Trevonne Bridges

Chandler Garret

Nathan Jodoin

Salma M Omary

SimCity Project Design Document

**Part 1 – Functionality**

// Insert Diagrams and Some Text Here

**Part 2 – Data Storage**

At runtime, the program will read in a series of characters from a provided file. Each character will be translated into a zone object child, and then stored in a two-dimensional vector. The two dimensional vector will be the only ful copy of each zone object, and all other data structures will retain pointers to the 2D vector addresses for each object.

In order to calculate remote adjacencies, it is most efficient for each node to retain a list of direct adjacencies, otherwise known as local adjacencies. Each zone object will have a vector data member of size 8 populated with zone pointers. Index 0 represents the top left corner adjacency, and each subsequent index represents a clockwise rotation around the the node.

Each cycle, commercial and industrial nodes need to probe surrounding commercial and industrial zones to draw on workers and goods. Calculating adjacencies at each update is computationally expensive. A series of adjacency lists for each node, calculated once at runtime, wil allow for the simulation to run more efficiently. As workers and goods may travel via road, a local adjacency list will not suffice. Each industrial and commercial node requires a list of distantly adjacent residential nodes which are capable of providing workers. Closer workers are preferred, so the list should be sorted by distance along the available path or paths. This may be accomplished naturally with breadth first search. Similarly, as commercial nodes also receive goods locally and by road, an additional list of industrial adjacencies is required.

In order to avoid instances of multiple state, our program will access all of our data from the top level through a series of ordered function calls. In order to facilitate this functional approach, there will be a minimum of three linked lists, one for each type of populated zone. During to update and display loop of the simulation, the lists will be the primary means by which our zone objects are accessed. For example, pollution, which affects populated zones, may be update through a function which iterates through the list of each industrial zone, and then for each zone updates the adjacent nodes, moving outward until the appropriate depth has been reached. The functional list sorting will be by population from highest to lowest. Each functional working list will be populated with zone object pointers, pointing at each object in the original two-dimensional vector.

**Part 3 – File Organization**