CAMPS

Andrew, Charley, Dale, Jian, Justin, Ryan


“Design for speed!”

Customer Requests

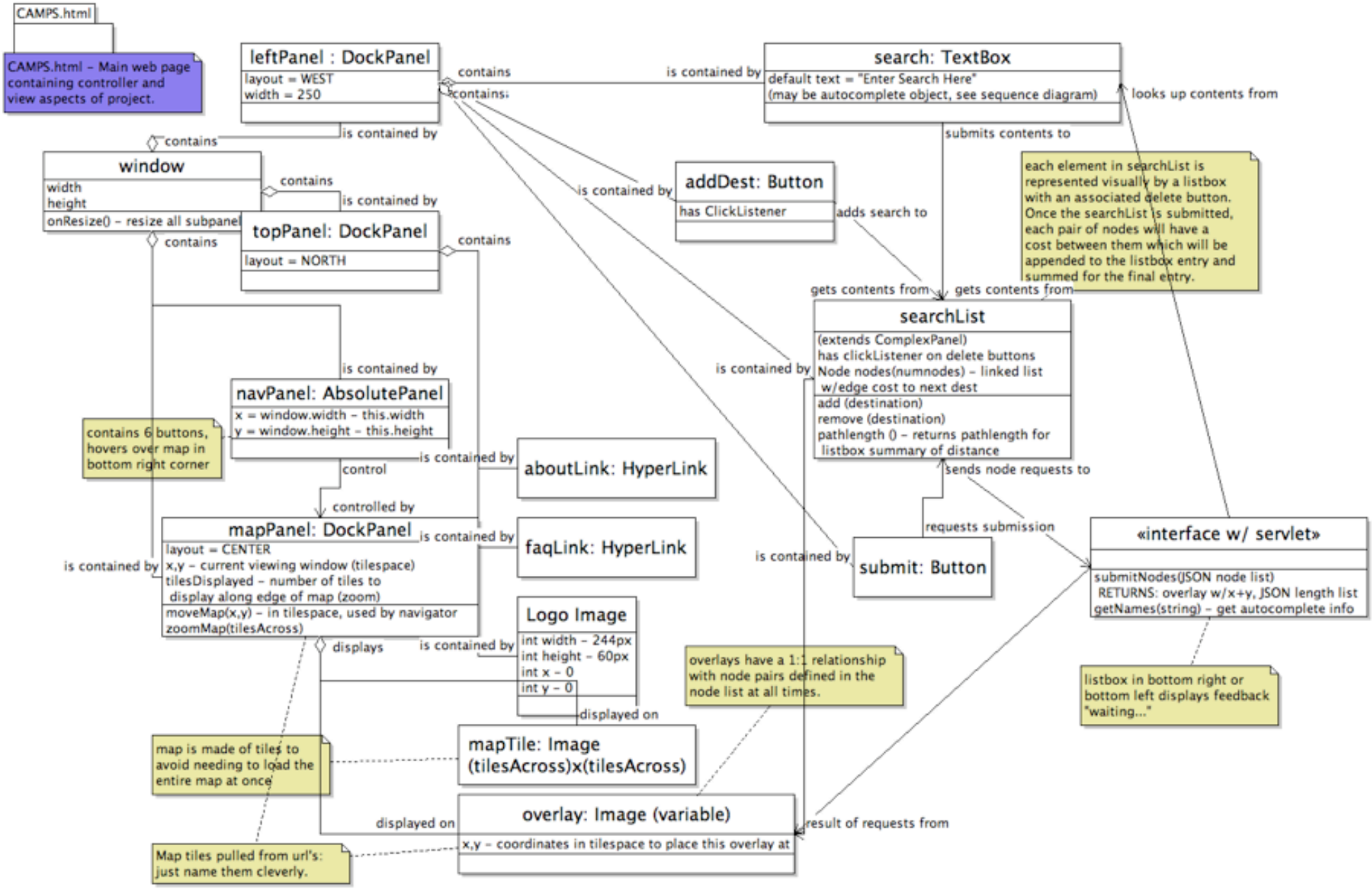
- Design for speed
  - Store the graph in memory to decrease response time
  - Find which search algorithm is faster and use it (Dijkstra or A*)

Zero Feature Release

- Prototype Database up and running with small data set for testing
  - Data will consist of a small test set of nodes, buildings, and edges
  - Data will not include path type or transportation method
- Map image segments rendered and spliced up for retrieval by UI
- Servlet is running on Tomcat platform:
  - Can accept incoming JSON requests and decrypt into internal data structure for analysis
- Partially-developed UI:
  - Search Box allows building a list of destinations
  - Submit button formats node list using JSON and queries servlet
  - Map loads intelligently (i.e. entire 8mb thing doesn't all load on start)
  - Panning is supported (not necessarily zooming)
  - UI is browser-safe and robust: support resizing, etc.
  - Links to primitive (possibly empty) faq and about
  - Style sheets developed and general site look and feel determined

Full Feature Release

- UI will draw multi-destination routes, where the paths between different destinations will appear in different colors.
- Can handle different path types and transportation types. The UI will allow the choosing of a transportation type (i.e. walk, bike)
- Algorithm will distinguish between different types of nodes (buildings, doors, normal) and will choose routes based on cost.
  - Quick Route Search
  - Lowest Cost Route Search
- Data will be more complete, more nodes, all paths.
- Fully customizable destination list with autocomplete (intelligently suggesting).
- Full support for zooming and panning by tiles.
- Mature, consistent looking interface.


CAMPS.html
CAMPS.html - Main web page containing controller and view aspects of project.
leftPanel : DockPanel
layout = WEST
width = 250
contains
is contained by
search: TextBox
default text = "Enter Search Here"
(may be autocomplete object, see sequence diagram)
looks up contents from
submits contents to
window
width
height
onResize() - resize all subpanel
topPanel: DockPanel
layout = NORTH
addDest: Button
has ClickListener
adds search to
each element in searchList is represented visually by a listbox with an associated delete button. Once the searchList is submitted, each pair of nodes will have a cost between them which will be appended to the listbox entry and summed for the final entry.
gets contents from
searchList
(extends ComplexPanel)
has clickListener on delete buttons
Node nodes(numnodes) - linked list w/edge cost to next dest
add (destination)
remove (destination)
pathlength () - returns pathlength for listbox summary of distance
navPanel: AbsolutePanel
x = window.width - this.width
y = window.height - this.height
contains 6 buttons, hovers over map in bottom right corner
control
controlled by
aboutLink: HyperLink
faqLink: HyperLink
mapPanel: DockPanel
layout = CENTER
x,y - current viewing window (tilespace)
tilesDisplayed - number of tiles to display along edge of map (zoom)
moveMap(x,y) - in tilespace, used by navigator
zoomMap(tilesAcross)
sends node requests to
requests submission
submit: Button
«interface w/ servlet»
submitNodes(JSON node list)
RETURNS: overlay w/x+y, JSON length list
getNames(string) - get autocomplete info
listbox in bottom right or bottom left displays feedback "waiting..."
Logo Image
int width - 244px
int height - 60px
int x - 0
int y - 0
displays
displayed on
overlays have a 1:1 relationship with node pairs defined in the node list at all times.
map is made of tiles to avoid needing to load the entire map at once
mapTile: Image (tilesAcross)x(tilesAcross)
overlay: Image (variable)
x,y - coordinates in tilespace to place this overlay at
result of requests from
Map tiles pulled from url's: just name them cleverly.

This is a sequence diagram for server side control flow when a user send a HttpRequest for finding routs for multiple buildings.