pickpos: A Tool for Preparing Missions

Sep 2023

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139
`project-pavlab/apps/app_pickpos`

1 Overview **1**
1.1 Motivation and Intended Usage . . . 1
1.2 Picking Vehicle Locations with Separation . . . 2
1.3 Picking Vehicle Locations from Multiple Polygons . . . 3
1.4 Picking Vehicle Locations from a Pre-Set List . . . 3
1.5 Picking Vehicle Names . . . 4
1.6 Picking Vehicle Speed . . . 5
1.7 Picking Vehicle Headings (In a Random Range) . . . 5
1.8 Picking Vehicle Headings (Relative to a Point) . . . 6
1.9 Picking Vehicle Colors . . . 6
1.10 Picking Vehicle Groups . . . 7

1 Overview

The `pickpos` utility is a command-line tool for generating starting points for simulated vehicles. It can generate points from a set of points given in a file, or by picking random points from within one or more convex polygons. The starting heading position can also be chosen, in a random range, or relative to a given position. Vehicle names, group names, vehicle speeds, and vehicle colors may also be generated.

Here is an example usage:

```
$ pickpos --amt=4 --polygon="60,-40:60,-160:150,-160"
x=100,y=-139
x=136,y=-146
x=64,y=-103
x=75,y=-73
```

Points will be chosen from the given convex polygon. Note the requested amount is 4, and 4 lines were generated.

This `pickpos` web page can be opened on most systems from the command line with `pickpos --web`. Running `pickpos --help` will also show the supported command line options.

1.1 Motivation and Intended Usage

The motivation for `pickpos` is to support simulations of multiple vehicles. Certain vehicle simulation parameters, such as position, heading, and speed, may need to be randomized. However, it may be

important to bound the random values. For example, we may want to pick vehicle positions within a certain region, or set of regions. However, it may also be important to ensure that no two vehicles are too close to each other to start with.

As with the example above, the pickpos output is in the form of several lines written to the terminal window. The typical usage is to re-direct this output to a file.

```
$ pickpos --amt=4 --polygon="60,-40:60,-160:150,-160" > positions.txt
$ ls
positions.txt
$ cat positions.txt
x=100,y=-139
x=136,y=-146
x=64,y=-103
x=75,y=-73
```

The resulting file contains as many lines as prescribed in the `--amt=N` argument. The information in this file may be used by any other tool, but in our lab, it will likely be ingested within a bash script, into a bash array:

```
VEHPOS=(‘cat positions.txt‘)
```

This array can then be used later in the script to assign the Nth vehicle the starting position found at the Nth position in the array. This strategy is repeated for other starting features, e.g. the vehicle name, vehicle color. For example:

```
$ pickpos --amt=4 --vnames > vnames.txt
$ ls
vnames.txt
$ cat vnames.txt
abe
ben
cal
deb
```

```
VNAMES=(‘cat vname.txt‘)
```

1.2 Picking Vehicle Locations with Separation

In the example above, pickpos generated four points within the specified polygon. The four chosen points are also at least 10 meters apart. The 10 meter buffer range is the default, and can be overridden with the `--buffer=N` argument. pickpos will successively choose a random point in the polygon, each time checking to see if it satisfies the buffer distance. If not, a new point will be chosen. For each point, pickpos will retry to satisfy the buffer criteria up to 1000 times. This can be overridden with the `--maxtries=N` parameter.

At some point, based on the polygon size and number of requested points and buffer distance, it will become impossible to squeeze in any more points while satisfying the buffer criteria. In this case `pickpos` will automatically scale back the buffer distance each time the maxtries (default is 1000) has failed to produce a point. On each successive round, the buffer distance will be scaled back to be 90 per cent of its previous value.

If `pickpos` is run with the `--verbose` option, the points will be produced along with the min distance from each point to all the other points:

```
$ pickpos --amt=4 --polygon="60,-40:60,-160:150,-160" --verbose
x=146,y=-155   nearest=80.40
x=77,y=-93   nearest=14.76
x=76,y=-75   nearest=18.03
x=66,y=-147   nearest=47.04
```

Try running the above example with `--amt=50`. Most points will be at least 10 meters apart, but some have been relaxed. Very few, if any, will be closer than 8 meters.

Note however, the `nearest` information on each line will likely interfere with the consumers of this file. The consumer, at least in our case, is quite particular about the format of each line, so the `nearest=N` component at the end of each line will likely break things further down the road. So it is best to only run `--verbose` to gauge how well `pickpos` is handling the current parameters, and then re-run without it.

1.3 Picking Vehicle Locations from Multiple Polygons

If multiple polygons are provided on the command-line, `pickpos` will generate random points across polygons. For example:

```
$ pickpos --amt=4 --polygon="60,-40:60,-160:150,-160" \
                  --polygon="260,-40:260,-160:350,-160" \
                  --polygon="460,-40:460,-160:550,-160" \
```

It does not matter if the given polygons overlap. It *does* matter if the polygon is non-convex. Only individual convex polygons may be given. However, clearly a non-convex region can be represented by several convex polygons.

Note: The `pickpos` algorithm, when choosing a random point using multiple polygons, first chooses a random polygon. The weight given to each polygon is the same. So if you have one polygon that is 100x bigger than the second one, the distribution of points between the two will be roughly the same, not 100 to 1.

1.4 Picking Vehicle Locations from a Pre-Set List

If the users wishes to choose random vehicle locations from a pre-set list of locations stored in a file, this can be done with the `--posfile` argument as follows:

```
$ pickpos --amt=4 --posfile=file.txt
```

This presumes that a file has been previously generated, either by hand, with pickpos, or some other means. Each choice will be a random choice from the file. No choice will be picked twice, all choices are unique if the lines in the file are unique. If the number of choices requested with the `--amt=N` argument are greater than the number of lines in the input file, no choices will be made, and pickpos will exit with a value of 1, rather than the normal exit value of 0.

1.5 Picking Vehicle Names

With the `--vnames` argument, pickpos will generate a set of vehicle names. These names are not random. They are from a hard-coded list of names in alphabetical order. This is handy in certain missions that are built to launch any number of vehicles. This ensures unique names across all vehicles. In practice there is a strong preference for very short, pair-wise distinct, and alphabetical naming. Using pickpos for generating names ensures the above preferences and provides some consistency across missions.

The first 26 names are three-letter names:

`abe,ben,cal,deb,eve,fin,gil,hal,ike,jim,kim,lou,mal,ned,opi,pal,que,ray,`
`sam,tim,ula,val,wes,xiu,yen,zan`

The second set of 26 names are four-letter names:

`apia,baka,cary,doha,evie,fahy,galt,hays,iola,jing,kiev,lima,mesa,nuuk,oslo,`
`pace,quay,rome,sako,troy,ubly,vimy,waco,xane,york,zahl`

Note: Currently only 52 unique names are supported. The 53rd name will revert to being `abe`. This likely will be extended in a future release.

If the user wishes to use a few names in addition to the pre-set list, this can be done as follows:

```
$ pickpos --amt=5 --vnames=fred,james,bingo
fred
james
bingo
deb
eve
```

The explicitly given names will be used first, followed by the normal sequence of names.

To generate names in reverse use the `--reverse_names` or `-r` argument in addition to the `--vnames` argument. pickpos will generate its list by starting from `zahl` and working backwards.

```
$ pickpos --amt=5 --vnames -r
zahl
york
xane
waco
vimy
```

1.6 Picking Vehicle Speed

Random speed values may be generated with `pickpos` using the `--spd=LOW:HIGH` argument. For example:

```
$ pickpos --amt=5 --spd=2:8
5.4
7
2.4
6.5
7.9
```

Note that values are rounded to the nearest 0.1. If a different resolution is desired, the `-ssnap=N` argument will be used. By default N=0.1.

```
$ pickpos --amt=5 --spd=2:8 --ssnap=0.05
2.15
2.45
4.95
2.6
6.45
```

There is no reason why this argument cannot be used for generating other random values in a given range. However, since this switch was originally created to generate random *speeds*, negative values are not supported. An error will be thrown (a return value of 1) if the LOW value is higher than the HIGH value. If the LOW value is equal to the HIGH value, that is fine, and will deterministically generate lines with the LOW/HIGH value.

1.7 Picking Vehicle Headings (In a Random Range)

As with choosing speeds discussed above, random heading values may be generated with `pickpos` using the `--hdg=LOW:HIGH` argument. For example:

```
$ pickpos --amt=5 --hdg=135:225
211
162
144
145
202
```

If the range of values wraps around true north (zero degrees), then the range to use is $[-180, 180]$. For example to use:

```
$ pickpos --amt=5 --hdg=-10:10
10
5
354
7
357
```

NOT:

```
$ pickpos --amt=5 --hdg=350:10
Unhandled arg: --hdg=350:10
```

1.8 Picking Vehicle Headings (Relative to a Point)

Sometime it is useful to choose a heading relative to a given point, e.g., the center of a polygon, or destination. This can be done by specifying the point, and the relative bearing. *This mode is supported only when also generating random points.* For example, the following chooses a heading pointing directly toward the point $(5, 5)$ from the randomly generated point:

```
$ pickpos --amt=5 --polygon="0,0:10,0:10,10:0,10" --hdg=5,5,0
x=2,y=2,heading=45
x=10,y=9,heading=231
x=9,y=0,heading=321
x=1,y=9,heading=135
x=4,y=6,heading=135
```

And the following generates a heading directly *away* from the point $(5, 5)$ for each randomly generate point:

```
$ pickpos --amt=5 --polygon="0,0:10,0:10,10:0,10" --hdg=5,5,180
x=4,y=9,heading=346
x=9,y=0,heading=141
x=1,y=0,heading=219
x=9,y=6,heading=76
x=5,y=1,heading=180
```

1.9 Picking Vehicle Colors

With the `--colors` argument, `pickpos` will generate a set of vehicle colors. These colors are not random. They are from a hard-coded list of colors. This is handy in certain missions that are built to launch any number of vehicles. This ensures unique colors across all vehicles.

There are 30 vehicle colors in the `pickpos` cache. They are the following:

```
 1. yellow          16. darkslateblue
 2. red             17. brown
 3. dodger_blue     18. burlywood
 4. green           19. goldenrod
 5. purple          20. ivory
 6. orange          21. khaki
 7. white           22. lime
 8. dark_green      23. peru
 9. dark_red        24. powderblue
10. cyan            25. plum
11. coral           26. sienna
12. brown           27. sandybrown
13. bisque          28. navy
14. white           29. olive
15. pink            30. magenta
```

If the user wishes to use a few colors in addition to the pre-set list, these colors will be used first by `pickpos`, replacing the first N cached colors. This can be done as follows:

```
$ pickpos --amt=5 --colors=olive,plum
olive
plum
dodger_blue
green
purple
```

1.10 Picking Vehicle Groups

In some mission, the group name matters. Perhaps there is a red-team and blue-team adversarial mission. Or sometimes there can be ownship(s) vessels, and some number of other contacts with different autonomy system. In any event, `pickpos` can help in generating random or alternating group names. The group name is provided on the command line, for example

```
$ pickpos --amt=4 --grps=red,blue,blue
```

Not above that `blue` was provided twice. This essentially gives `blue` a 2/3 probability that it will be chosen rather than 1/2 probability.

For command-line buffs, try:

```
$ pickpos --amt=10000 --grps=red,blue,blue | fgrep blue | wc -l
```

It should produce a single number, consistently pretty close to 6666.

Sometimes it is useful to generate groups by alternating among the group choices. This can be done by adding the `:alt` suffix to the list of groups. For example:

```
$ pickpos --amt=4 --grps=red,blue:alt
red
blue
red
blue
```

If you're curious, add the `:alt` suffix to the long example above with 10,000 entries. It should always produce exactly 6666.

Final note: While this feature was added with idea of generating group names, this feature can be used for any type of string parameter. For example if there is a Boolean vehicle configuration parameter, that we'd like to set randomly, the `--grps` parameter can be used:

```
$ pickpos --amt=4 --grps=true,false
true
false
false
true
```