Overview

In this lab we exercise our knowledge in:
- MOOS and Helm configurations, by continuing to extend the Alpha mission
- Behavior updates parameter, Loiter, MaxDepth, ConstantDepth behaviors.

A few new topics we touch, and learn on the fly:
- Simulated drift
- IvP Helm Decision Space
- Behavior durations

Lab Exercises
- **s8_alpha**: Change the Alpha Mission to use a loiter behavior
- **s9_alpha**: Turn the vehicle into UUV, adding constant depth behavior
- **s10_alpha**: Change the vehicle depth with pMarineViewer
- **s11_alpha**: Use uTimerScript to periodically randomly choose a new depth
- **s12_alpha**: Use MaxDepth behavior to limit the maximum depth
- **s13_alpha**: Periodic surface for fixed time (GPS fix)
- **s14_alpha**: Add simulated drift to the mission
The s8_alpha Exercise

To Start: Copy the s1_alpha mission, calling it s8_alpha

- Remove the Waypoint Survey behavior and replace with a Loiter Behavior
- Loiter at Position 75,-75, with a radius of 30 meters, in an octagon
Requirements of the s8_alpha Exercise

1. Remove the Waypoint behavior
2. Add the Loiter Behavior

```plaintext
//==============================================
Behavior = BHV_Loiter
{
  name = loiter
  priority = 100
  condition = RETURN = false
  condition = DEPLOY = true
  updates = UP_LOITER
  speed = 1.3
  clockwise = false
  radius = 6.0
  slip_radius = 25.0
  polygon = format=radial, x=75, y=-75, radius=30, pts=8, snap=1
  visual_hints = nextpt_color=yellow, nextpt_lcolor=khaki
  visual_hints = edge_color=white, vertex_color=dodger_blue
  visual_hints = edge_size=1, vertex_size=3, label=LOITER_POLYGON
  visual_hints = nextpt_vertex_size=5
}
```

The s9_alpha Exercise
The s9_alpha Exercise

To Start: Copy the s8_alpha mission, calling it s9_alpha

- Add Depth to the mission.
- Set the Loiter depth to be 20 meters

Requirements of the s9_alpha Exercise

1. Add the ConstantDepth Behavior
2. Configure the Helm to reason about depth
3. Configure the Simulator to reason about depth
4. Configure the PID controller to reason about depth

1. Add the ConstantDepth Behavior (to the alpha.bhv file)

```plaintext
// -------------------------------------------------
Behavior=BHV_ConstantDepth
{
    name = const_depth
    pwt  = 100
    condition = DEPLOY = true
    duration = no-time-limit
    updates = DEPTH_UPDATE
    depth = 20
}
```

This behavior will be active when surveying and returning
Requirements of the s9_alpha Exercise

2. Configure the Helm (pHelmIvP) to reason about depth

```plaintext
// HelmIvP configuration block
ProcessConfig = pHelmIvP
{
  AppTick = 4
  CommsTick = 4
  behaviors = alpha.bhv
  domain = course:0:359:360
  domain = speed:0:4:41
  domain = depth:0:100:101
}
```

Add this line:
The helm will now also consider depths 0, 1, 2, ... 100.

Requirements of the s9_alpha Exercise

3. Configure the Simulator (uSimMarine) to reason about depth

```plaintext
// uSimMarine configuration block
ProcessConfig = uSimMarine
{
  AppTick = 4
  CommsTick = 4
  start_x = 0
  start_y = -20
  start_heading = 180
  start_speed = 0
  prefix = NAV
  turn_rate = 40
  thrust_map = 0:0, 20:1, 40:2, 60:3, 80:4, 100:5
  buoyancy_rate = 0.075
  max_depth_rate = 5
  max_depth_rate_speed = 2.0
  default_water_depth = 400
}
```

Add these lines:
Requirements of the s9_alpha Exercise

4. Configure the PID Controller (pMarinePID) to reason about depth

```cpp
Change from false to true
```

```cpp
Add these lines:
```

The s10_alpha Exercise
The s10_alpha Exercise

To Start: Copy the s9_alpha mission, calling it s10_alpha

- Add buttons to pMarineViewer to command depth=20 and depth=40
- Use the updates parameter for the ConstantDepth behavior

Requirements of the s10_alpha Exercise

1. Use the updates parameter to allow dynamic changes to the ConstantDepth behavior

```c
//------------------------------
Behavior=BHV_ConstantDepth
{
    name       = const_depth
    pwt        = 100
    condition  = DEPLOY = true
    duration   = no-time-limit
    updates    = DEPTH_UPDATE
    depth      = 20
}
```

Add this line
Requirements of the s10_alpha Exercise

2. Configure pMarineViewer with two new command buttons

```cpp
// pMarineViewer configuration block
ProcessConfig = pHelmIvP
{
    AppTick    = 4
    CommTick   = 4
    button_three = DEPTH_20 // DEPTH_UPDATE = depth=20
    button_four  = DEPTH_40 // DEPTH_UPDATE = depth=40
}
```

Add these two lines

<table>
<thead>
<tr>
<th>Button label</th>
<th>MOOS Variable</th>
<th>Behavior Parameter</th>
<th>Parameter Value</th>
</tr>
</thead>
</table>

The s11_alpha Exercise
The s11_alpha Exercise

To Start: Copy the s10_alpha mission, calling it s11_alpha

- Add a Timer Script to randomly change depths every two minutes
- Choose random depths in the range 20 to 80 meters

Requirements of the s11_alpha Exercise

1. Add \texttt{uTimerScript} to the alpha.moos Antler configuration block

```c
// Antler configuration block
ProcessConfig = ANTLER
{
    MSBetweenLaunches = 200
    Run = MOOSDB    @ NewConsole = false
    Run = pLogger   @ NewConsole = false
    Run = uSimMarine@ NewConsole = false
    Run = pMarinePID@ NewConsole = false
    Run = pHelmIvP  @ NewConsole = false
    Run = pMarineViewer@ NewConsole = false
    Run = uProcessWatch@ NewConsole = false
    Run = pNodeReporter@ NewConsole = false
    Run = uTimerScript @ NewConsole = false
}
```

Add this line
Requirements of the s11_alpha Exercise

1. Add the `uTimerScript` configuration block to the `alpha.moos` file

```moos
// uTimerScript configuration block
ProcessConfig = uTimerScript
{
  AppTick   = 4
  CommsTick = 4
  condition = DEPLOY = true
  randvar   = varname=RND_DEPTH, min=20, max=80, key=at_reset
  event     = var=DEPTH_UPDATE, val=depth=$[RND_DEPTH], time=120
  reset_max = nolimit reset_time = all-posted
}
```

Requirements of the s11_alpha Exercise

Confirm by looking at the alog file with alogview.
The s12_alpha Exercise

To Start: Copy the s11_alpha mission, calling it s12_alpha

- Add a MaxDepth behavior
- Choose random depths in the range 20 to 80 meters, but MaxDepth limits to 50m
Requirements of the s12_alpha Exercise

1. Add the MaxDepth behavior to the alpha.bhv file.

```c
// -------------------------------------------------------------
Behavior=BHV_MaxDepth
{
    name       = max_depth
    pwt        = 200
    condition  = DEPLOY = true
    duration   = no-time-limit
    max_depth  = 50
    basewidth  = 0
    // Priority weight higher than constant depth behavior
    // No tolerance beyond max_depth
}
```

Requirements of the s12_alpha Exercise

Confirm by looking at the alog file with alogview
To Start: Copy the s9_alpha mission, calling it s13_alpha

- Add a Timer behavior to periodically (30 secs) turn off the depth behavior
- Float to the surface, wait at the surface for 30 secs, then resume
Requirements of the s13_alpha Exercise

1. Add Two Timer behaviors to the alpha.bhv file.

```plaintext
Behavior=BHV_Timer
{name       = timer_need_gps
condition  = DEPLOY   = true
condition  = NEED_GPS = false
perpetual  = true
endflag    = NEED_GPS = true
duration   = 300
duration_idle_decay = false
}
```

Only running when NEED_GPS is true
Shuts off its own condition (NEED_GPS) when it completes

```plaintext
Behavior=BHV_Timer
{name       = timer_wait_gps
condition  = DEPLOY = true
condition  = NAV_DEPTH < 2
condition  = NEED_GPS = true
perpetual  = true
endflag    = NEED_GPS=false
duration   = 30
duration_idle_decay = false
}
```

This timer starts only when at the surface
After a 30 secs at the surface it completes
Set timer to NOT decay when idle

2. Initialize the variable indicating a need for GPS to be false

In the alpha.bhv file:

```plaintext
initialize   DEPLOY   = false
initialize   RETURN   = false
initialize   NEED_GPS = false
```

Add this line
The s14_alpha Exercise

To Start: Copy the s13_alpha mission, calling it s14_alpha

- Add a drift vector to the simulator, accessible by buttons in pMarineViewer
- Note the performance of the UUV with effects of drift
Requirements of the s14_alpha Exercise

1. Add two buttons to pMarineViewer to turn OFF/ON the drift vector

```plaintext
button_one = DEPLOY # DEPLOY=true
button_one = MOOS_MANUAL_OVERRIDE=false # RETURN=false
button_two = RETURN # RETURN=true

button_three = DRIFT_ON # DRIFT_VECTOR = 45, 0.3
button_four = DRIFT_OFF # DRIFT_VECTOR = 0, 0
```

Add these lines to uSimMarine registers for this mail. The mail is a string "drift angle, meters/sec".

END