Puck Tracking

4/27/16 Brent Roman brent@mbari.org

Without Puck Tracking

- Previously, ESP did not remember puck positions
  - Whenever app started
    - assumed Clamps empty and FlushPuck garaged

- ESP has always tracked the number of pucks in each tube
  - In text file: /var/log/esp/real.puck

- Simple, reliable behavior
  - Operator must return ESP to this “safe” state
    - Via obscure, low-level commands
    - Nerve wracking when the can is closed
  - Especially likely when mission includes system reboots
    - As “long missions” do

ESP Puck Tracking

- Enhances the .puck file to record “unsafely” positioned pucks
  - On app exit
    - Logs warnings of any misplaced pucks
  - Will not work if ESP is powered off while app is running!

- Pucks automatically return to their recorded positions
  - Immediately on app reentry if ESP.ready?
  - Or, immediatley after running ESP.ready!

- Will automatically garage a FlushPuck left in the Hand
  - So that the Hand can hold pucks that were left in Clamps
- Works in simulation modes, too!

Puck Tracking Assumtions

- Puck Tracking logic will be “confused” if:
  - Pucks moved by hand
    - Or using “low-level” Arm. commands
  - ESP app is not allowed to exit normally
    - Exit the ESP app before cutting power!
  - The .puck file is removed or modified
    - /var/log/esp/${ESPmode}.puck
- Use the pucks, clear!, or forgetESPstate commands
  - To correct puck tracking state

Puck Tracking File

- Example /var/log/esp/${ESPmode}.puck:

```
@START,21:33:33PDT27-Apr-16
1:0,2:22,3:22,4:22,5:22,6:22,7:22   #fill!
@EXIT,21:33:33PDT27-Apr-16
@START,21:33:33PDT27-Apr-16
2:21    #Puck.move 2,1
1:1
!CC.holds FlushPuck   #exitted with FlushPuck in CC
*EXIT,21:34:40PDT27-Apr-16
```

Puck Tracking Commands

- clear! tube(s)
  - Forget number of pucks in each specified
    - or all tubes, if argument omitted
  - Example: clear! 2..4,7 #forget pucks in tubes 2..4 and 7
- access tube
  - Rotate carousel to access specified tube and clear! it
  - Always use when manually loading or unloading pucks
- safe?
  - Confirm pucks are stowed in their “safe” positions
- reset!
  - Forget all puck counts, tracking and long mission state
  - Optional arguments are passed into fill!

Declaring Puck Stack Heights

- Puck stack height cannot be measured in simulation
  - Puck load must be prescribed in simulations
- Every new mission should define the number of pucks expected to be loaded in each tube!
  - Optional in “real” mode, but...
  - Isn't it better to “fail early” if puck load is wrong?
- Excerpt of mission with 6 pucks in tubes 2, 3 and 4:

```
pucks 2=>6, 3=>6, 4=>6    # see next slides
mission startTube:  2, until:  “9AM  4/10/15”  do
    <mission phases>
end
```

- Fails immediately if tube 2 did not start with exactly six pucks

Declaring Puck Stack Heights

- New commands to set and query the expected stack height:
  - clear! *tubeList=1..7*
    - Clears each specified tube's stack height
  - fill! *numPucks=22*, *tubeList=2..6*
    - Puts the specified number of pucks in each listed tube
  - pucks *tubeHash={}*
    - Puts the specified number of pucks in specified tubes
    - If tubeHash omitted, just displays the # of pucks in each tube

Detailed Stack Height Setting

- fill!
  - Fills all tubes except #1 (for typical fully loaded carousel mission)
- fill!; clear! 2, 4..7
  - Ends up with tube 3 containing 22 pucks, others empty
- fill! 9
  - Fills all tubes except #1 with 9 pucks
- fill! 9, 1, 3..5, 7
  - Fills tube 1, 3, 4, 5 and 7 with 9 pucks
- pucks 2=>22, 6=>18
  - Fills tube #2 with 22 pucks, tube #6 with only 18
- pucks
  - Changes nothing
  - Just returns the hash of pucks in tubes.