UNCC\_ASTR1 Spring 2017

Design: Movement And Pathfinding Tuner App (MAP-Tune)

William Karnavas

January 16, 2017

# Summary:

Create an app to allow fast, broad tweaking and tuning of robot pathfinding algorithms before implementation in LabVIEW and slower, physical testing. Visualizations and quick tuning will be emphasized, as well as swift development.

Final deliverables will include a set of initial parameters to tune our LabVIEW pathfinding and movement modelling code with, and diagrams and animations illustrating robot navigation.

# Critical Requirements:

* Build Time: <= 2 weeks
* Audience: Just me, WK.
* Produce illustrations suitable for posters & documentation
  + Animations?
* Allow quick, gui- or console-based tweaking of path algorithm & parameters at runtime
* Simulate:
  + Pathfinding algorithm
  + Movement modelling
  + Movement error, maybe, if needed

# Project Steps:

1. Choose tools (Lua + LOVE for visuals, over sockets with Java or VEE for controls, as judged swiftest)
   1. Learn tools
      1. LOVE tutorial
      2. Lua or LOVE networking
2. Create detailed initial requirements
   1. Controls & Parameters needed
   2. Visualizations wanted
3. Start coding project (all simultaneously)
   1. Interface, controls, and visualizations
   2. Pathfinding algorithm
   3. Movement modelling
4. Iterate project code

# Getting Results:

1. Iterate sets of parameters
2. Decide on factors to consider when choosing parameters
3. Continue iterating parameters, optimizing to chosen parameters
   1. Try scenarios
      1. From trough to mining area
      2. From mining areas to trough
      3. Lateral movement (as when correcting moving and missing a target point)
      4. Spinning (as when correcting bad alignment when arriving at trough)
4. With near-final parameters, create visualizations
   1. Low res for documents
   2. High res for poster and slides

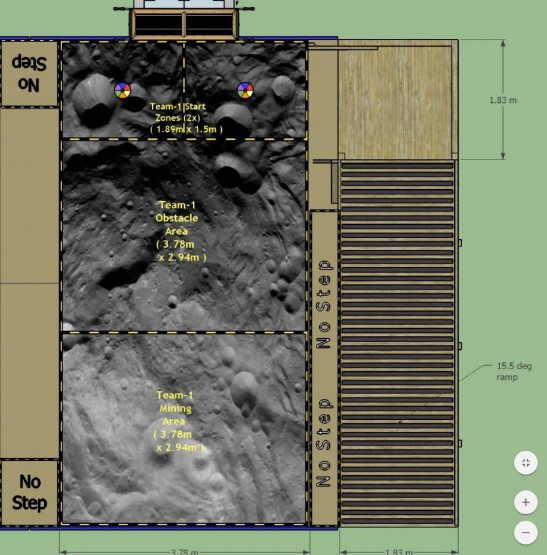
# Initial Requirements:

Below is a list of capabilities MAP-Tune could include.

* Controls & Parameters
  + Situation
    - Start Position, Angle
    - End Position, Angle
      * Pos & Angle Tolerances
    - Whether to choose best mining location as a destination
    - Previously mined locations
  + Environment
    - Path position density (x, y)
  + Path Scoring
    - Weighting constants for all factors, including
      * Straightness per segment
      * Length per segment
      * # of Movements
      * Destination desirability
  + Movement Modelling (if using iterative solution)
    - Movement per iteration (i.e. resolution)
  + Pathfinding
    - Max # of Movements
  + Mining Destination Selection
    - Score gradient (start, end values)
* Readouts & Stats
  + % of paths that reach destination
  + # of paths considered
  + Top N paths:
    - Steps, should show on a map w/modelled curves
    - Scores, total and for each step
* Algorithms
* Visualizations
  + [All arena visualizations]
    - Faint points
    - Already mined areas
    - Start location marker?
  + Points on arena
    - Numbers & Labels
  + Movement model
    - Possible end orientations & “can reach” for each point in arena (sea of arrows)
  + “Track” of a single or multiple paths across the arena
    - Red lines for motion
    - Animated movement?
  + Best destinations heatmap
    - In mining area
  + Some animation of the pathfinding process
  + Path tree visualization
    - Showing “shape” of destination tree, color coding on nodes for “valid destination”
      * Hopefully patterns show up, and we can design a heuristic
  + Save screenshots of visualizations
  + ------------------------------------LIST INCOMPLETE----------------------------------

# Notes:

Arena Size: 3.78m x 7.38m



# Notes on Movement Modelling:

Two approaches to movement modelling were considered. After failing to find a useful solution to the problem using polynomial curves, an iterative solution was found.

The state of the polynomial curve solution is as follows:

* Given information: initial heading, initial position, end position, maximum curvature of curve.
* Expected result: whether curve is possible without exceeding curvature limit, end heading

After informal evaluation of the solution, there appears to be no single possible curve with these characteristics. Study of the problem ended here.

The iterative solution involves inching the “robot” forward from its initial position, changing its heading by a constant maximum amount towards the destination position, as needed. Iteration continues until the “robot” is within tolerance of the destination, or the Euclidian distance of the robot away from the destination begins to increase again (passing destination).