousand tiles, we could achieve this without any noticeable impact on the frame rate.

To accomplish this we need to reduce the search space to navigate our maps. High-level pathfinding achieves this through a several-phase process. First, our preprocess phase must identify each tile in the world as being of one, and only one, path type. With those types defined, we need to analyze the world and build up a knowledge base of terrain information before the game begins, which we accomplish by performing terrain analysis [3]. The second phase involves using the data stored in our world knowledge base to find a fuzzy, non-detailed path between two points using an algorithm such as A*. Our final phase involves the actor refining the fuzzy path to plan the exact route to their destination.

Sound like more than a few days of work? It should. Implementing high-level pathfinding can consume a significant amount of development time, but is well worth the effort.

4.4 Preprocess Phase

The preprocessing phase has two main parts—design and terrain analysis. In the design portion of this phase, we focus on identifying unique path tile types. This step is done without any coding, and it relies on full knowledge of actors and their movement rules. Designers should be prepared to define every actor rule such as prohibiting submarines from approaching shore lines, or declaring that only ninjas can climb walls. A rule only needs to be added only if it has a significant affect on an actor. For example, if movement on grass and movement on sand are the same for all actors in terms of pathfinding, then there is no need to define them as different region types. If, however, sand is illegal for a unicycle, then it needs to be its own path type.

The last part of this phase is terrain analysis, and this is where the creation of path regions occur. For an in-depth explanation of terrain analysis and it's many uses, it's strongly recommended that "Terrain Analysis in an RTS—The Hidden Giant" [3] is read before beginning any actual coding.

Design Time

Prior to coding, it's crucial to categorize each tile into a unique pathfinding type. For example, a water tile is rarely just a water tile. A given water tile may be best classified as being part of a river, bay, deep ocean, or coral reef. To identify these regions, we determine if all actors have the same restriction for traversing or avoid-