**Project Short Description**  
Michael Arnold

* Problem Description: A group of friends want are having a Game Night and are trying to decide what to have for dinner. Some in the group are willing to go out if they go to a restaurant they like, while others could go for a delivery service. The Intelligent Agent will use domain knowledge about restaurants and the preferences of each person present and will find the solution that satisfies the maximum number of people. It will then give the shortest distance path to the goal restaurant.
* Proposed Solution and Implementation Details:

• The Agent will use the A\* search because it is optimal.

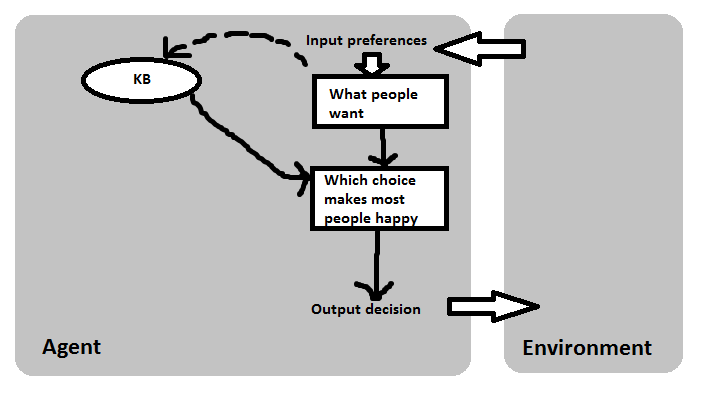
• Two (2) heuristics (one admissible and another inadmissible):   
 1.Straight Line Distance. (Admissible)  
 2. Straight Line Distance plus the number of nodes left to visit. (Inadmissible)

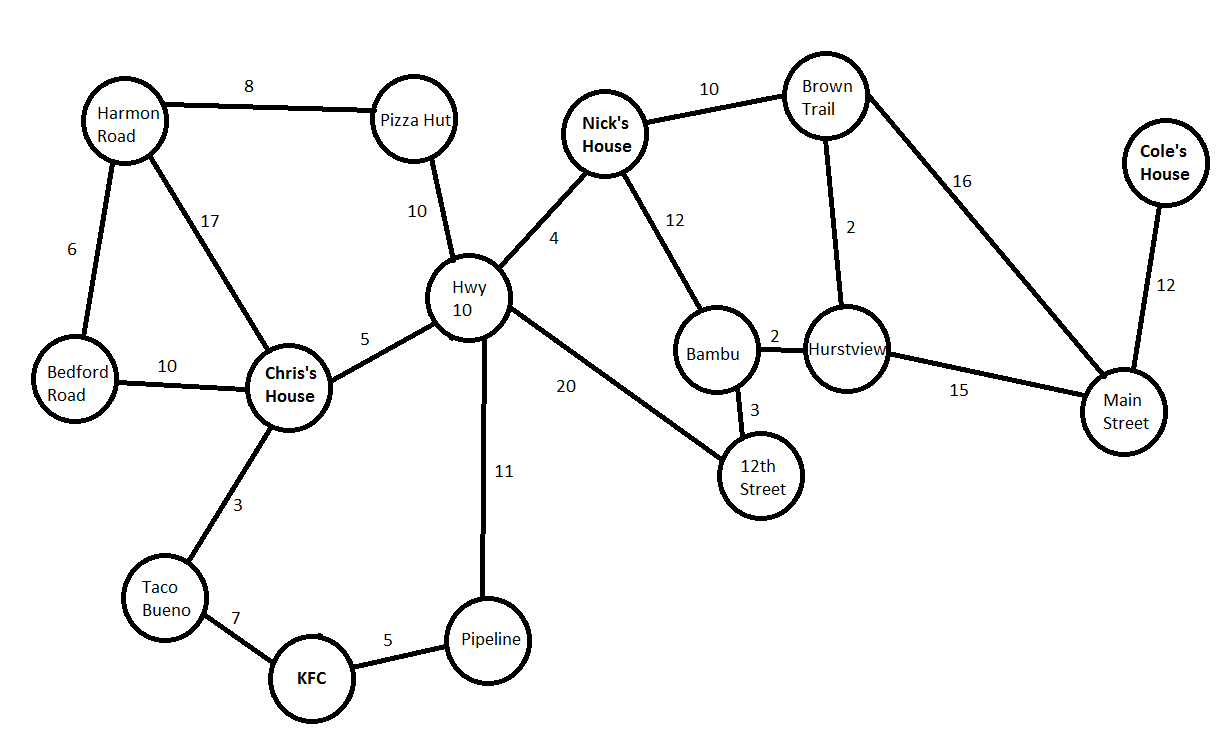
• Knowledge representation technique with examples: A graph, where the start node is the current location of the group. (See the graph at the bottom) For example, the group start’s at Cole’s House and the goal node is the Chinese Restaurant. The search would expand from Cole’s house the shortest node (the only one in this case, Main Street). Then it would expand the next shortest node, Hurstview. Lastly it would expand the next shortest node which is the desired Chinese Restaurant.

• Reasoning mechanism with examples of newly inferred knowledge: If one category of restaurant satisfies more people than the current category, then pick the category that satisfies more people. If there’s a tie between categories, pick the one that has cheaper food. Etc.

• Example Scenario, Knowledge Base Facts, Input, and Output:  
 Scenario: Chris, Cole, Austin, and Nick are having a game night and want to get something to eat. Depending on which house everyone is at, they will decide how willing they are to go out to eat as opposed to having something delivered.  
 KB: “Fried chicken” is\_part\_of “KFC”; “Pizza” is\_ part\_of “Pizza Restaurant”; “Pasta” is\_ part\_of “Pizza Restaurant”; “Pho” is\_ part\_of “Chinese Restaurant”; “Chinese Restaurant” is\_type\_of “Delivery”; “Pizza Restaurant” is\_type\_of “Delivery”; “KFC” is\_type\_of “Dine-in”  
 Input: - Everyone is at Cole’s house.  
 - Cole’s house is far away from most restaurants  
 - Chris doesn’t want to travel far for food.  
 - Nick doesn’t want to travel far for food.  
 - Austin doesn’t want to travel far for food.  
 - Cole likes “Pho”  
 - Nick likes “Chinese Restaurant”  
 - Chris likes “Pasta”  
 - Austin likes “Fried Chicken”  
 - Austin likes “Pho”  
 Output: Based on the fact that most people want to have something delivered, and that the delivery option that satisfies the most people is “Chinese Restaurant”, the group should order Chinese Food. The system will then tell them the quickest way to the restaurant, by going to Main Street > Hurstview > Chinese Restaurant (See map below).

• Programming tools (including third party software tools to be used): Java program, and stored KB test files.

• Architectural Diagram:   




Problems:  
The biggest problem for me was the implementation of the straight line distance for the search. I understand that you can just add up the lengths of every node until you reach the destination, but that is not computationally optimal (likely causes a stack overflow). So I was stuck with having to figure out how to dynamically estimate the straight line distance so as to be admissible, correct, and computationally optimal. I had to do a little bit of trial and error, but I found that searching one layer deeper for each neighbor (looking for either the goal node, or the shortest distance to the next neighbor from there) gave me correct estimates.  
My algorithm finds the optimal path most of the time. I am not confident that my implementation of the straight line distance estimate is correct 100% of the time.

Pending Issues:  
Haven’t given the user the option for delivery, which was just an optional feature I would have liked, but this implementation works great as it is.

Potential Improvements:  
I would say we needed to go over a way to implement the straight line distance in class. All the examples we used just had the values given to us as we practiced the traversal. The implementation of this seemingly simple number is a lot harder than I had imagined.