Amazing Project: Design

Kartik Menon, Zach Kratochvil, Allison Wang

Functional Specification:

1. Input

./AMStartup [AVATAR NUMBER] [DIFFICULTY] [HOST NAME]

[AVATAR NUMBER]
Requirement: Integer $2 < n < 9$

[DIFFICULTY]
Requirement: Integer $0 < n < 9$

[HOSTNAME]
Requirement: Must receive an AM_INIT_OK message to start maze client
Hostname is pierce.cs.dartmouth.edu

./client [avatarID] [shmID] [nAVATAR] [DIFFICULTY] [HOST IP] [MAZEPORT] [FILENAME] [width] [height] [shmid] [ignored] [optional: verbose]

[avatarIP]
Requirement: Must be reachable IP.

[shmID]
Zero or valid shared memory ID.

[MAZEPORT]
Requirement: Must be valid.

[FILENAME]
Requirement: Created before or created by program.

[width]
The width of the maze: a positive integer.

[height]
The height of the maze: a positive integer.

[shmid]
Valid integer key to valid shared char array initialized as a maze.

[ignored]
Required buffer argument that is ignored but allows the optional -v to be detected.

[optional: verbose]
The -v argument will cause textual output of every step.

2. Output

   Should collect log files and .pngs deposited by server and save in files titled according to the provided arguments and containing the number of moves required to solve the maze. Terminal will clear and become ASCII graphics, updated every move with no delay. Avartars and their previous positions are indicated by a single numeric digit. Walls are represented by - and |. And x's indicate paths marked as dead ends. The algorithm uses the graphics' character string, so by commenting out the single DelAvatar call in client.c the graphics can be changed to stop displaying history and only display the current avatar position, but this will disable the dead end blocking algorithm.

Design Specification:

3. Data Structures

   client

   XYPos: {int x, int y} Coordinates of avatar
   Avatar: {int fd, XYPos pos}
   myAvatar:
   {int fd; XYPos pos; XYPos prev; int lastMoveSuccess; int lastMoveDir; } -in header.h
   char *maze

   AMStartup

   No special data structures, AM_MESSAGE

4. Pseudocode

   AMStartup-AMStartup.c

   - Validate command line arguments
   - Converts all command line arguments to relevant form to send to server
   - Send AM_INITIALIZE and check for server response, error if FAILED
   - Get Mazeport information, maze dimensions, IP from AM_INIT_OK
   - Start client with information returned by AM_INIT_OK
   - initialize maze in shared memory
   - prepare arguments to client
   - For each avatar (i = 0; i < nAVATARS; i++), process = fork. Switch on the fork to see if 0 (success), < 0 (failure). If success, execv a new client process.
   - have each process run client
   - Terminate

   client-client.c

   - Validate arguments passed by AMStartup (specifically IP stuff from AM_INIT_OK)
   - Send AM_AVATAR_READY, parameter is ID
   - Wait for AM_AVATAR_TURN response from server.
   - Get current Avatar's x, y from server response save in myAvatar struct
   - Now run an infinite loop that will break if AM_MAZE_SOLVED, AM_SERVER_TIMEOUT, AM_TOO_MANY_MOVES, AM_NO_SUCH_AVATAR_ID, AM_UNKNOWN_MSG_TYPE are received.
   - Use ifs to deal with messages.

- Otherwise send an AM_AVATAR_MOVE to server with direction determined by algorithm (below) and call PrintMaze in graphics.c.
- Wait for AM_AVATAR_MOVE from server with new coordinates.
- Upon receiving AM_MAZE_SOLVED, print solved and exit.
- Clean up and log

Algorithms-graphics.c and lefthandrule.c

Simple: Right or left-hand rule. Have all but one avatar move according to the right/left hand rule. Have moving avatars move right/left until they hit a wall, then north, and then try right/left again. If north AND right/left are impossible, try left/right. Then try north, and continue cycle. This happens in lefthandrule.c.

a. When getting next move, check shared maze if there is an already found wall in that direction ('|' or '-'), or a mark of dead end ('x') using isKnown() in graphics.c.
b. Before making a move, check whether last move indicates wall or dead end and add where appropriate using AddWall and AndMark, respectively, in graphics.c.