CS 350 Project Part 3 Team 8
Evan Nilson
Samir Ouahhabi
Colton Prettyman

A. Fighter Tests

Test A.1: Definition

1.

Tests to see whether a fighter jet can be properly defined and created.

2.

We will define a fighter agent with the following attribute values:
Fighter: speed min **10** max **20** delta inc **40** dec **35** turn **15** climb **10** descent **20** weight **55** fuel **89** delta **32**
OLS: diameter **10**
Boom: length **200** diameter **40** flow **40**
Tailhook: time **10**
Tank: amount **50**
Initial coordinates: **49*39'50"/117*25'00"** Altitude: **2000** Heading: **090** Speed: **25**

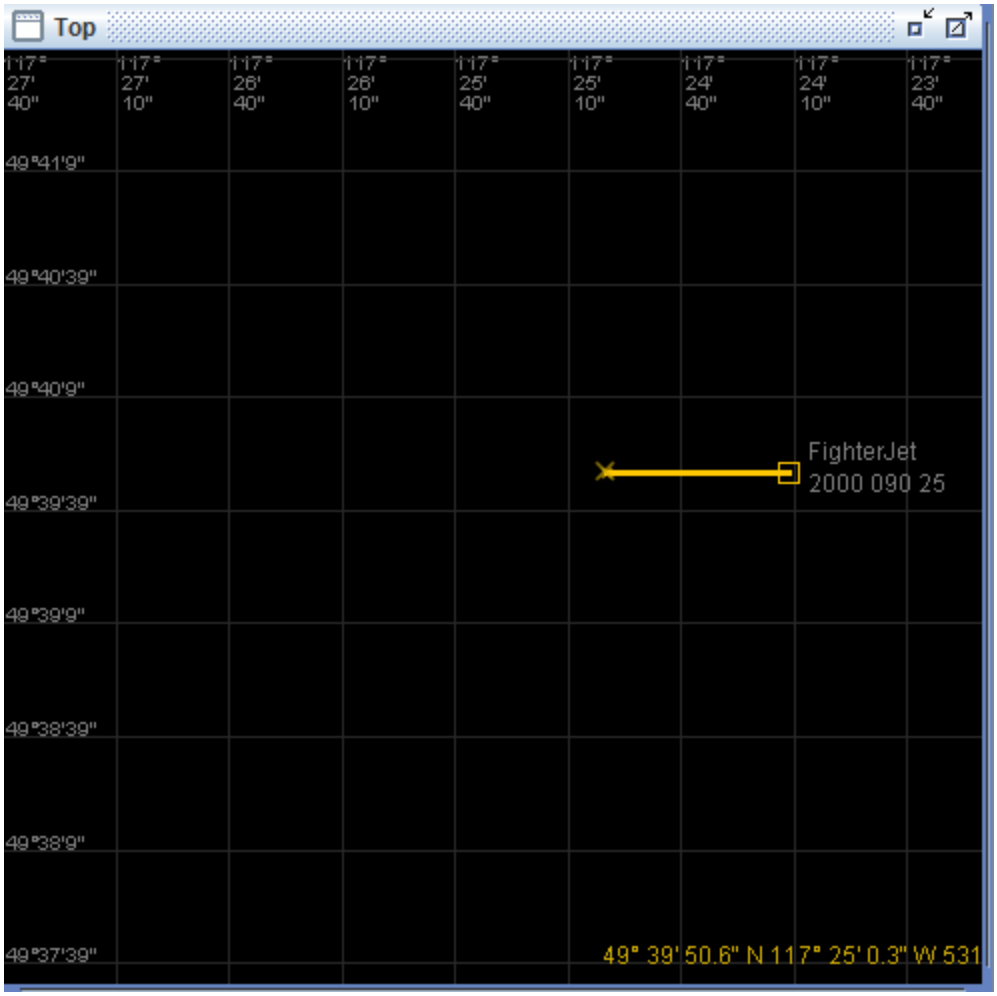
3.

The following commands were used for the test:

```
DEFINE FIGHTER F_1 SPEED MIN 10 MAX 20 DELTA INCREASE 40 DECREASE 35 TURN 15
CLIMB 10 DESCENT 20 EMPTY WEIGHT 55 FUEL INITIAL 89 DELTA 32
DEFINE OLS_RCV RCV_1 DIAMETER 10
DEFINE BOOM MALE BM_1 LENGTH 200 DIAMETER 40 FLOW 40
DEFINE TAILHOOK TH_1 TIME 10
DEFINE AUX_TANK AT_1 AMOUNT 50
CREATE OLS_RCV Receiver FROM RCV_1
CREATE BOOM BoomMale FROM BM_1
CREATE TAILHOOK THook FROM TH_1
CREATE AUX_TANK FighterTank FROM AT_1
CREATE FIGHTER FighterJet FROM F_1 WITH OLS Receiver BOOM BoomMale TAILHOOK
THook TANKS FighterTank AT COORDINATES 49*39’50”/117*25’00” ALTITUDE 2000
HEADING 090 SPEED 25

POPULATE WORLD WITH FighterJet

COMMIT
```

4.
We expect the simulation world to be populated with the fighter agent defined above, starting at the coordinates specified in **A.1.3.**

5.
Agent FighterJet was defined on the coordinates 49*39’50”/117*25’00”, with a heading of 090 degrees and speed of 25.



6.
Test results are consistent with the expected results.

7.
To further expand the test, we should consider conducting a similar test where multiple Fighter agents are created from the same defined OLS, Boom, Tailhook, and Tank.

Test A.2: Acceleration

1.
Tests to see whether the Fighter agent can properly accelerate to target speed.

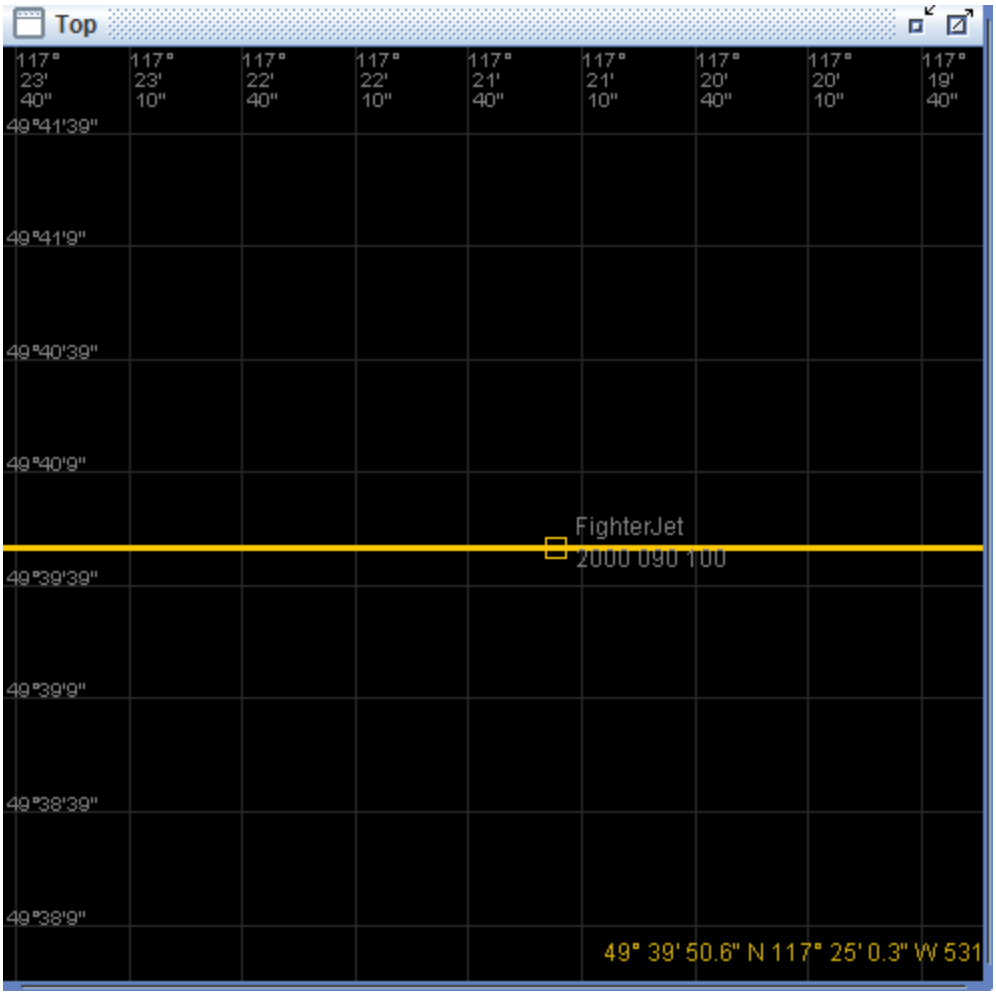
2.
From an initial speed of 25, we will set the Fighter agent's speed to 100, and see if it accelerates.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39'50"/117*25'00";@DO FighterJet FORCE SPEED 25`

The following commands were used for the test:
`DO FighterJet SET SPEED 100`

4.
We expect the Fighter agent to gradually accelerate to a speed of 100, and no further.

5.
The Fighter agent maintained its current heading, and accelerated from a speed of 25 to 100 over time. The agent continued its heading, and maintained the set speed of 100.



6.
The test results are consistent with the expected results.

7.
The test could be expanded by trying to accelerate the Fighter agent to a negative speed.

Test A.3: Deceleration

1.
Tests to see whether the Fighter agent can properly decelerate to target speed.

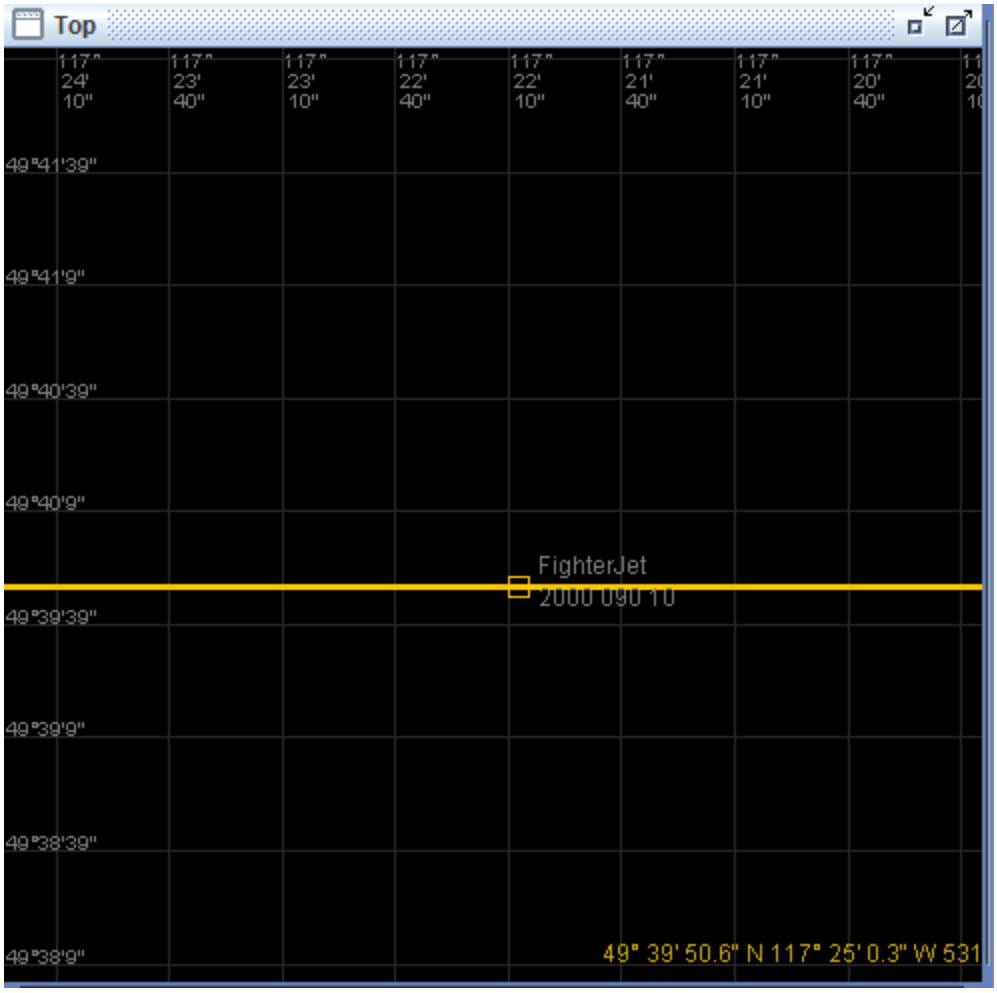
2.
From an initial speed of 100, we will set the Fighter agent's target speed to 10, and see if it decelerates to attain that speed.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39'50"/117*25'00"`
`@DO FighterJet FORCE SPEED 100`

The following commands were used for the test:
`DO FighterJet SET SPEED 10`

4.
We expect the Fighter agent to gradually lower its speed until it reaches a speed of 10, and then continue its heading at that speed.

5.
The agent FighterJet gradually reduced its speed until it reaches the target value of 10.
It maintained its set heading while doing so.



6.
The test results are consistent with the expected test results.

7.
The test could be expanded by trying to decelerate the Fighter agent to a speed of 0, which should only be possible when an aircraft is grounded.

Test A.4: Climb

1.
Tests to see if a Fighter agent is capable of increasing altitude.

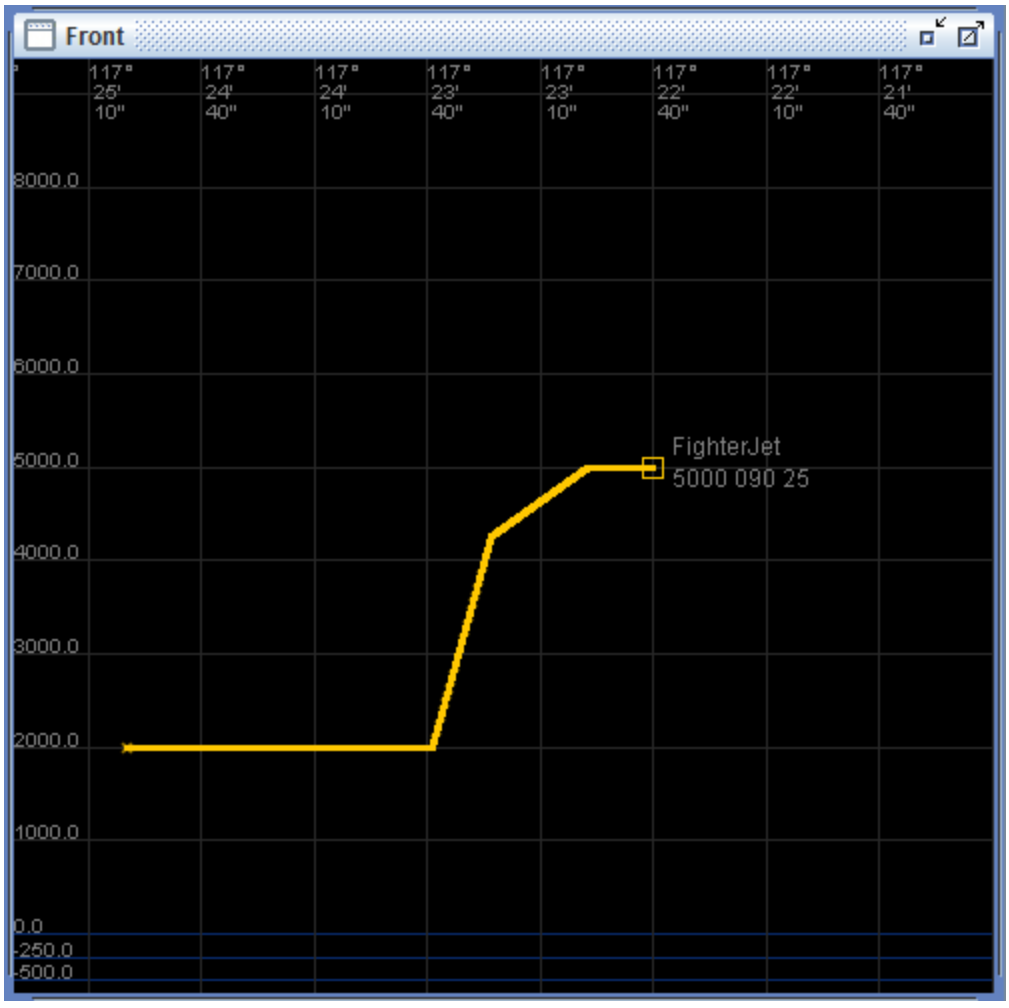
2.
From an initial altitude of 2000, we will climb to an altitude of 5000.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE ALTITUDE 2000`

The following commands were used for the test:
`DO FighterJet SET ALTITUDE 5000`

4.
The Fighter agent should gradually increase its altitude to 5000, while maintaining its current heading and speed. It should not exceed 5000 altitude.

5.
The fighter jet climbed up to an altitude of 5000, maintaining its heading and speed.



6.
The test results are consistent with expected results, however the climb the Fighter agent made to reach its target altitude was remarkably steep.

7.
We could expand the test by trying to change the Fighter agent's altitude while it's already changing altitude.

Test A.5: Descent

1.
Tests to make sure the Fighter agent can decrease altitude.

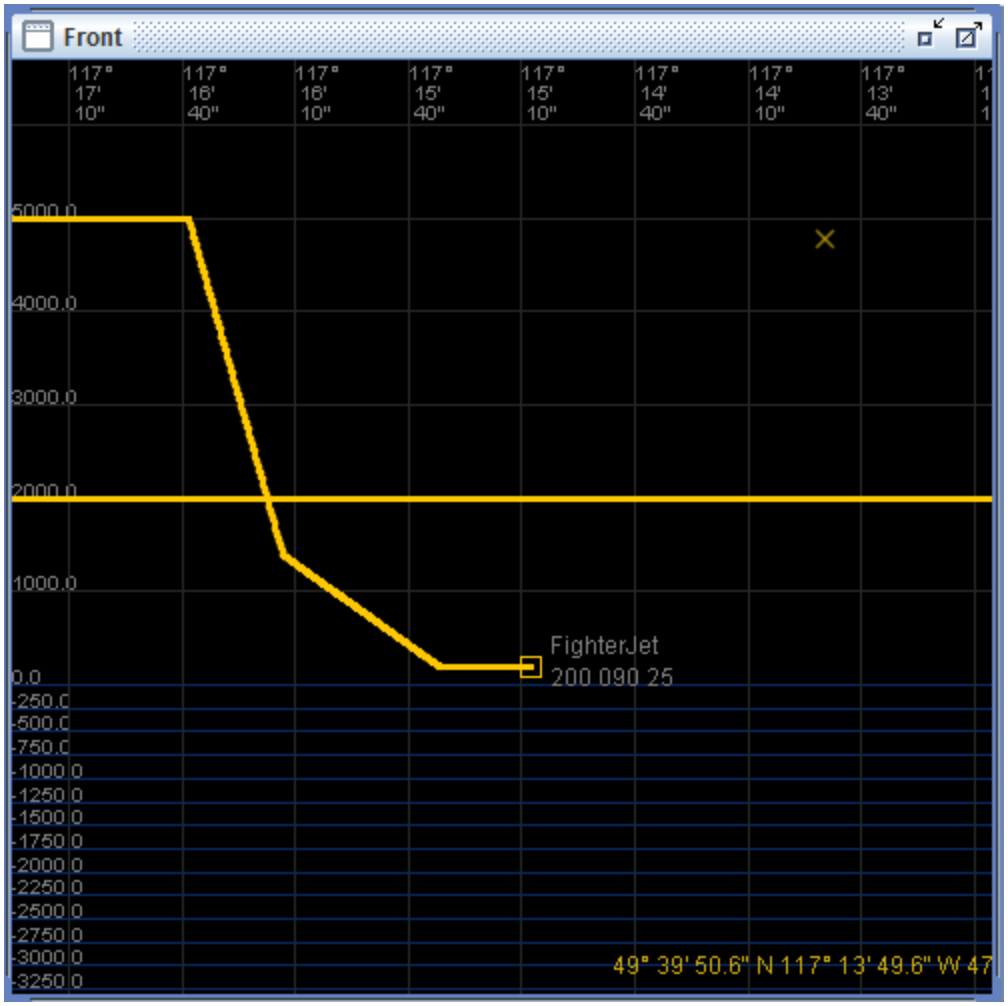
2.
From a starting altitude of 2000, the Fighter agent must descend to an altitude of 200.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39'50"/117*25'00";`
`@DO FighterJet FORCE ALTITUDE 2000`

The following commands were used for the test:
`DO FighterJet SET ALTITUDE 200`

4.
We expect the Fighter agent to gradually decrease altitude to 200, without changing heading or speed.

5.
The Fighter agent made a sharp descent to its target altitude of 200.



6.
The test results are consistent with expected results, but the decline the agent made to reach its target altitude was quite steep.

7.
This test could be further expanded by attempting to go to a negative altitude, which would make the Fighter agent descend into the water.

Test A.6: Heading Change Left Clean

1.
Tests to see if the Fighter agent can change its heading from a larger heading value to a smaller heading value in a counterclockwise direction.

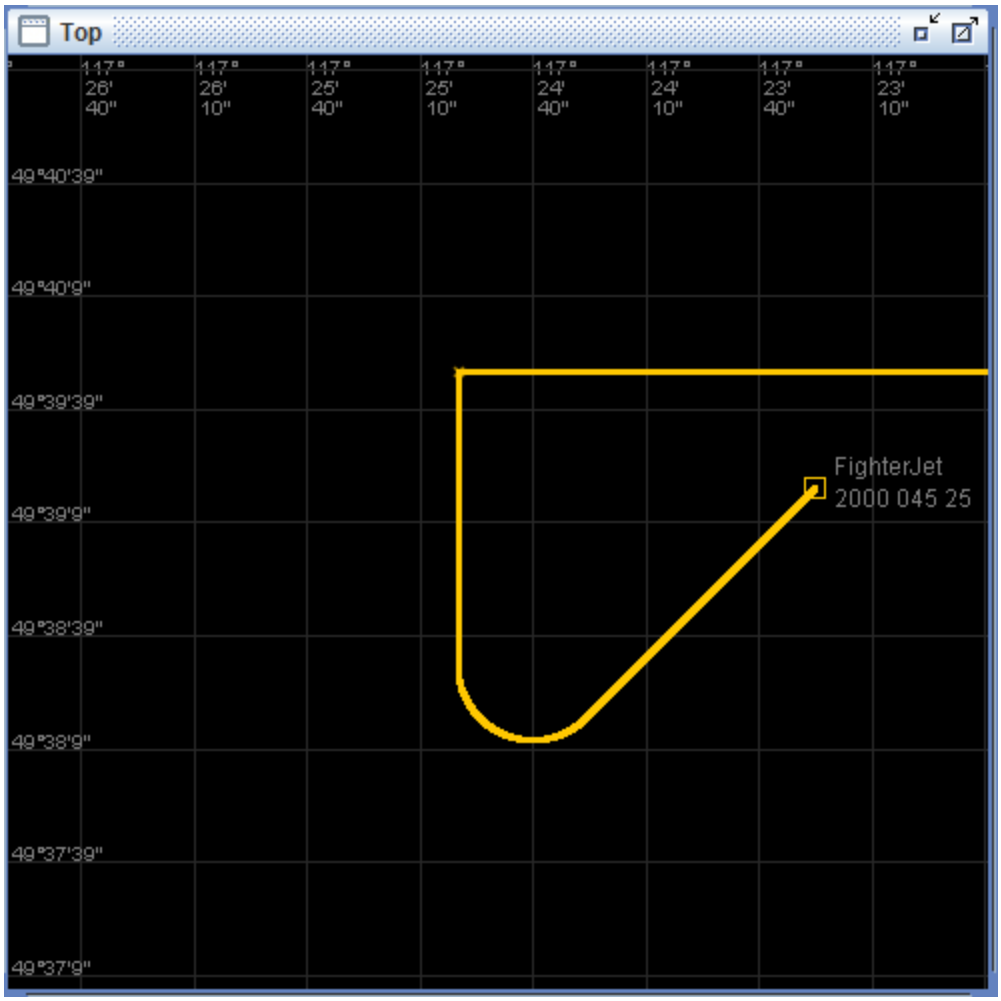
2.
From an initial heading of 180, the Fighter agent must change heading to 45 counterclockwise.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE HEADING 180`

The following commands were used for the test:
`DO FighterJet SET HEADING 045 LEFT`

4.
The Fighter agent will change its heading from 180 to 45 by turning left.

5.
The Fighter agent turned left in order to change its heading to 45. It maintained its speed and altitude.



6.
Test results are consistent with expected results.

7.
Test could be expanded by making a heading change before the agent reaches its target heading.

Test A.7: Heading Change Left Straddling

1.
Tests the behavior of the Fighter agent when we make a heading change from a smaller heading to a larger heading counterclockwise.

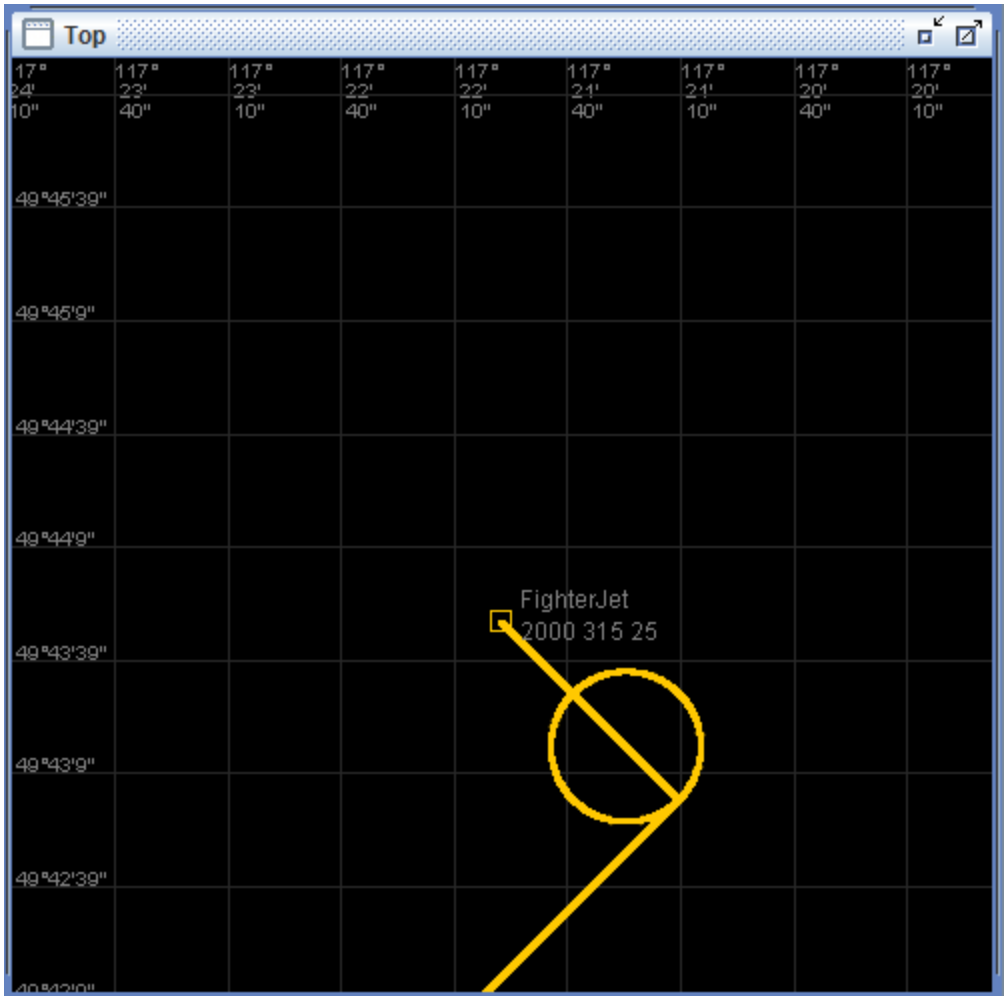
2.
From an initial heading of 45, the Fighter agent must change its heading to 315 counterclockwise.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39'50"/117*25'00";@DO FighterJet FORCE HEADING 45`

The following commands were used for the test:
`DO FighterJet SET HEADING 315 LEFT`

4.
The agent should turn left to change its heading to 315.

5.
Initially, the Fighter agent turned left to change its heading to 315. It continued turning left even after it hit the target heading. It kept turning until it made a full circle, then it immediately changed heading to 315 in a sharp left turn.



6.
The test results were inconsistent with the expected results. The jet did not stop once it reached target heading, even though it ultimately reached the target heading. It made a complete circle instead of stopping once it attained target heading.

7.
The test could be expanded to see if it is possible for the Fighter agent to change heading at a speed of 0.

Test A.8: Heading Change Right Clean

1.
Tests to see if the Fighter agent is capable of changing from a smaller heading to a larger heading by turning right.

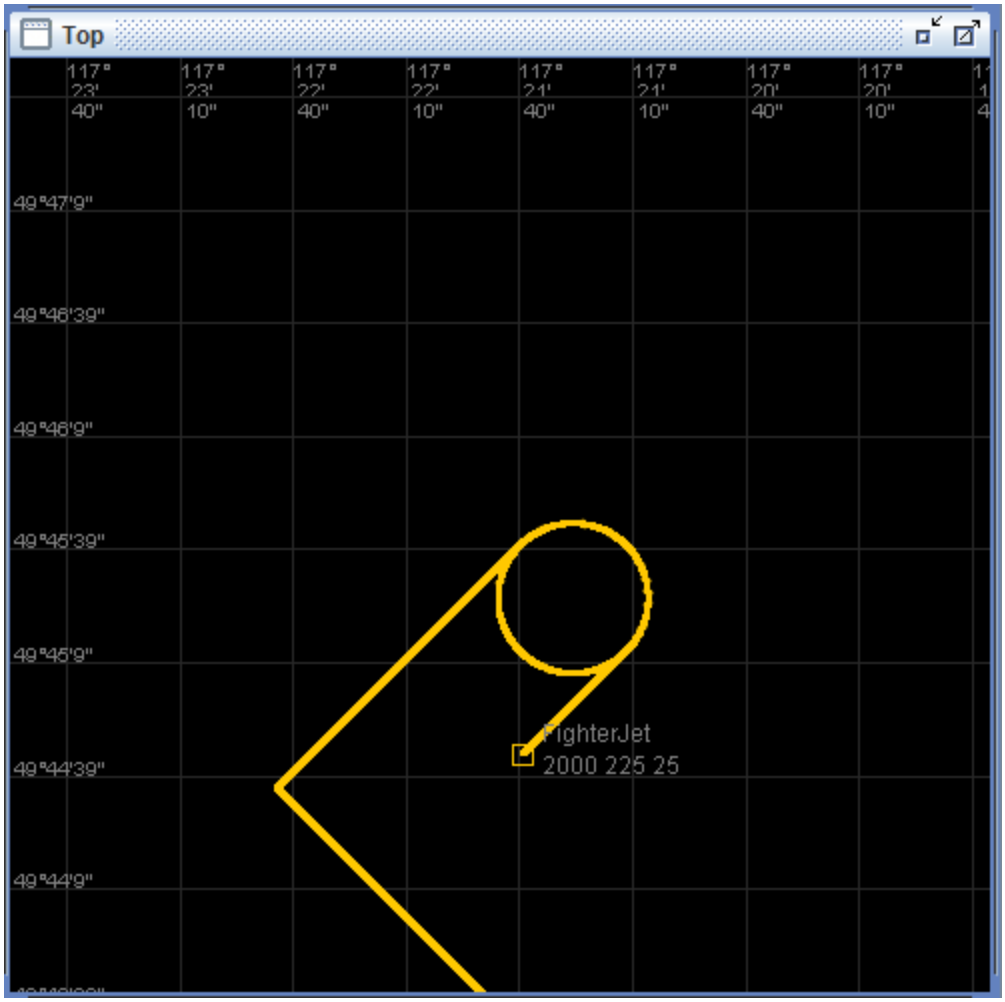
2.
From an initial heading of 45, the fighter agent must change heading to 225 by turning right.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE HEADING 45`

The following commands were used for the test:
`DO FighterJet SET HEADING 225 RIGHT`

4.
We expect the Fighter agent to the target heading by turning right.

5.
The Fighter agent changed to target heading by turning right, but did not stop once it had reached the target heading. It kept turning until it had made a complete circle, then continued until it had reached a heading of 225. Once it had reached the heading 225, it behaved normally.



6.
The test results were not consistent with the expected results. The agent made a complete circle before actually trying to change to target heading.

7.
We could further expand heading tests by testing for an agents ability to recognize a negative heading value.

Test A.9: Heading Change Right Straddling

1.
Tests to see if a Fighter agent is capable of changing from a larger heading to a smaller heading by turning clockwise.

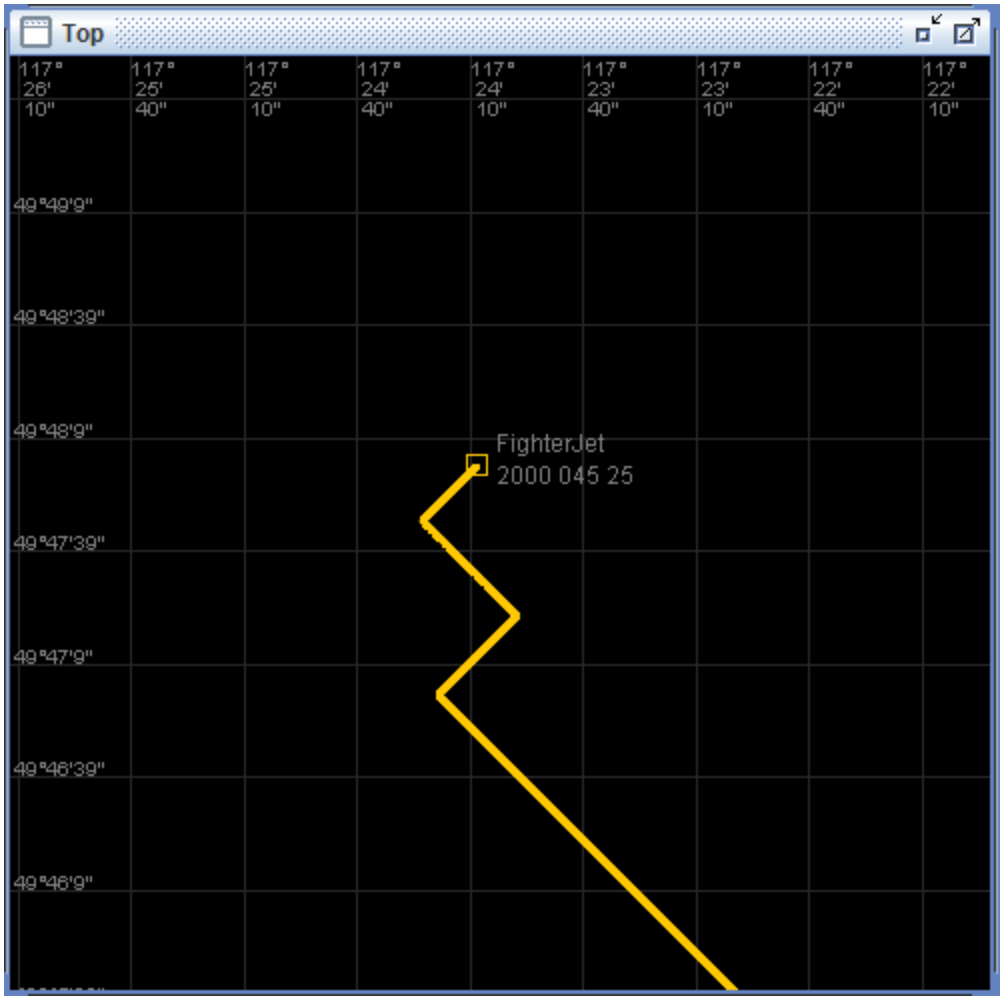
2.
From an initial heading of 315, the agent must change to a heading of 45 by turning right.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39'50"/117*25'00";@DO FighterJet FORCE HEADING 315`

The following commands were used for the test:
`DO FighterJet SET HEADING 045 RIGHT`

4.
The Fighter agent should gradually turn right in order to reach its target heading of 45.

5.
The Fighter agent achieved its target heading, but did not do so gradually; instead it instantly changed its heading to the target value.



6.
The test results are not consistent with the expected results. Instead of turning gradually, the agent instantly changed its heading to the target.

7.
For this particular command, we could test to see what will happen if we enter a heading greater or equal to 360.

Test A.10: Heading Change Shortest Clean

1.
Tests to see how the Fighter agent behaves when it must go from a smaller heading to a larger heading without a specified direction to turn.

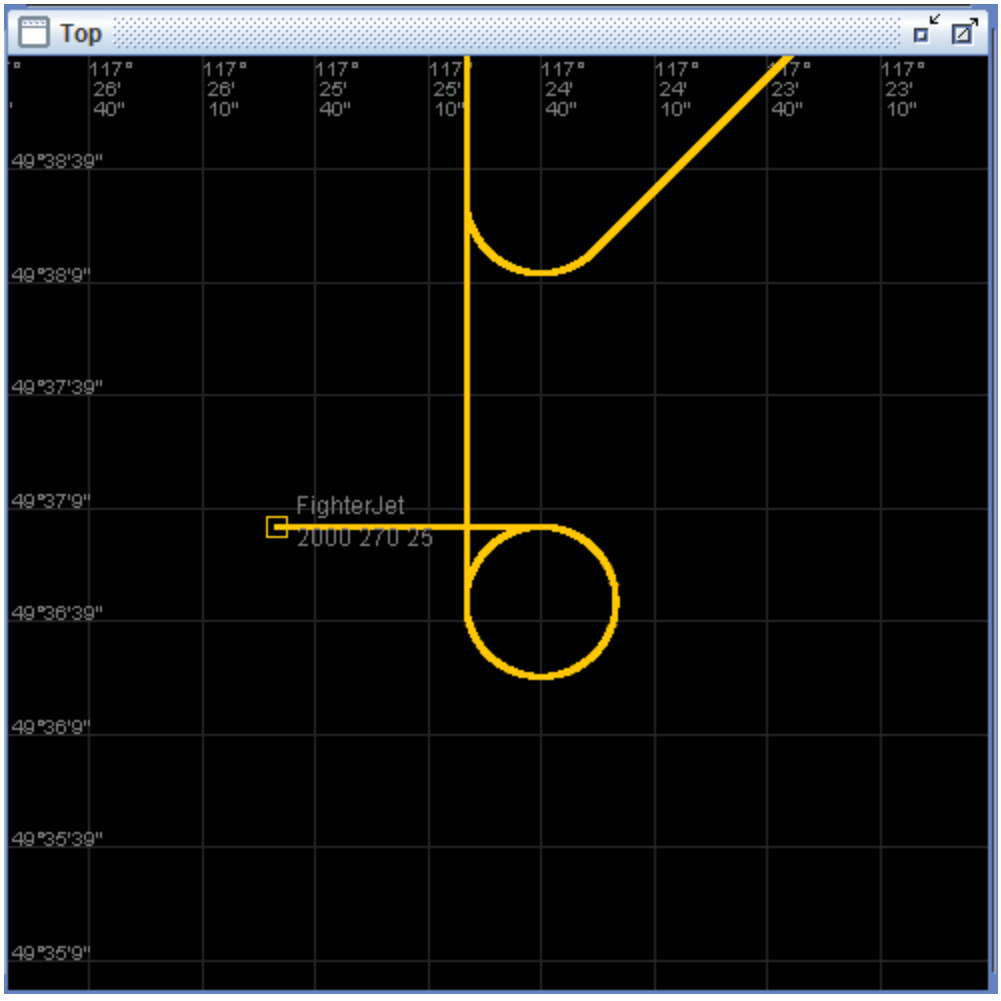
2.
The agent must change to a target heading of 270 from a starting heading of 180.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE HEADING 180`

The following commands were used for the test:
`DO FighterJet SET HEADING 270`

4.
We expect the Fighter agent to gradually turn right to reach the target heading of 270, as this is shorter than turning left to reach heading 270.

5.
The Fighter agent gradually turned left, completing a full circle, in addition to another 3/4ths of a circle before finally stopping at its target heading of 270.



6.

The test results are not consistent with the expected results. Not only did the agent turn in a different direction than we had expected, but it also made 1 and 3/4ths of a circle before stopping at its target heading.

7.
We could conduct further testing by seeing if the agent would still choose to turn left if the target heading was 181 with the current being 180. If this is the case, perhaps the agent always defaults to a left turn if no direction is specified.

Test A.11: Heading Change Shortest Straddling

1.
Tests to see whether a Fighter agent is capable of choosing the most efficient path to reach target heading.

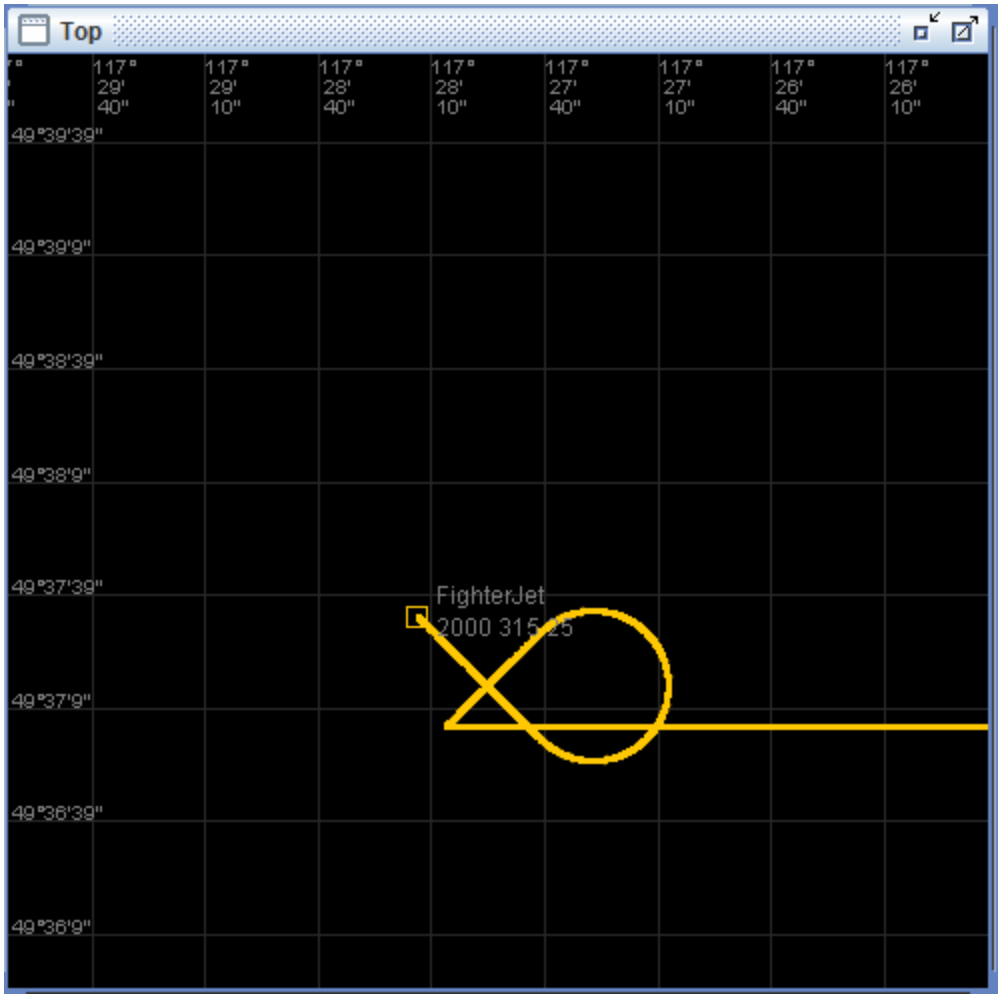
2.
From an initial heading of 315, the Fighter agent must choose the most efficient path to reach a target heading of 45 when no direction is specified.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39'50"/117*25'00";@DO FighterJet FORCE HEADING 315`

```
DO FighterJet SET HEADING 045
```

4.
We expect the Fighter agent to turn left in order reach the target heading.

5.
The Fighter agent turned right in order to reach the target heading of 315.



6.
The test results are inconsistent with the expected results. Instead of turning left, the agent turned right in order to reach the target heading of 315.

7.
It is possible the shortest turn to reach target heading isn't always the best route. We could conduct further tests for different heading changes and be able to derive a logical behavior from it.

Test A.12: Heading Change Shortest Straddling 2

1.
Tests the behavior of a Fighter agent when changing heading without a specified direction.

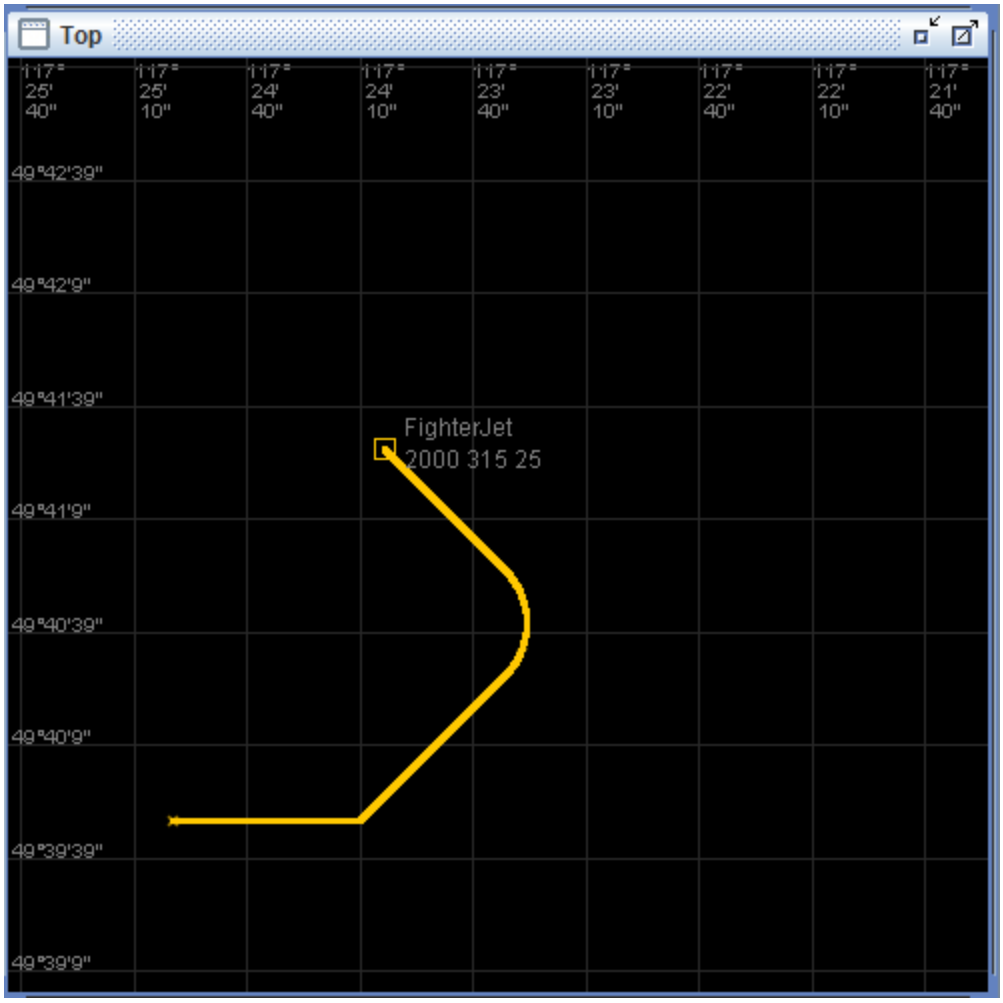
2.
The Fighter agent must change from a heading of 45 to a heading of 315 without a specified direction. We hope that the Fighter agent chooses the most efficient path to reach the target heading.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE HEADING 045`

The following commands were used for the test:
`DO FighterJet SET HEADING 315`

4.
We expect the Fighter agent to gradually turn left toward the target heading of 315 in order to make the shortest possible turn.

5.
The Fighter agent made a gradual left turn in order to reach the target heading of 315.



6.
The test results are consistent with the expected results.

7.
Results from this test conflict with the results of the test A.10 and A.11.
The clean test refers to a heading change that does not require crossing the heading interval of [0, 360). Straddling tests do require the Fighter agent to cross this boundary, overlapping to the correct heading value.

Test A.13: Acceleration + Climb

1.
Tests to see if the Fighter agent is capable of increasing altitude while increasing speed.

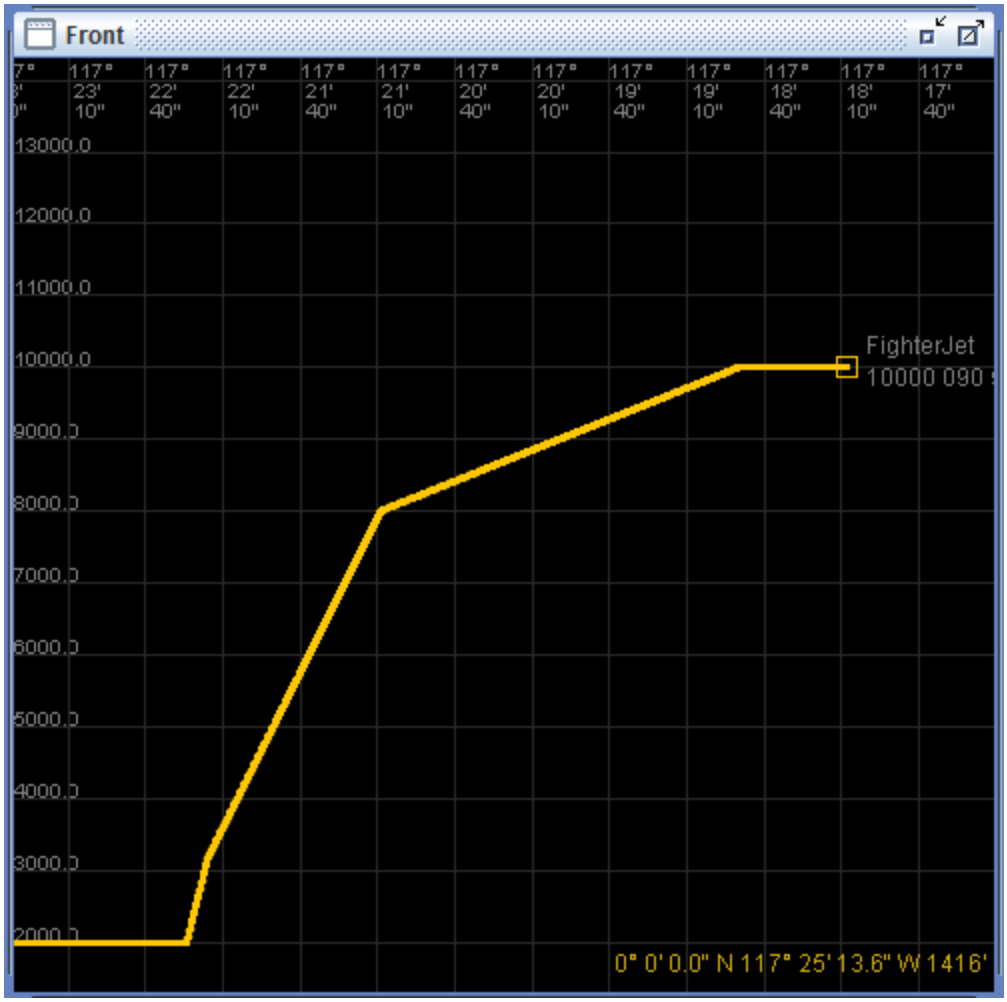
2.
From an altitude of 2000 and speed of 25, the Fighter agent must climb to an altitude of 10000 while accelerating to a speed of 50.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39'50"/117*25'00";@DO FighterJet FORCE ALTITUDE 2000;@DO FighterJet FORCE SPEED 25`

The following commands were used for the test:
`DO FighterJet SET ALTITUDE 10000;DO FighterJet SET SPEED 50`

4.
The Fighter agent will gradually accelerate to the target speed of 50 while slowly climbing to an altitude of 10,000

5.
The Fighter agent achieved the target acceleration and altitude gradually.



6.
The test results are consistent with the expected results.

7.
It is possible that there is an altitude limit enforced in the simulation. This could be tested by having the Fighter agent try to reach an incredibly high altitude.

Test A.14: Acceleration + Descent

1.
Tests to see if the Fighter agent is capable of decreasing altitude while increasing speed.

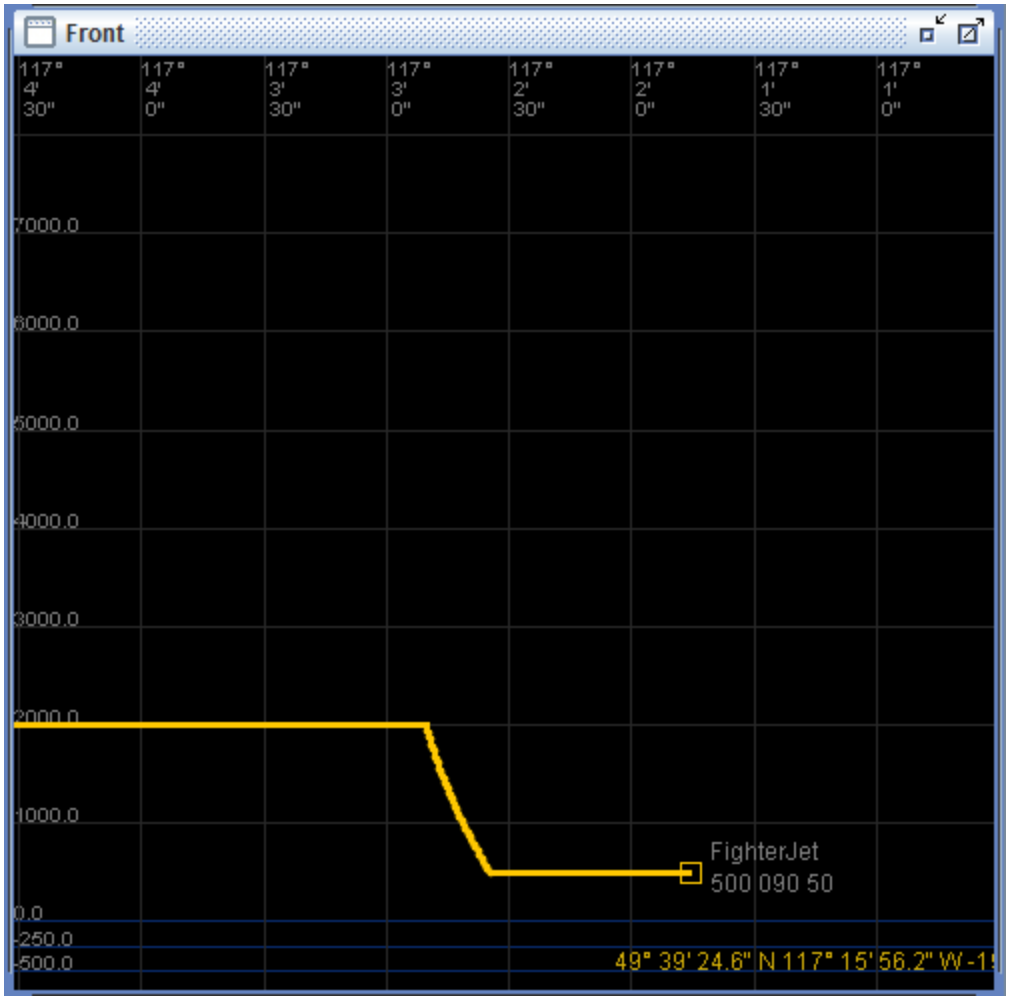
2.
From an altitude of 2000 and speed of 25, the Fighter agent must accelerate to a speed of 50 while descending to an altitude of 500.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE SPEED 25;@DO FighterJet FORCE ALTITUDE 2000`

The following commands were used for the test:
`DO FighterJet SET SPEED 50;DO FighterJet SET ALTITUDE 500`

4.
The Fighter agent will gradually accelerate to the target speed of 50 while slowly descending to altitude 500.

5.
The Fighter agent quickly reached the target speed and altitude.



6.
The Fighter agent achieved the target altitude and speed, but at an unexpectedly steep slope.

7.
The test could be expanded by attempting the same maneuver at a much higher base speed. Theoretically, the faster the agent is going, it should be more difficult to rapidly change altitude.

Test A.15: Deceleration + Climb

1.
Tests to see if the Fighter agent is capable of increasing altitude while decreasing speed.

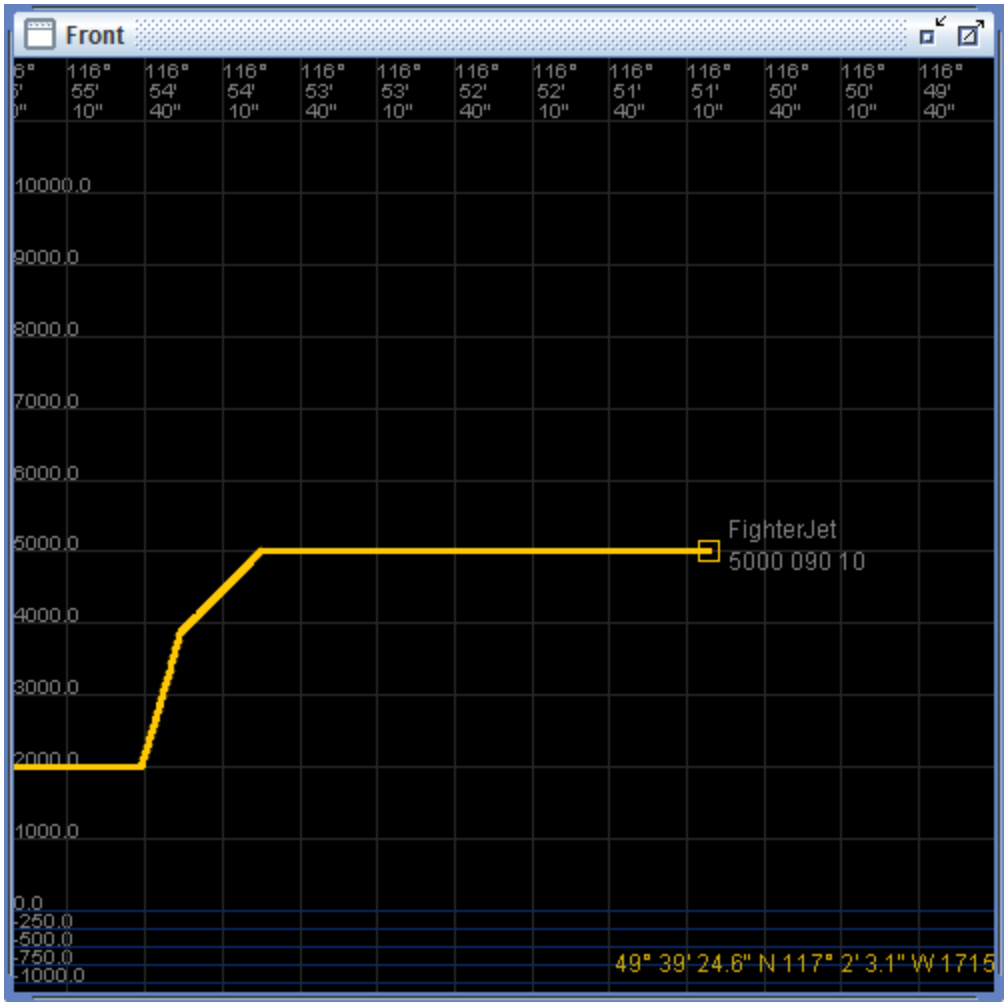
2.
From an altitude of 2000 and a speed of 25, the Fighter agent must decelerate to a speed of 10 while climbing to an altitude of 5000.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39'50"/117*25'00";@DO FighterJet FORCE ALTITUDE 2000;@DO FighterJet FORCE SPEED 25`

The following commands were used for the test:
`DO FighterJet SET SPEED 10;DO FighterJet SET ALTITUDE 5000`

4.
The Fighter agent will decelerate to the target speed of 10 while slowly climbing to an altitude of 5000.

5.
The Fighter agent achieved target altitude normally, but decelerated to the target speed of 10 much slower than expected.



6.
The Fighter agent achieved the target altitude and speed as expected, but took an extended amount of time decelerating to speed 10.

7.
Theoretically, a pilot could use an altitude climb to assist with deceleration. We could test how the simulation handles this by having the Fighter agent decelerate while performing a climb to a much higher altitude.

Test A.16: Deceleration + Descent

1.
Tests to see if the Fighter agent is capable of decreasing altitude while decreasing speed.

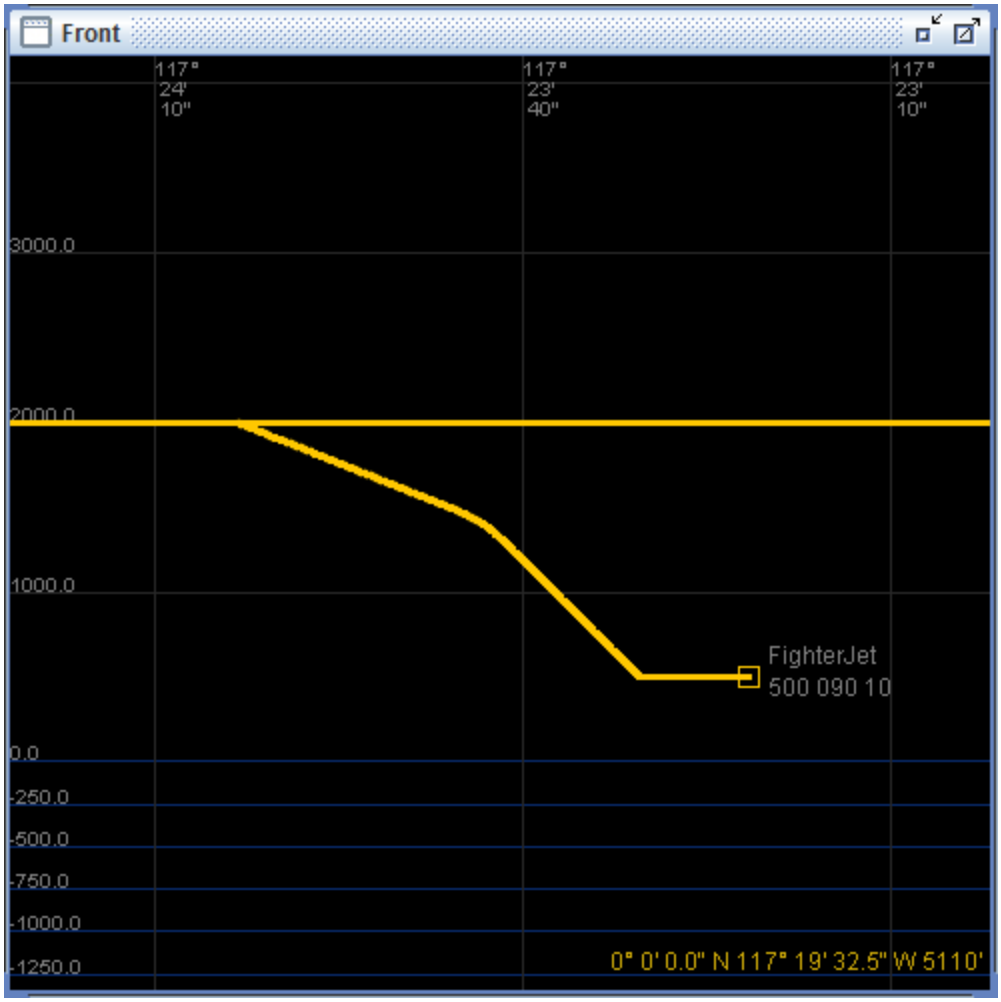
2.
From an altitude of 2000 and speed of 25, the Fighter agent must decelerate to a speed of 10 while descending to an altitude of 500.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE ALTITUDE 2000;@DO FighterJet FORCE SPEED 25`

The following commands were used for the test:
`DO FighterJet SET ALTITUDE 500;DO FighterJet SET SPEED 10`

4.
The Fighter agent will gradually make its descent while slowly reaching the target speed.

5.
The Fighter agent achieved the target speed and altitude as expected.



6.
Test results are consistent with the expected results.

7.
We could test to see if the Fighter agent would continue toward its target altitude while the Fighter agent slowly decelerates to 0.

Test A.17: Acceleration + Heading Change

1.
Tests to see if the Fighter agent is capable of increasing speed while changing heading.

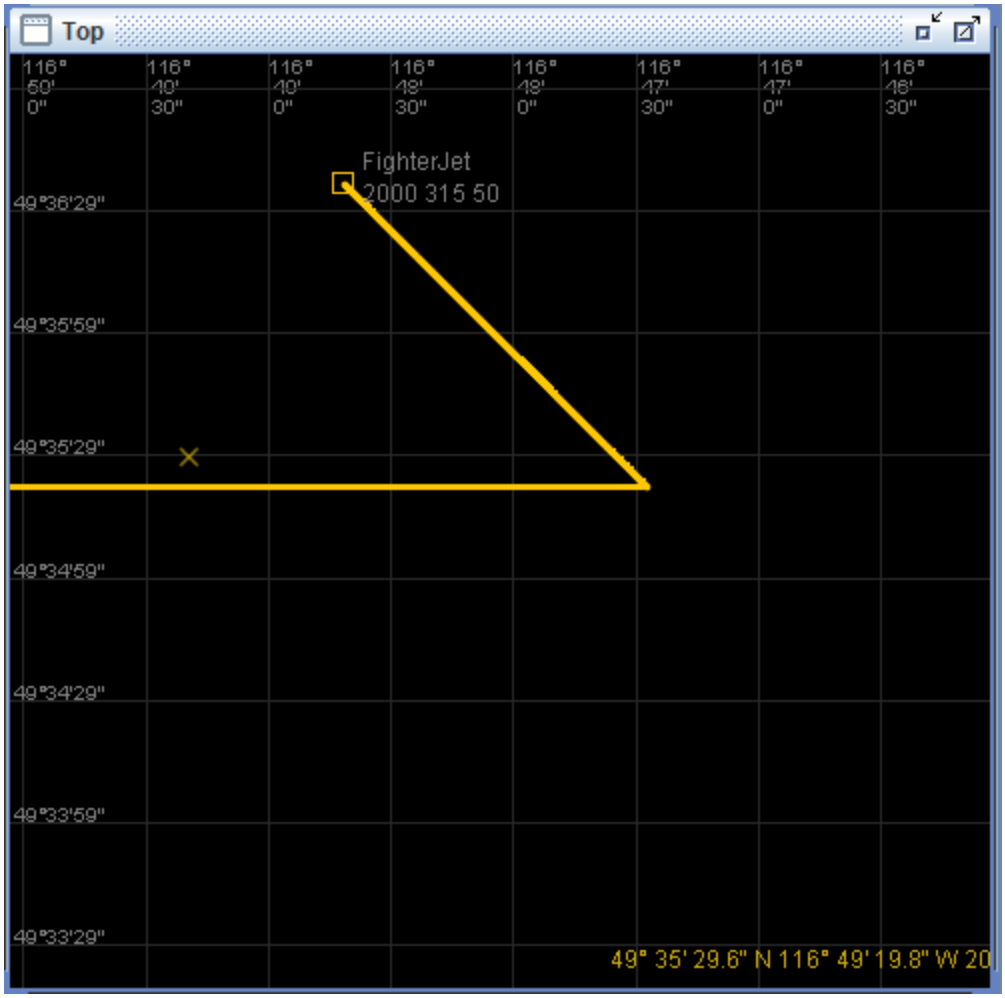
2.
From a speed of 25 and a heading of 90, the Fighter agent must accelerate to a speed of 50 while turning 135 degrees to the left.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00 and @DO FighterJet FORCE HEADING 090 and @DO FighterJet FORCE SPEED 25`

The following commands were used for the test:
`DO FighterJet SET SPEED 50; DO FighterJet SET HEADING 315`

4.
The Fighter agent will accelerate at a slow rate while turning to the left toward the new heading; acceleration will pick up after it has achieved the target heading.

5.
The Fighter instantly assumed the target heading and speed.



6.
The Fighter agent, once given the command to do so, immediately assumed the target heading of 315 and target speed of 50.

7.
We can further investigate this unusual behavior by trying to perform the same operation, but turning to the right instead of left.

Test A.18: Deceleration + Heading Change

1.
Tests to see if the Fighter agent is capable of decreasing speed while changing heading.

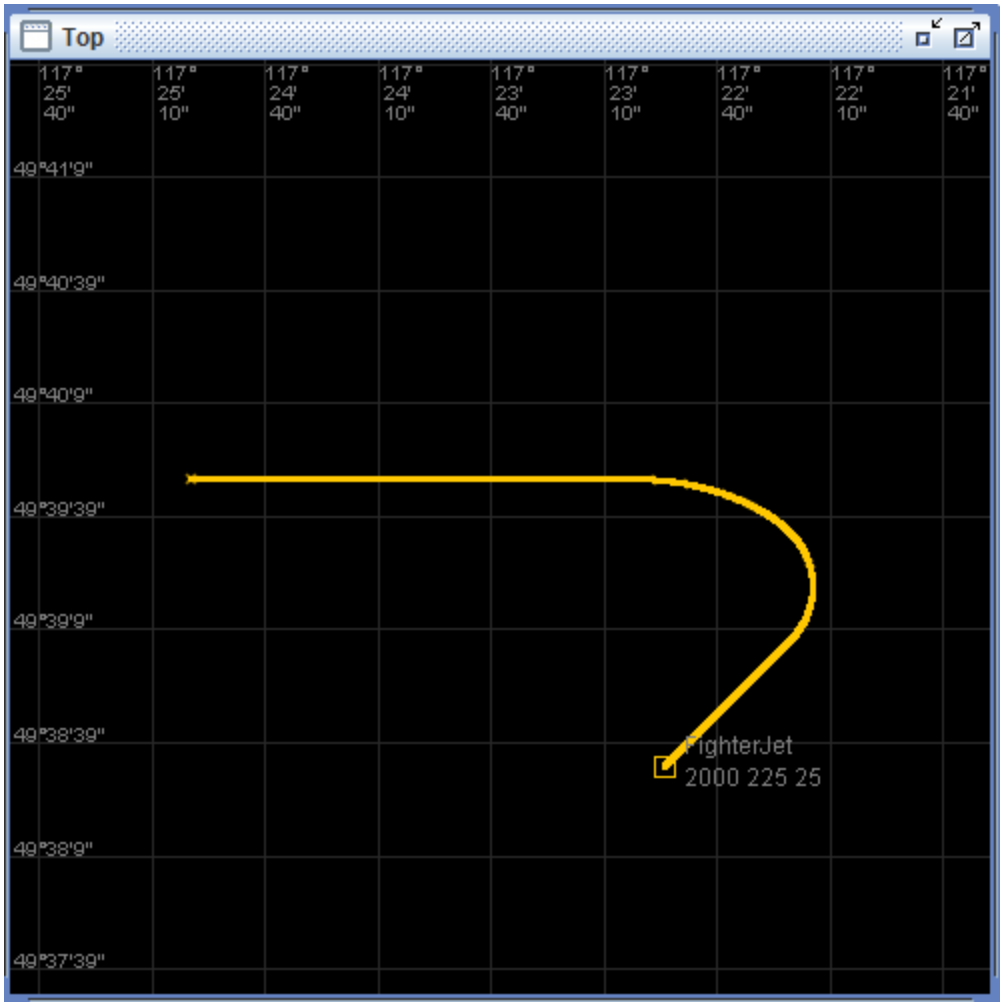
2.
From a speed of 100 and a heading of 90, the Fighter agent must decelerate to a speed of 25 while turning 135 degrees to the right.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39'50"/117*25'00";@DO FighterJet FORCE HEADING 090;@DO FighterJet FORCE SPEED 100`

The following commands were used for the test:
`DO FighterJet SET SPEED 25;DO FighterJet SET HEADING 225`

4.
The Fighter agent is expected to decelerate gradually to the target speed while it turns right toward the target heading.

5.
The Fighter agent successfully decelerated to the target speed while turning towards the target heading.



6.
The test results are consistent with the expected results.

7.
A test could be conducted on the behavior of a right turn while decelerating toward a speed of 0.

Test A.19: Climb + Heading Change

1.
Tests to see if the Fighter agent is capable of increasing altitude while changing heading.

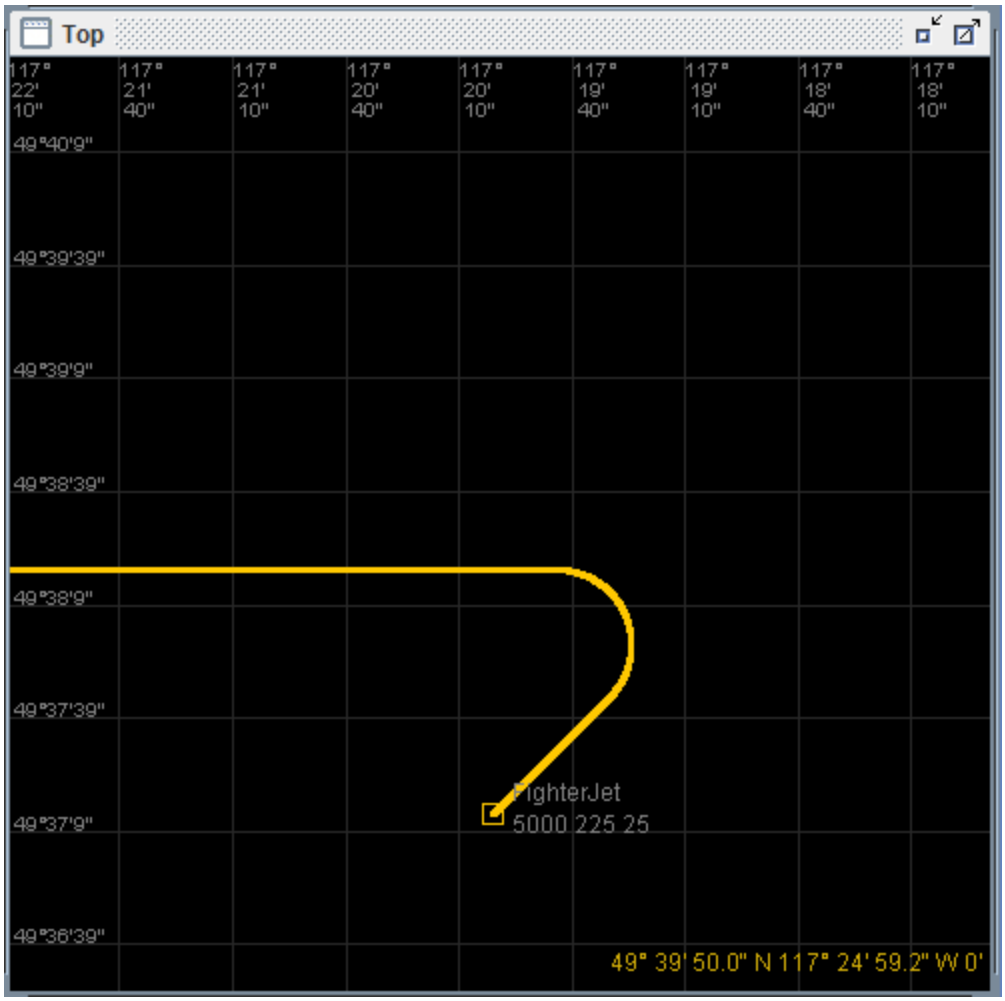
2.
From an altitude of 2000 and a heading of 90, the Fighter agent must climb to an altitude of 5000 while turning 135 degrees to the right.

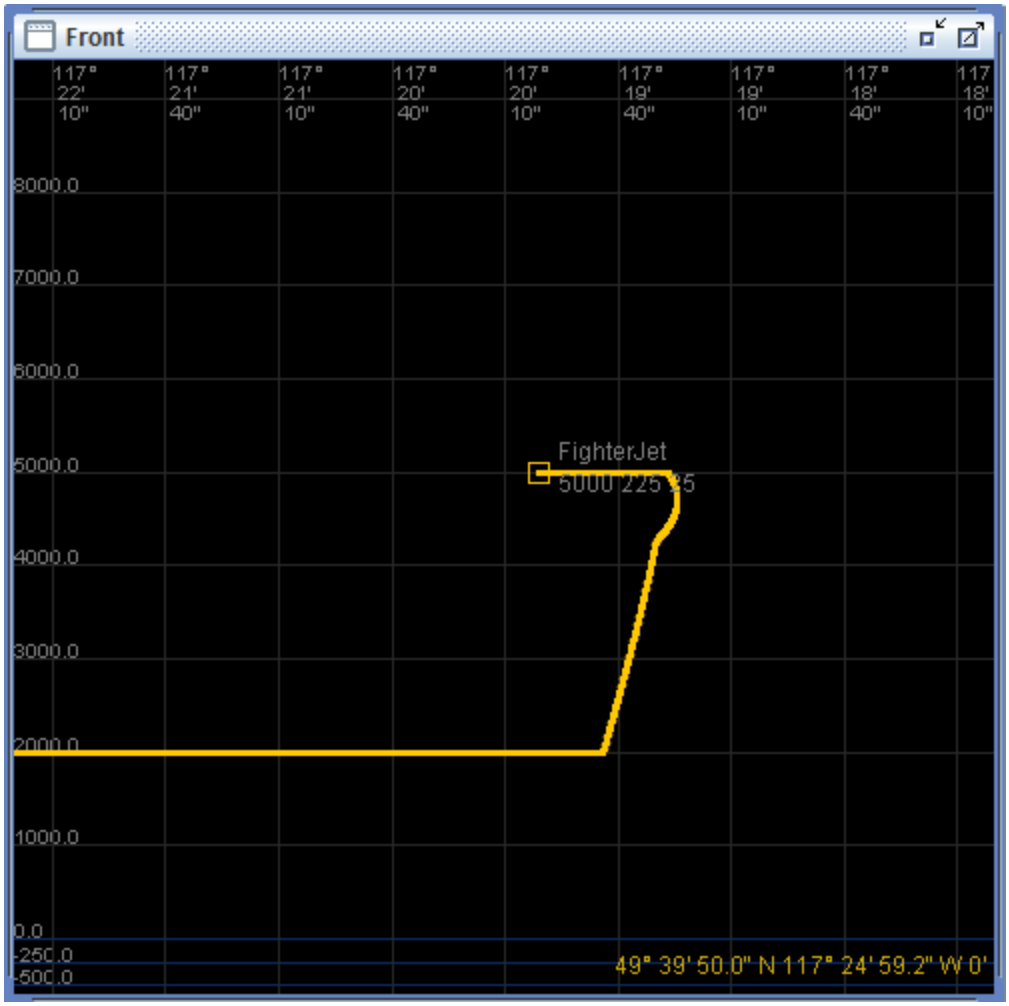
3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE ALTITUDE 2000;@DO FighterJet FORCE HEADING 090`

The following commands were used for the test:
`DO FighterJet SET HEADING 225 RIGHT; DO FighterJet SET ALTITUDE 5000`

4.
The Fighter agent is expected to be capable of changing heading to the right while increasing altitude.

5.
The Fighter agent successfully changed to the target heading while climbing to the target altitude.





6.
The test results are consistent with the expected results.

7.
Since speed directly affects a Fighter agent's ability to turn, we could test the behavior of a Fighter agent performing the same maneuver at a much higher speed.

Test A.20: Acceleration + Climb + Heading Change

1.
Tests to see if the Fighter agent is capable of increasing altitude and speed while changing heading.

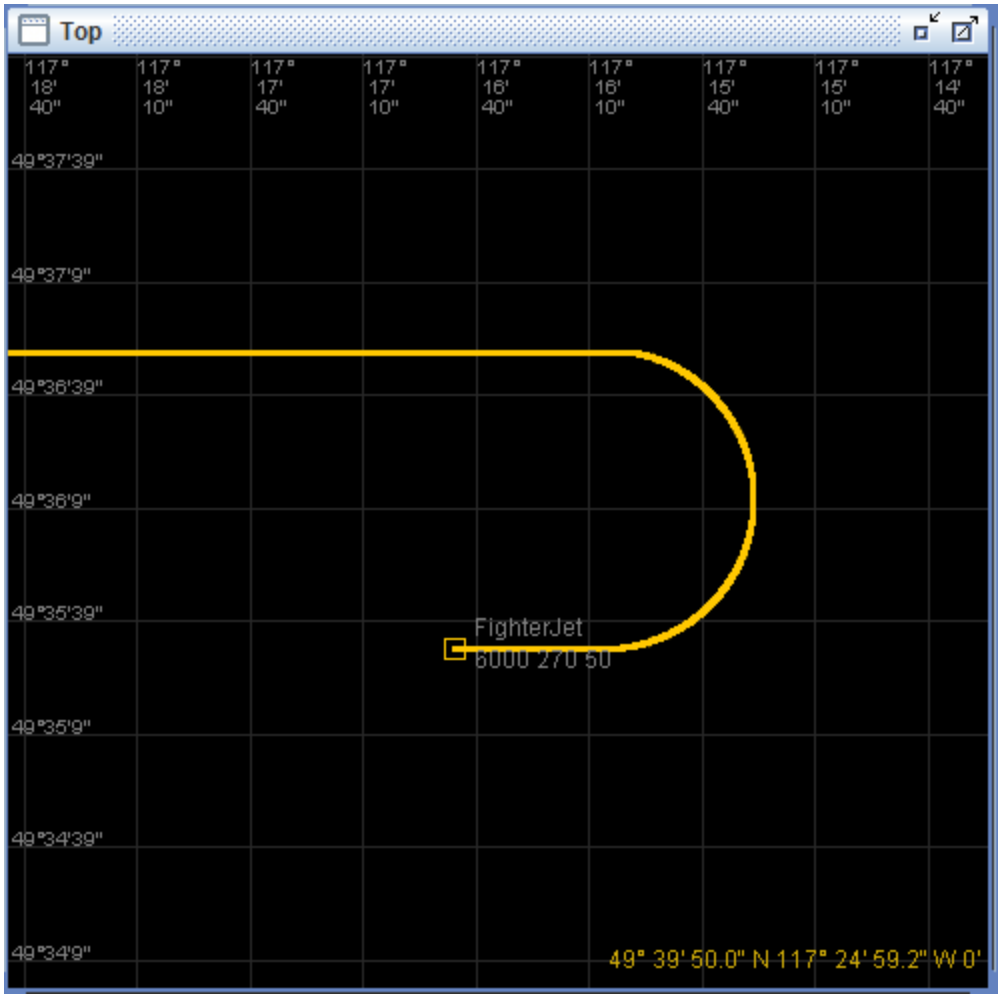
2.
From an altitude of 2000, speed of 25, and heading of 90, the Fighter agent must climb to an altitude of 6000, accelerate to a speed of 50, and turn 180 degrees to the right simultaneously.

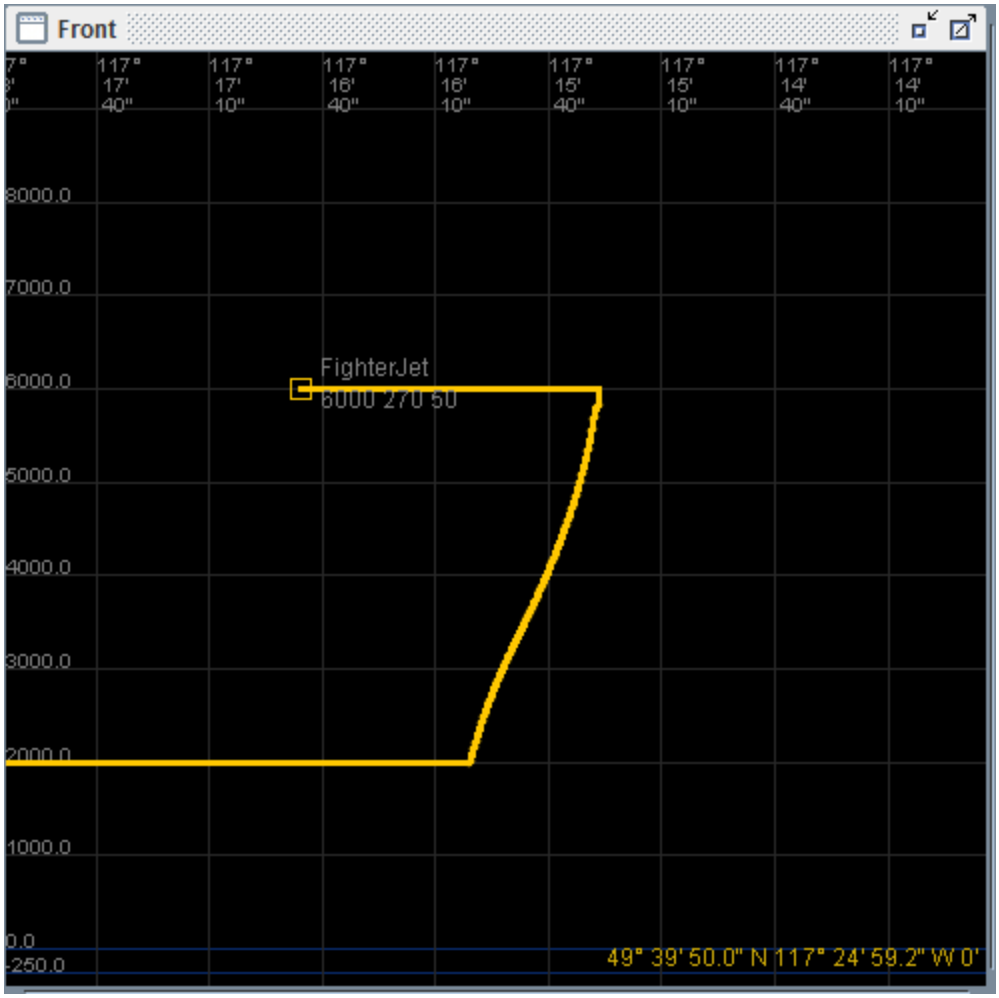
3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE SPEED 25;@DO FighterJet FORCE HEADING 090;@DO FighterJet FORCE ALTITUDE 2000`

The following commands were used for the test:
`DO FighterJet SET ALTITUDE 6000;DO FighterJet SET SPEED 50;DO FighterJet SET HEADING 270 RIGHT`

4.
The Fighter agent is expected to be able to increase altitude, increase speed, and change heading simultaneously.

5.
The Fighter agent accelerated to reach the target speed, climbed to the target altitude, and turned to reach the target heading simultaneously.





6.
The test results are consistent with the expected results.

7.
If we can climb, accelerate, and change heading; we can assume that the Fighter agent is capable of descending, decelerating, and changing heading as well. However, this can be tested to guarantee that it works.

Test A.21: Deceleration + Climb + Heading Change

1.
Tests to see if the Fighter agent is capable of increasing altitude, decreasing speed, and changing heading simultaneously.

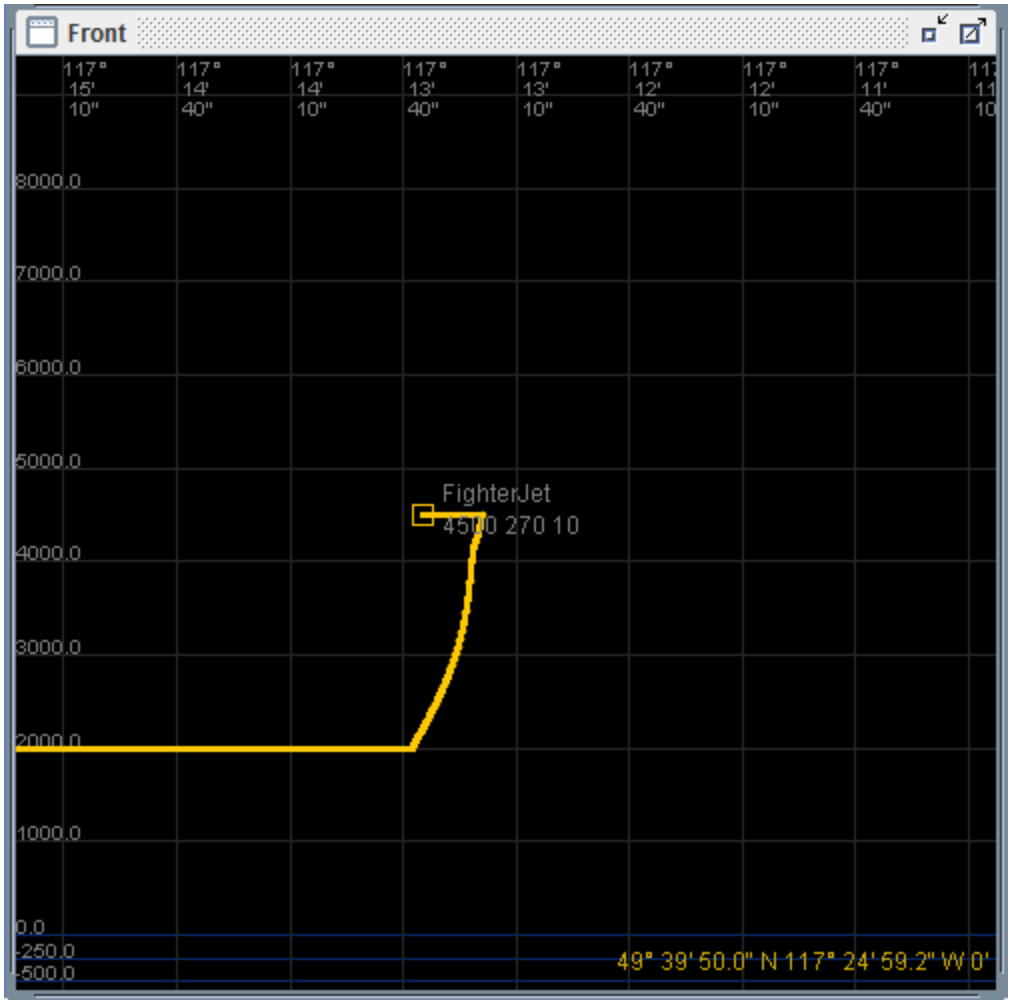
2.
From an initial speed of 50, altitude of 2000, and heading of 90; the Fighter agent must decelerate to a speed of 10, climb to altitude 4500, and turn 180 degrees to the right simultaneously.

3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”;@DO FighterJet FORCE SPEED 50;@DO FighterJet FORCE HEADING 090;@DO FighterJet FORCE ALTITUDE 2000`

The following commands were used for the test:
`DO FighterJet SET SPEED 10;DO FighterJet SET ALTITUDE 4500;DO FighterJet SET HEADING 270 RIGHT`

4.
The Fighter agent is expected to decelerate, climb, and change heading simultaneously.

5.
The Fighter agent performed as expecting, performing all the operations simultaneously, resulting in a tight right turn at an upward angle.



6.
The test results are consistent with the expected results.

7.
It appears that the Fighter agent was able to make a tighter turn toward the target heading because it was not only decelerating, but also climbing to a higher altitude.

Test A.22: Tailhook

1.
Tests to see if the Fighter agent's tailhook is functioning correctly.

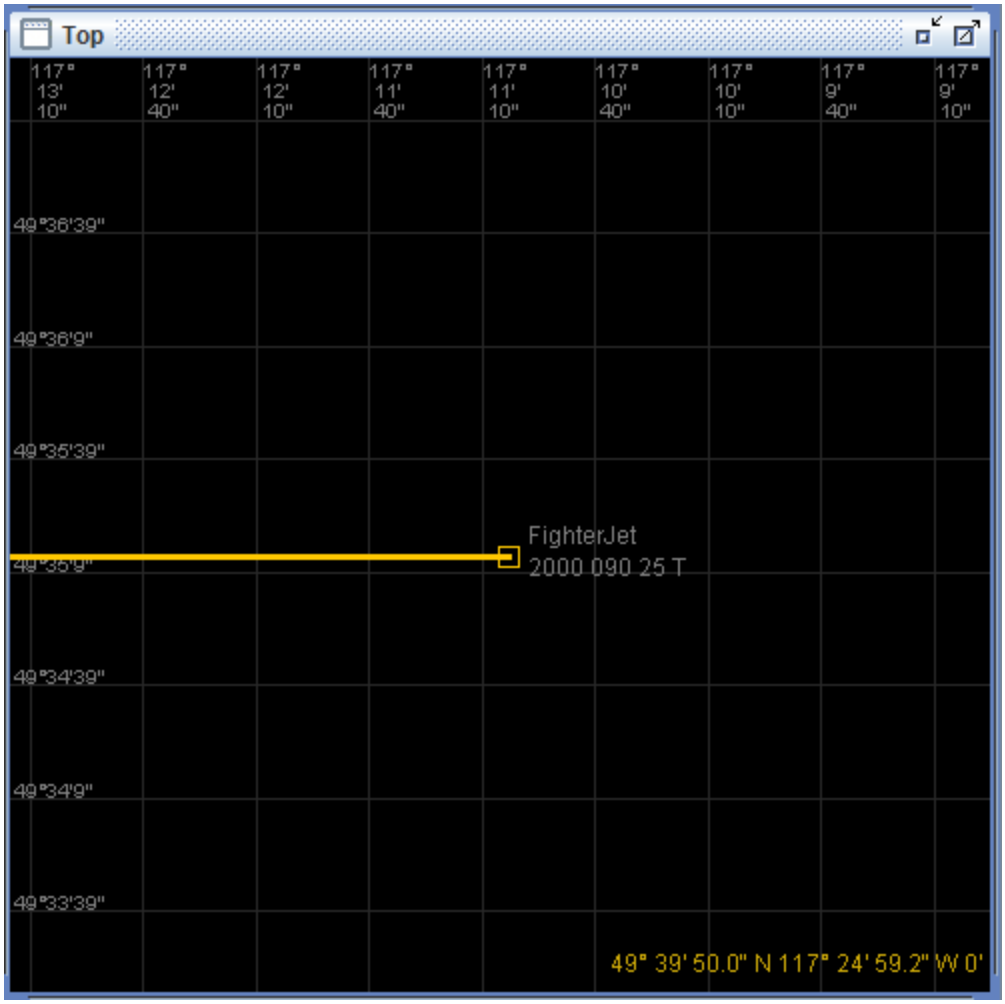
2.
The Fighter agent must successfully lower and raise the tailhook.

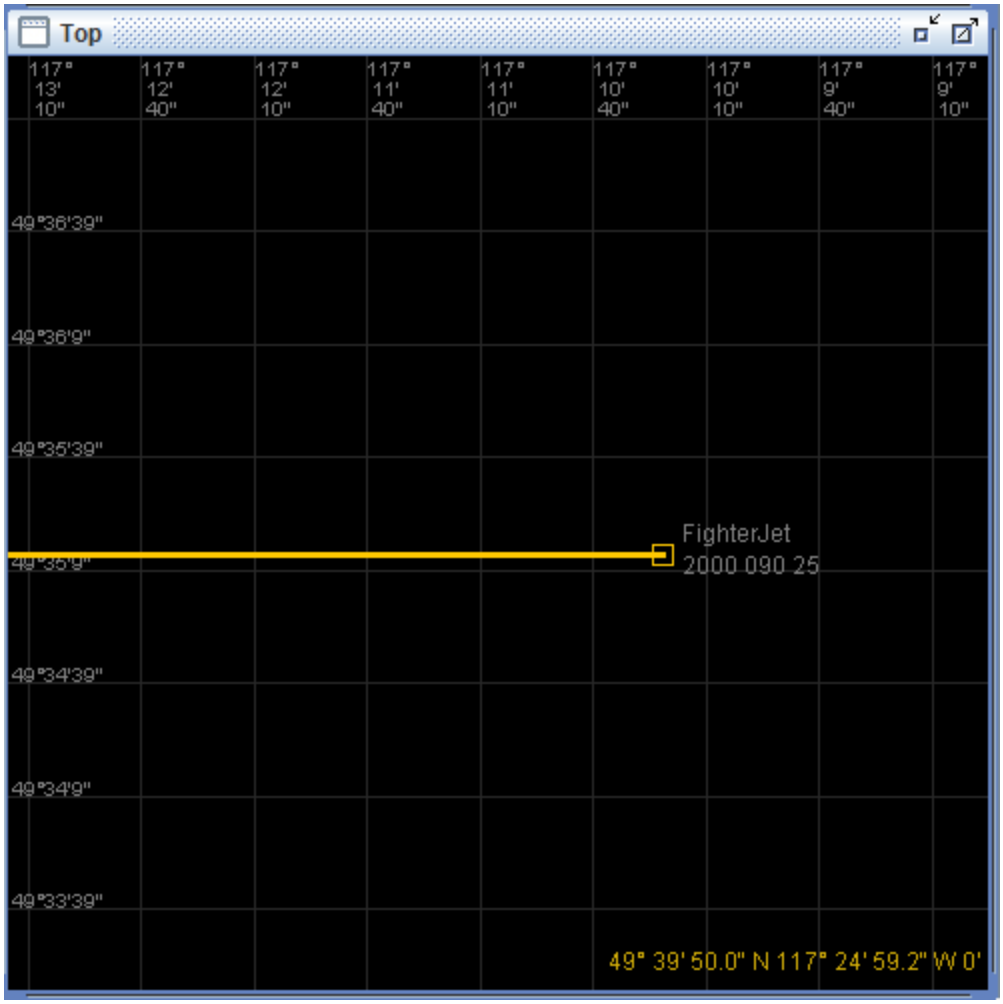
3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”`
The following commands were used for the test:

```
DO FighterJet TAILHOOK DOWN

DO FighterJet TAILHOOK UP
```

4.
The Fighter agent should lower and raise the tailhook as specified.

5.
The Fighter agent successfully lowered and raised its tailhook.





6.
Test results are consistent with the expected results.

7.
Testing could be expanded by trying to lower the tailhook when it has already been lowered, or raise the tailhook when it is already raised.

Test A.23: Boom

1.
Tests to see if the Fighter agent's boom is functioning correctly.

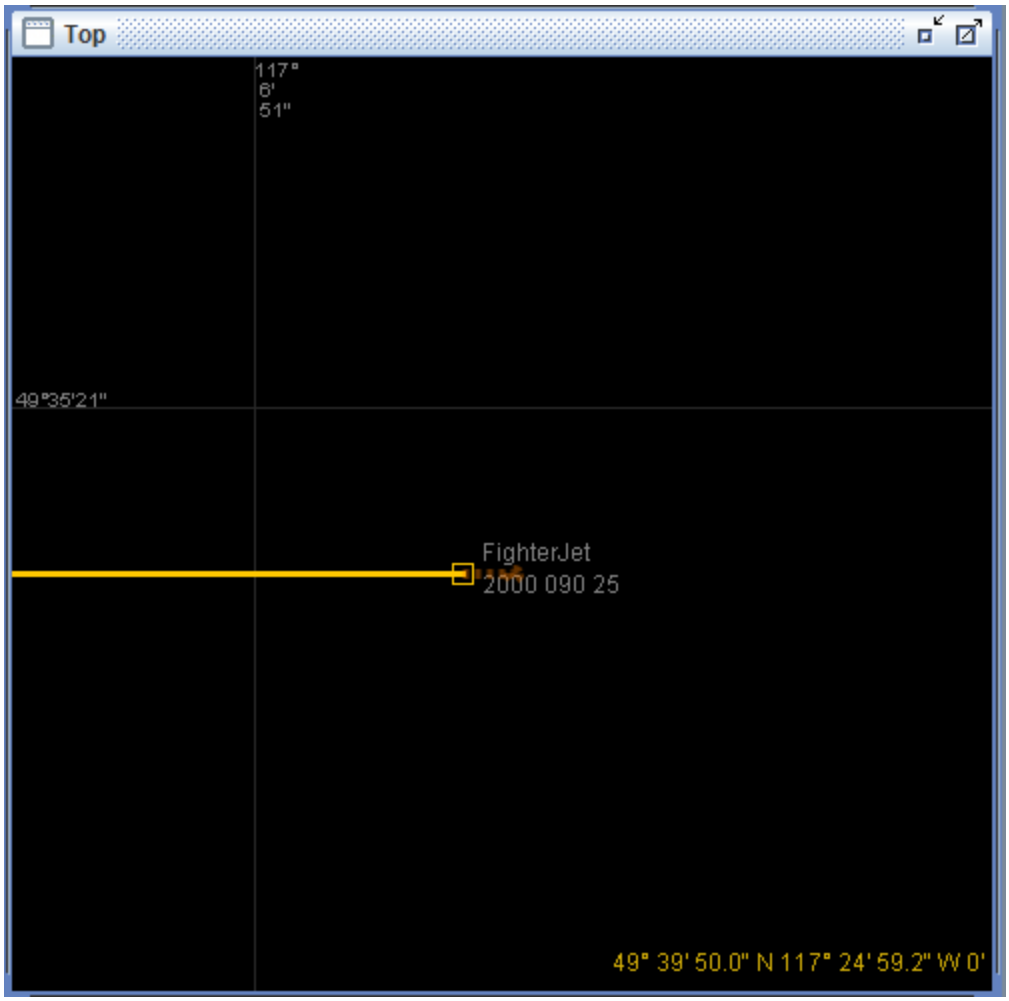
2.
The Fighter agent must successfully extend and retract its boom mechanism.

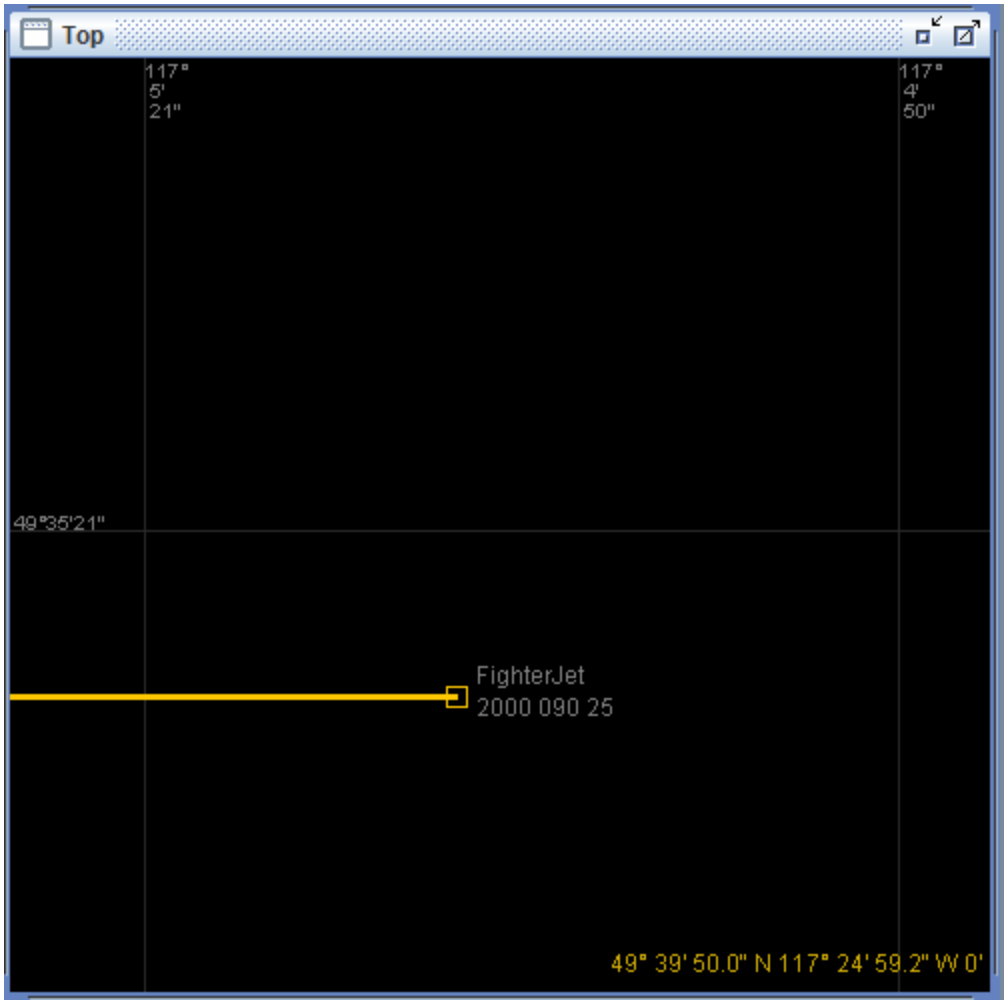
3.
For Fighter agent setup, refer to **A.1.3.**
For test reset, use: `@DO FighterJet FORCE COORDINATES 49*39’50”/117*25’00”`
The following commands were used for the test:

```
DO FighterJet BOOM EXTEND

DO FighterJet BOOM RETRACT
```

4.
The Fighter agent is expected to be able to successfully extend and retract its boom.

5.
The Fighter agent successfully extended and retracted its boom.





6.
The test results are consistent with the expected results.

7.
We could test the behavior of the boom by ordering the Fighter agent to extend the boom when it is already extended, or alternatively tell the agent to retract the boom when it has not been extended.

B. Tanker Tests

Test 1: Tanker Definition

1.
Tests to see whether a tanker can be properly defined and created.

2.
This test defines a tanker with the following parameters:

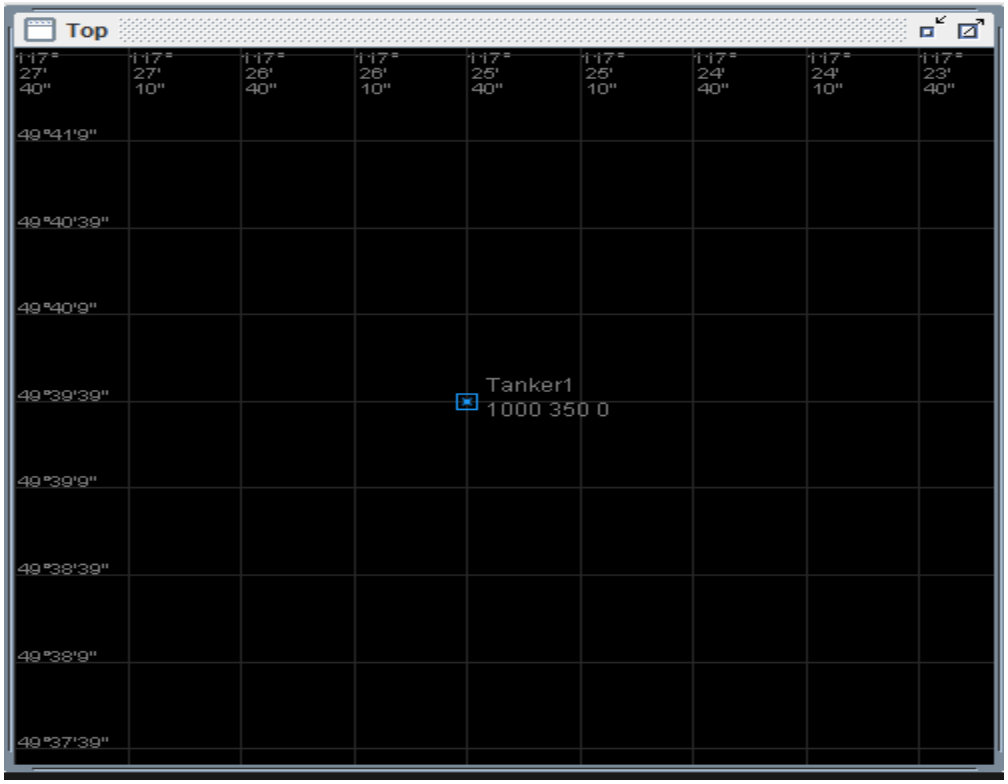
tanker: speed min **10** max **20** delta increase **30** decrease **40** turn **50** climb **50** descent **60** tank **70**
boom: tboomf1 length **200** diameter **20** elevation **20** flow **40**
initial: **49*39'40"/117*25'40"** altitude **1000** heading **350** speed **0**

3.
The following commands were used for the test:

```
DEFINE TANKER T_1 SPEED MIN 10 MAX 20 DELTA INCREASE 30 DECREASE 40 TURN 50
CLIMB 50 DESCENT 60 TANK 70
DEFINE BOOM FEMALE tboomf1 LENGTH 200 DIAMETER 20 ELEVATION 20 FLOW 50

CREATE BOOM FemaleBoom FROM tboomf1
CREATE TANKER Tanker1 FROM T_1 WITH BOOM FemaleBoom AT COORDINATES
49*39’40”/117*25’40” ALTITUDE 1000 HEADING 350 SPEED 0

POPULATE WORLD WITH Tanker

COMMIT
```

4.
The test is expected to populate the simulation world with the tanker and its secondary agents as defined and starting at the coordinates specified in **B.1.2.**

5.
Tanker1 populated the world at coordinates 49*39’40”/117*25’40” with an altitude of 1000, a heading of 350 and a speed of 0.



6.
The test results are consistent with the expected test results.

7.
This test could be extended by creating multiple tankers, attempting to use the same booms as each other and sharing names to see what happens.

Test 2: Boom

1.
Test whether the boom can be extended or retracted.

2.
This tests should extend then retract the boom

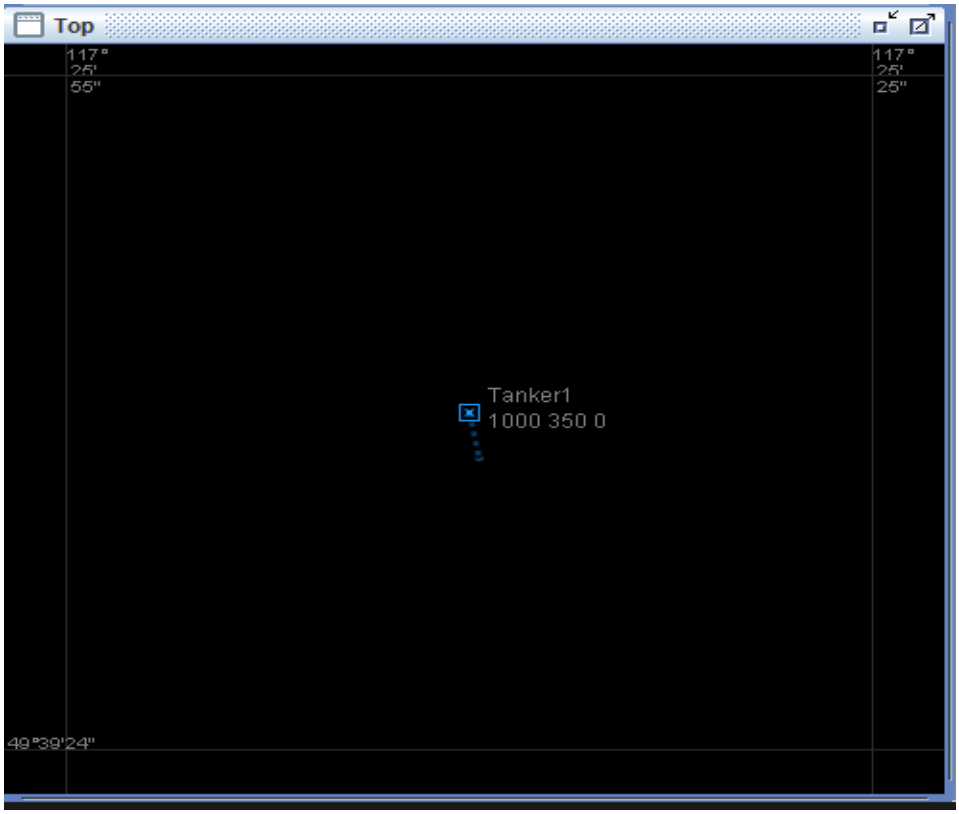
3.
For tanker agent setup refer to **B.1.3**

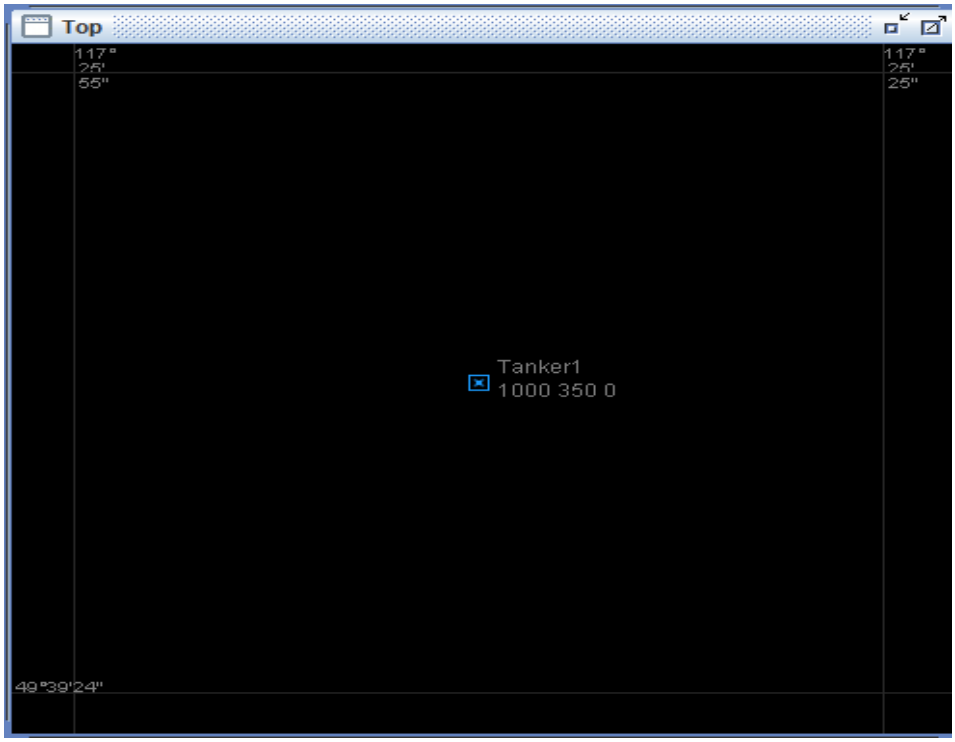
The following commands were used for the test:

```
DO Tanker1 BOOM EXTEND
DO Tanker1 BOOM RETRACT
```

4.
The tanker is expected to extend a boom like device from its underside. It is then expected to retract this boom like device.

5.
For the extend command the tanker extended a boom like device. For the retract command the tanker retracted the boom like device.





6.
The actual results did not differ from the expected results.

7.
This test could be extended by trying to retract the tanker before it is all the way extended, by trying to retract or extend an already extended or retracted boom.

C. Carrier Tests

Test 1: Carrier Definition

1.
This test's purpose is to check if the define/create commands work properly with a carrier.

2.
We will define and create the following carrier:
Carrier: speed max **10** delta increase **20** decrease **30** turn **40** layout '**carrier.acg**'
Catapult: origin **+4.4:-141** azimuth **0** length **400** acceleration **20** limit weight **50** speed **45** reset **20**
Barrier: origin **+2.4:-119.5** azimuth **0** width **60** time **50**
Trap: origin **-14.9:+256.3** azimuth **-8.5** width **400** limit weight **50** speed **35** miss **30**
OLS: origin **-14.9:+261.3** azimuth **172** elevation **5** range **10000** diameter **500**
Setup: coordinates **49*39'00"/117*26'00"** heading **0** speed **0**

3.

The following commands were used for the test:

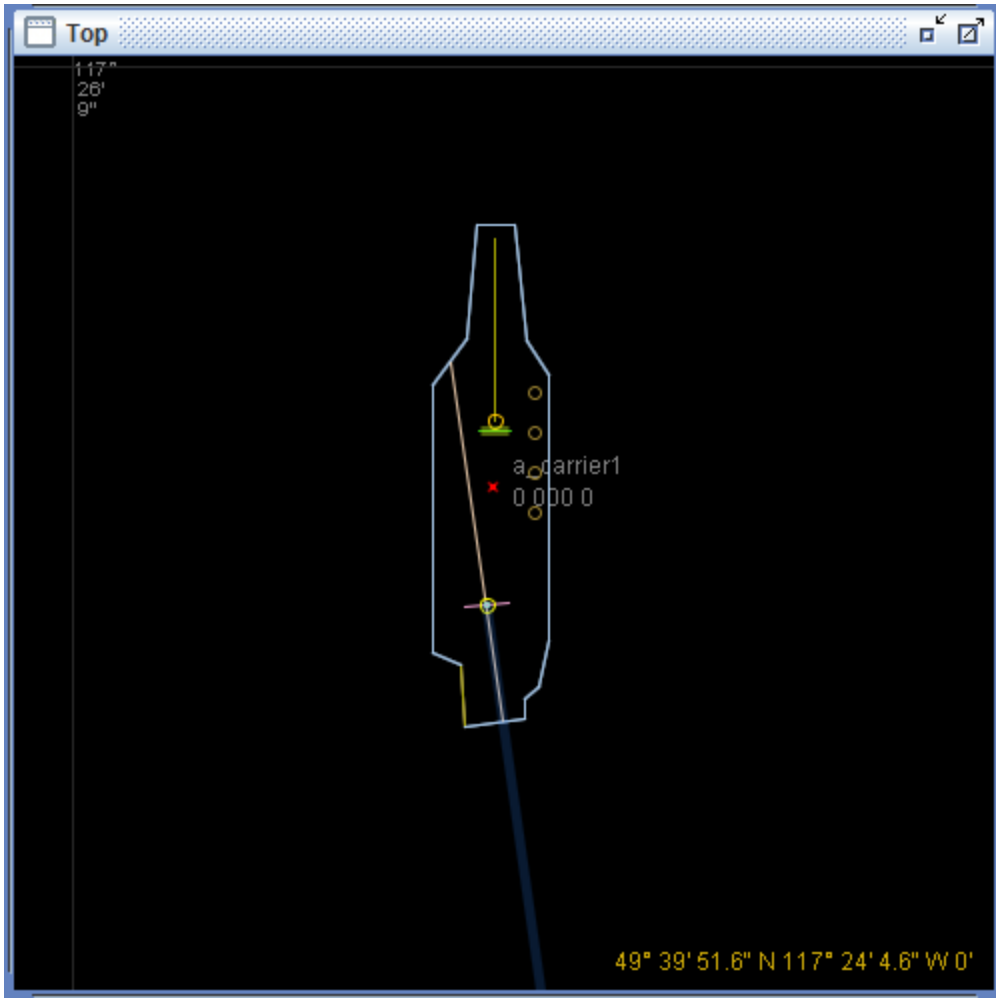
```
DEFINE CARRIER t_carrier1 SPEED MAX 10 DELTA INCREASE 20 DECREASE 30 TURN 40
LAYOUT 'carrier.acg'
DEFINE CATAPULT t_cat1 ORIGIN +4.4:-141 AZIMUTH 0 LENGTH 400 ACCELERATION 20
LIMIT WEIGHT 50 SPEED 45 RESET 20
DEFINE BARRIER t_bar1 ORIGIN +2.4:-119.5 AZIMUTH 0 WIDTH 60 TIME 50
DEFINE TRAP t_trap1 ORIGIN -14.9:+256.3 AZIMUTH -8.5 WIDTH 400 LIMIT WEIGHT
50 SPEED 35 MISS 30
DEFINE OLS_XMT t_ols1 ORIGIN -14.9:+261.3 AZIMUTH 172 ELEVATION 5 RANGE
10000 DIAMETER 500
CREATE CATAPULT a_cat1 FROM t_cat1
CREATE BARRIER a_bar1 FROM t_bar1
CREATE TRAP a_trap1 FROM t_trap1
CREATE OLS_XMT a_ols1 FROM t_ols1
CREATE BARRIER a_carrier1 FROM t_carrier1 WITH CATAPULT a_cat1 BARRIER
a_bar1 TRAP a_trap1 OLS a_ols1 AT COORDINATES 49*39'00"/117*26'00" HEADING
000 SPEED 0
POPULATE WORLD WITH a_carrier1
COMMIT
```

4.

This should produce a carrier at coordinates `49*39'00"/117*26'00"`

5.

A carrier was created at coordinates `49*39'00"/117*26'00" WITH SPEED 0 AND HEADING 000`



6.
The results don't differ from expected.

7.
We can expand this test by trying to create a second carrier that uses the same ols, trap, barrier and catapult as the first carrier.

Test 2: Barrier

1.
This test checks if the barrier on a carrier works properly.

2.
We start with the carrier from C1 and we enter the command to control the barrier.

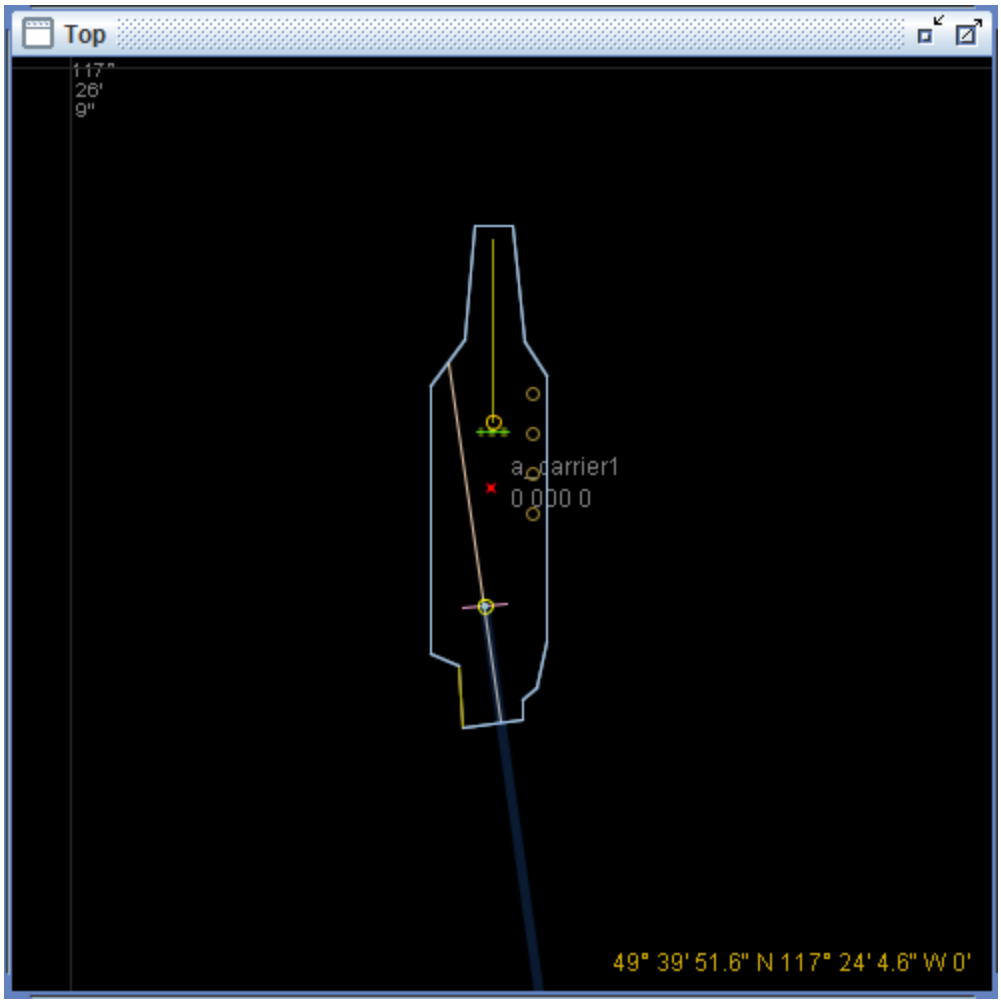
3.
For Carrier agent setup, refer to **C.1.3**

The following commands were used for the test:

```
DO a_carrier1 BARRIER UP
DO a_carrier1 BARRIER DOWN
```

4.
These commands should allow to change the state of the barrier.

5.
The barrier displayed a little moving icon whenever a barrier command was entered



6.
The results didn't differ from expected.

7.
We could expand this test by trying to raise the barrier when it's already up or lower it when it's already down.

D. Fighter-Tanker Tests

Test D.1: Refueling 1

1.
This test verifies the process of refueling a fighter from a tanker. The fighter needs to couple with the tanker and maintain altitude, speed, and heading.

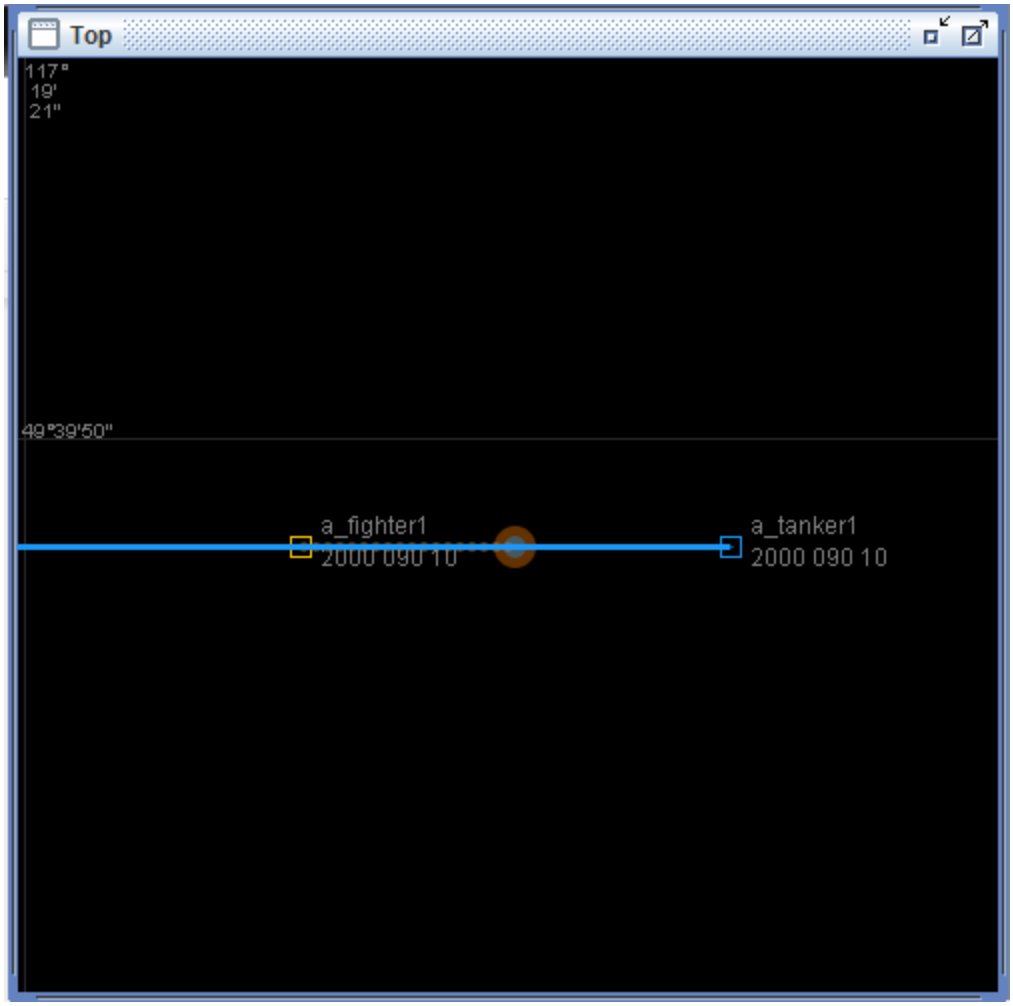
2.
The test starts out by defining and creating a tanker and a fighter and placing them on the same trajectory while having the tanker in front of the fighter. We then extend the booms on both agents and we speed up the fighter enough to connect the boom. Once coupled, the agents keep the same speed.

3.
The following commands were used for the test:

```
DEFINE FIGHTER t_fighter1 SPEED MIN 10 MAX 20 DELTA INCREASE 40 DECREASE 35
TURN 15 CLIMB 10 DESCENT 20 EMPTY WEIGHT 55 FUEL INITIAL 89 DELTA 32
DEFINE OLS_RCV t_olsrcv1 DIAMETER 10
DEFINE BOOM MALE t_boomm1 LENGTH 200 DIAMETER 40 FLOW 40
DEFINE TAILHOOK t_tailhook1 TIME 10
DEFINE AUX_TANK t_auxtank1 AMOUNT 50
CREATE OLS_RCV a_olsrcv1 FROM t_olsrcv1
CREATE BOOM a_boomm1 FROM t_boomm1
CREATE TAILHOOK a_tailhook1 FROM t_tailhook1
CREATE AUX_TANK a_auxtank1 FROM t_auxtank1
CREATE FIGHTER a_fighter1 FROM t_fighter1 WITH OLS a_olsrcv1 BOOM a_boomm1
TAILHOOK a_tailhook1 TANKS a_auxtank1 AT COORDINATES 49*39'50"/117*25'40"
ALTITUDE 2000 HEADING 090 SPEED 10
DEFINE TANKER T_1 SPEED MIN 10 MAX 20 DELTA INCREASE 30 DECREASE 40 TURN 50
CLIMB 50 DESCENT 60 TANK 70
DEFINE BOOM FEMALE tboomf1 LENGTH 200 DIAMETER 20 ELEVATION 20 FLOW 50
CREATE BOOM a_boomf1 FROM tboomf1
CREATE TANKER a_tanker1 FROM T_1 WITH BOOM a_boomf1 AT COORDINATES
49*39'50"/117*25'0" ALTITUDE 2000 HEADING 090 SPEED 10
COMMIT
DO a_fighter1 BOOM EXTEND
DO a_tanker1 BOOM EXTEND
DO a_fighter1 SET SPEED 15
DO a_fighter1 SET SPEED 10
DO a_tanker1 TRANSFER START
```

4.
When we speed up the fighter the booms will get close enough and once they connect we should get something indicating it.

5.
Once the booms connect the circles close on each other and we can begin fueling.



6.
The results don’t differ from expected

7.
We can expand this test and try to change the altitude of one of the agents during the fueling process and see how that affects the process.

Test D.2: Refueling 2

1.

This test builds off test **D.1** but we want to see what would happen if we have the tanker pull ahead slowly until the coupling breaks.

2.

The test starts where we left off in test **D.1** but now we increase the tanker speed until the coupling breaks.

3.

This test is set up exactly like test **D.1**, with the addition of the following commands:

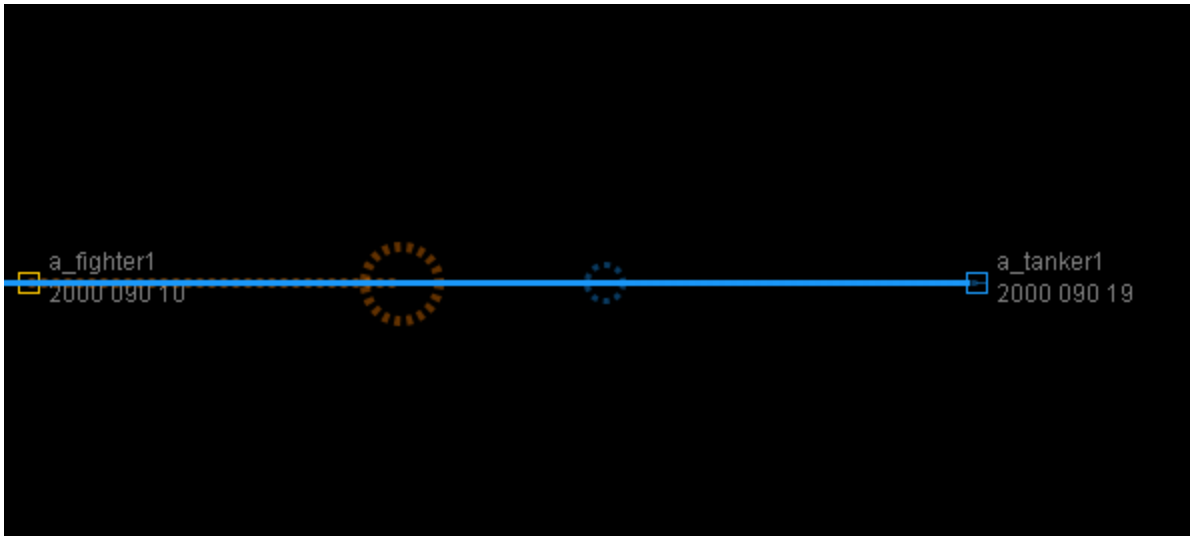
```
DO a_tanker1 SET SPEED 11
```

4.

We expect the connection to break and possibly have an error stating that the fueling process was interrupted.

5.

The coupling did break as expected.



6.

We did not get any error stating that the fueling process was interrupted.

7.

We could expand this test to check what would happen if we try to play with tanker's fuel transfer right after the coupling breaks

Test D.3: Refueling 3

1.

Similar to test **D.2**, now we want to see what would happen to the fueling if we have the tanker turn away until the coupling breaks.

2.

We start with same setup from test **D.1** and we change the heading of the tanker until the coupling breaks.

3.

This test is set up exactly like test **D.1**, with the addition of the following commands:

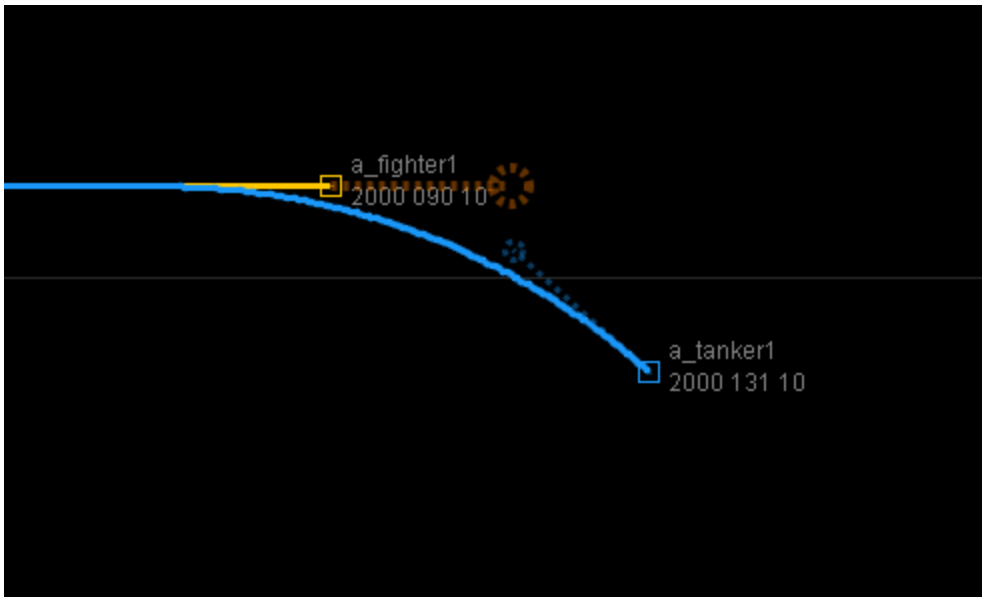
```
DO a_tanker1 SET HEADING 180 RIGHT
```

4.

Based on our results from test **D.2,** we expect the coupling to just break without any further reactions.

5.

The tanker changed heading and as the distance got larger the connection started breaking.



6.

The results didn't differ.

7.

We could expand this test to check what would happen if we try to play with tanker's fuel transfer right after the coupling breaks

Test D.4: Refueling 4

1.

Similar to the previous tests **D.2** and **D.3**, we now want to see what happens if we lower the aircraft speed until the coupling breaks

2.

We start with same setup from **D.1** and we decrease the speed of the fighter.

3.

This test is set up exactly like test **D.1**, with the addition of the following commands:

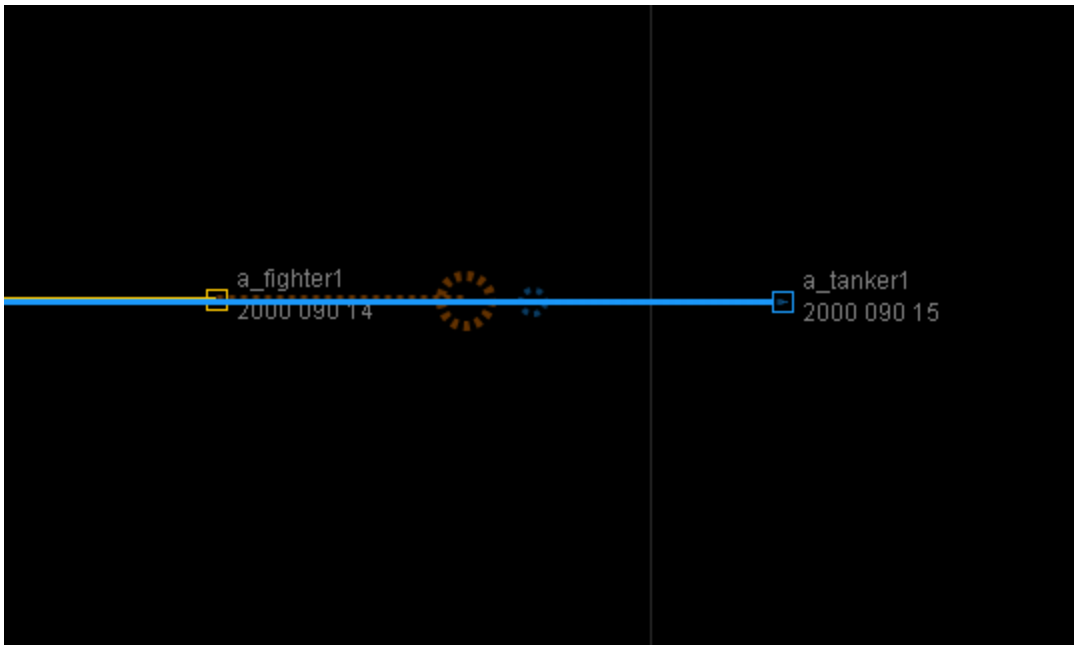
```
DO a_fighter1 SET SPEED 9
```

4.

Based on previous experiences, the coupling should break with no further consequences.

5.

The coupling broke but nothing else occurred.



6.

The results don't differ from expected.

7.

We could expand this test to check what would happen if we try to play with the tanker's fuel transfer right after the coupling breaks

Test D.5: Refueling 5

1.

Similar to the previous tests **D.2**, **D.3** and **D.4**, now we want to see what would happen when we change the heading of the fighter during the coupling.

2.

We start with the setup from **D.1** and we change the heading of the fighter.

3.

This test is set up exactly like test **D.1**, with the addition of the following commands:

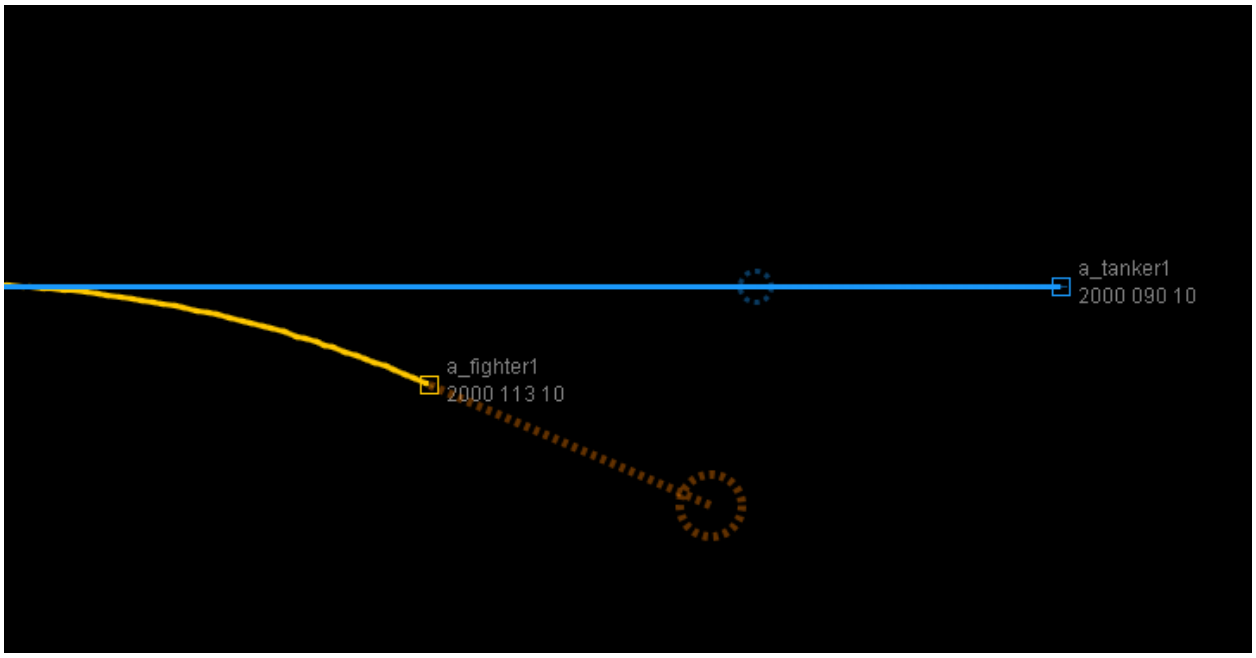
```
DO a_fighter SET HEADING 180
```

4.

Based on previous experience the coupling should break without any further consequences.

5.

The coupling broke as expected but nothing else happened.



6.

The results don't differ from expected.

7.

We could expand this test to check what would happen if we try to play with tanker's fuel transfer right after the coupling breaks

E. Fighter-Fighter Tests

Test E.1: Refueling Attempt

1.

Test to see whether a fighter can refuel another fighter in the same manner that a tanker can refuel a fighter, as seen in test **D.1**.

2.

Refuel a fighter from a fighter. The fighter needs to couple with the fighter and maintain altitude, speed, and heading. Follow **A.1.2** as a template for fighter definations.

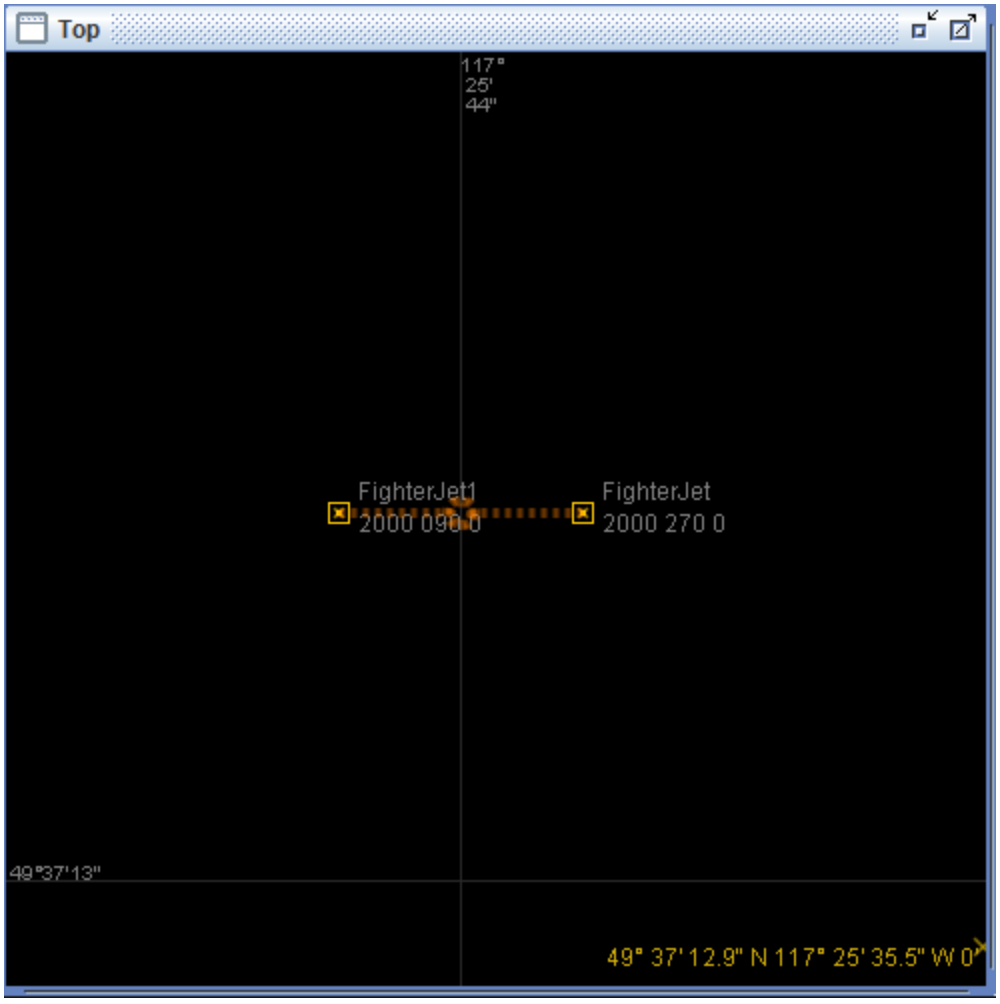
3.

Define and create the two fighters.

```
DEFINE FIGHTER F_1 SPEED MIN 10 MAX 20 DELTA INCREASE 40 DECREASE 35 TURN 15
CLIMB 10 DESCENT 20 EMPTY WEIGHT 55 FUEL INITIAL 89 DELTA 32
DEFINE OLS_RCV RCV_1 DIAMETER 10
DEFINE BOOM MALE BM_1 LENGTH 200 DIAMETER 40 FLOW 40
DEFINE TAILHOOK TH_1 TIME 10
DEFINE AUX_TANK AT_1 AMOUNT 50

CREATE OLS_RCV Receiver FROM RCV_1
CREATE BOOM BoomMale FROM BM_1
CREATE TAILHOOK THook FROM TH_1
CREATE AUX_TANK FighterTank FROM AT_1
CREATE FIGHTER FighterJet FROM F_1 WITH OLS Receiver BOOM BoomMale TAILHOOK
THook TANKS FighterTank AT COORDINATES 49*37’20”/117*27’42”ALTITUDE 2000
HEADING 270 SPEED 0

CREATE OLS_RCV Receiver1 FROM RCV_1
CREATE BOOM BoomMale1 FROM BM_1
CREATE TAILHOOK THook1 FROM TH_1
CREATE AUX_TANK FighterTank1 FROM AT_1
CREATE FIGHTER FighterJet1 FROM F_1 WITH OLS Receiver1 BOOM BoomMale1
TAILHOOK THook1 TANKS FighterTank1 AT COORDINATES
49*37'20"/117*25'46"ALTITUDE 2000 HEADING 090 SPEED 0

POPULATE WORLD WITH FighterJet
POPULATE WORLD WITH  FighterJet1
COMMIT
DO FighterJet BOOM EXTEND
DO FighterJet1 BOOM EXTEND
DO FighterJet1 TRANSFER START
DO FighterJet1 TRANSFER STOP
```

4.
This test is expected to be unsuccessful. Both boom's are male and its illogical that they connect to refuel one another.

5.
The two fighters had to have headings in opposing directions in order for the booms to align. None of the refueling commands crashed however the two booms did not appear to join forming a working connect.



6.
The results of this test differed from the expected. FighterJet1 was not expected to be able to start refueling without some sort of exception being thrown. Also an exception was expected to be throw when the two planes were refueling while not moving in the air and with opposite headings.

7.
This test could be extended by trying to create a female boom on one of the fighters.

F. Fighter-Carrier Tests

Test F.1: Launch Stationary

1.

This tests the functionality of creating a fighter jet one the carrier and its ability to launch a fighter jet from standstill.

2.

Create a carrier as defined in **C.1** and a fighter jet as follows, and then populate the carrier with the fighter jet.

Fighter: speed min **10** max **20** delta inc **40** dec **35** turn **15** climb **10** descent **20** weight **55** fuel **89** delta **32**
OLS: diameter **10**
Boom: length **200** diameter **40** flow **40**
Tailhook: time **10**

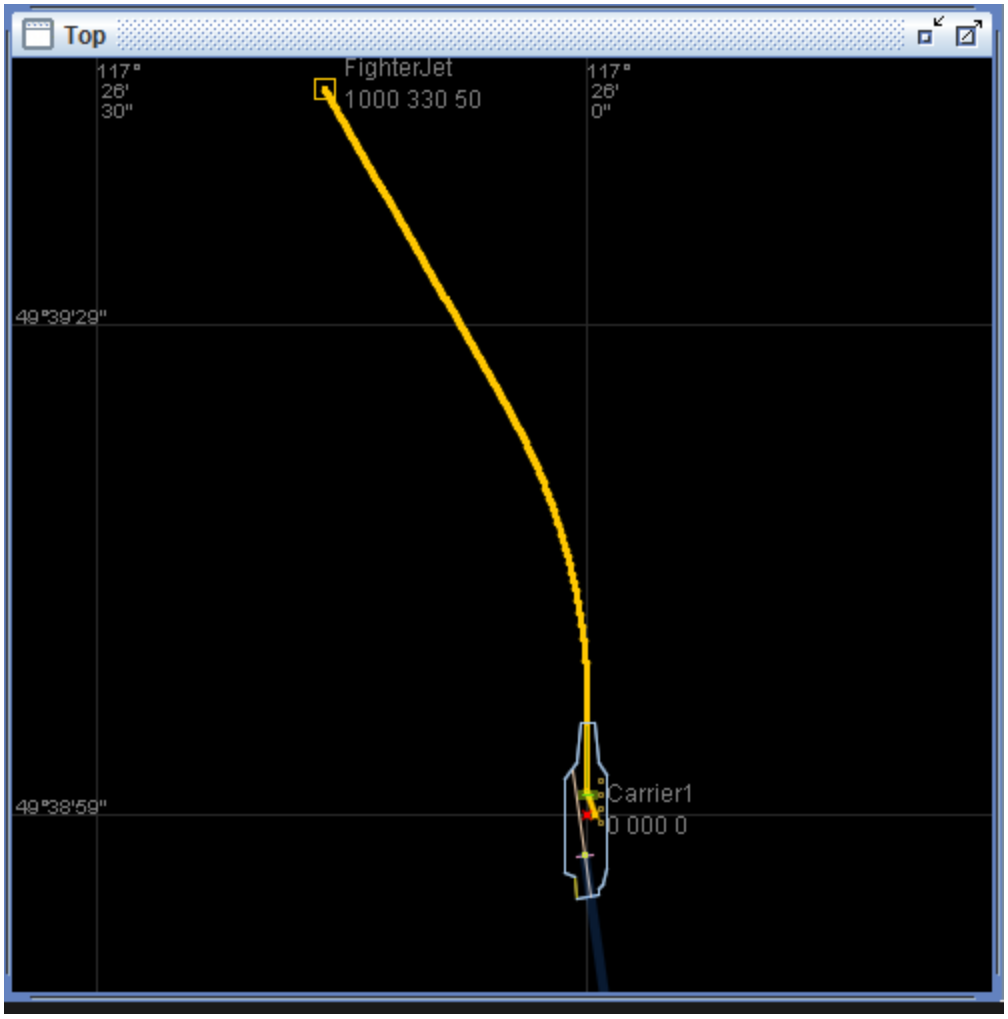
carrier: speed max **10** delta increase **20** decrease **30** turn **40** layout 'carrier.acg'
catapult: origin **+4.4:-141** azimuth **0** length **400** acceleration **20** limit weight **50** speed **45** reset **20**
barrier: origin **+2.4:-119.5** azimuth **0** width **60** time **50**
trap: origin **-14.9:+256.3** azimuth **-8.5** width **400** limit weight **50** speed **35** miss **30**
ols: origin **-14.9:+261.3** azimuth **172** elevation **5** range **10000** diameter **500**
setup: coordinates **49*39'00"/117*26'00"** heading **0** speed **0**

3.

```
DEFINE FIGHTER F_1 SPEED MIN 10 MAX 20 DELTA INCREASE 40 DECREASE 35 TURN 15
CLIMB 10 DESCENT 20 EMPTY WEIGHT 55 FUEL INITIAL 89 DELTA 32
DEFINE OLS_RCV RCV_1 DIAMETER 10
DEFINE BOOM MALE BM_1 LENGTH 200 DIAMETER 40 FLOW 40
DEFINE TAILHOOK TH_1 TIME 10
DEFINE AUX_TANK AT_1 AMOUNT 50
CREATE OLS_RCV Receiver FROM RCV_1
CREATE BOOM BoomMale FROM BM_1
CREATE TAILHOOK THook FROM TH_1
CREATE AUX_TANK FighterTank FROM AT_1
CREATE FIGHTER FighterJet FROM F_1 WITH OLS Receiver BOOM BoomMale TAILHOOK
THook TANKS FighterTank
DEFINE CARRIER C_1 SPEED MAX 10 DELTA INCREASE 20 DECREASE 30 TURN 40 LAYOUT
'carrier.acg'
DEFINE CATAPULT CT_1 ORIGIN +4.4:-141 AZIMUTH 0 LENGTH 400 ACCELERATION 20 LIMIT WEIGHT 50
SPEED 45 RESET 20
DEFINE BARRIER B_1 ORIGIN +2.4:-119.5 AZIMUTH 0 WIDTH 60 TIME 50
DEFINE TRAP T_1 ORIGIN  -14.9:+256.3 AZIMUTH -8.5 WIDTH 400 LIMIT WEIGHT 50
SPEED 35 MISS 30
DEFINE OLS_XMT OLS_1 ORIGIN -14.9:+261.3 AZIMUTH 172 ELEVATION 5 RANGE 10000
DIAMETER 500

CREATE CATAPULT Catapult1 FROM CT_1
CREATE BARRIER Barrier1 FROM B_1
CREATE TRAP Trap1 FROM T_1
CREATE OLS_XMT OLS1 FROM OLS_1
CREATE CARRIER Carrier1 FROM C_1 WITH CATAPULT Catapult1 BARRIER Barrier1
TRAP Trap1 OLS OLS1 AT COORDINATES 49*39'00"/117*26'00" HEADING 000 SPEED 0

POPULATE WORLD WITH Carrier1
POPULATE CARRIER Carrier1 WITH FIGHTER FighterJet
COMMIT

DO FighterJet POSITION
DO Carrier1 BARRIER UP
DO Carrier1 CATAPULT LAUNCH WITH SPEED 5
DO Carrier1 BARRIER DOWN
```

4.
This test is expected to launch the plane normally from a standstill

5.
The test causes the plane to launch successfully from the carrier.



6.
This test results are not any different from what was expected.

7.
This test could be extended by trying to add more than one fighter to the carrier and try to launch another plane while one is in the staging area.

Test F.2: Launch Moving

1.
This tests whether a fighter can be launched off a carrier which is moving.

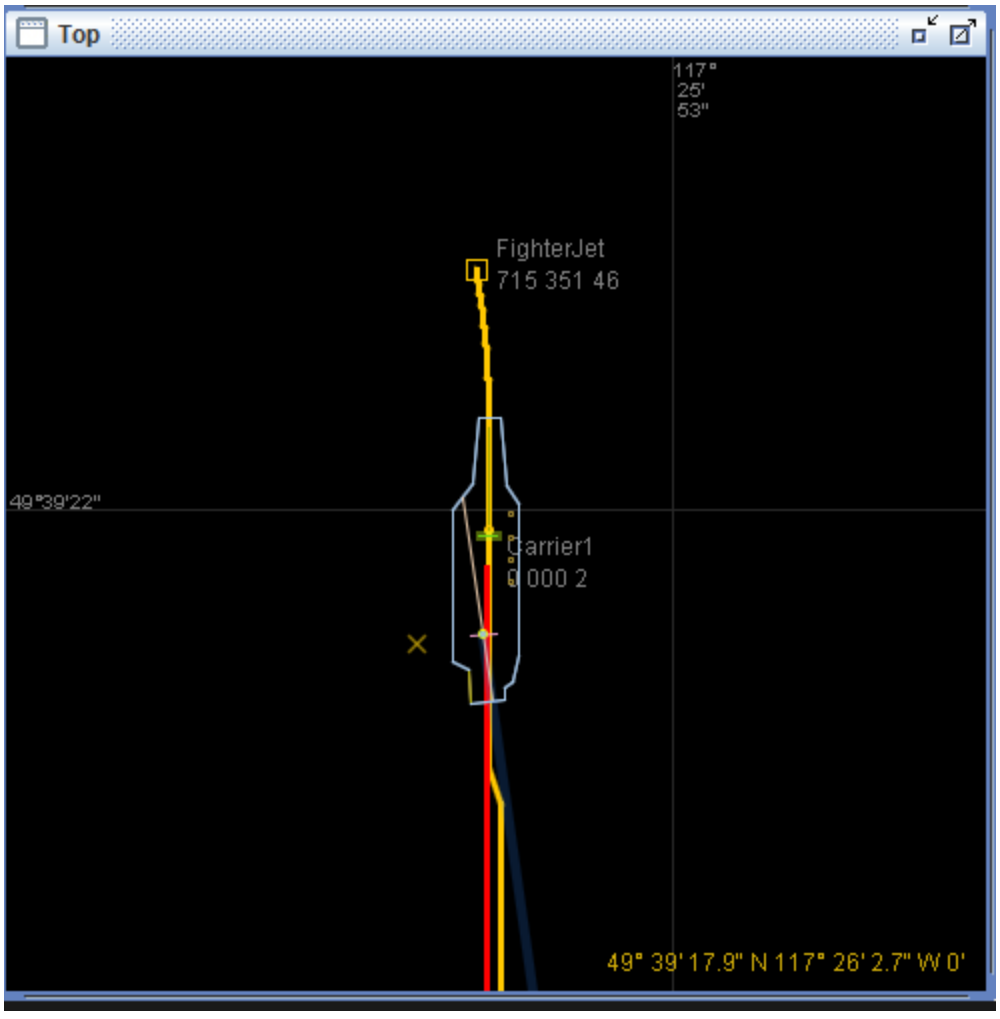
2.
Create a carrier and fighter as defined in **F.1.2** but instead with a speed of 10.

3.
Create a carrier and fighter as illustrated in **F.1.2** the enter the following command to set the carrier speed.

```
DO Carrier1 SET SPEED 10
```

4.
This test is expected to launch the plane normally as in the standstill test **F.1**.

5.
The plane launches from the moving carrier normally.



6.
The actual test results do not differ from the expected results.

7.
This test could be expanded to test what happens if the carrier speed were higher than the catapult launch speed.

Test F.3: Recovery Stationary Trap

1.
This test creates a carrier at speed zero and a fighter in the air and tries to land the fighter on the carrier.

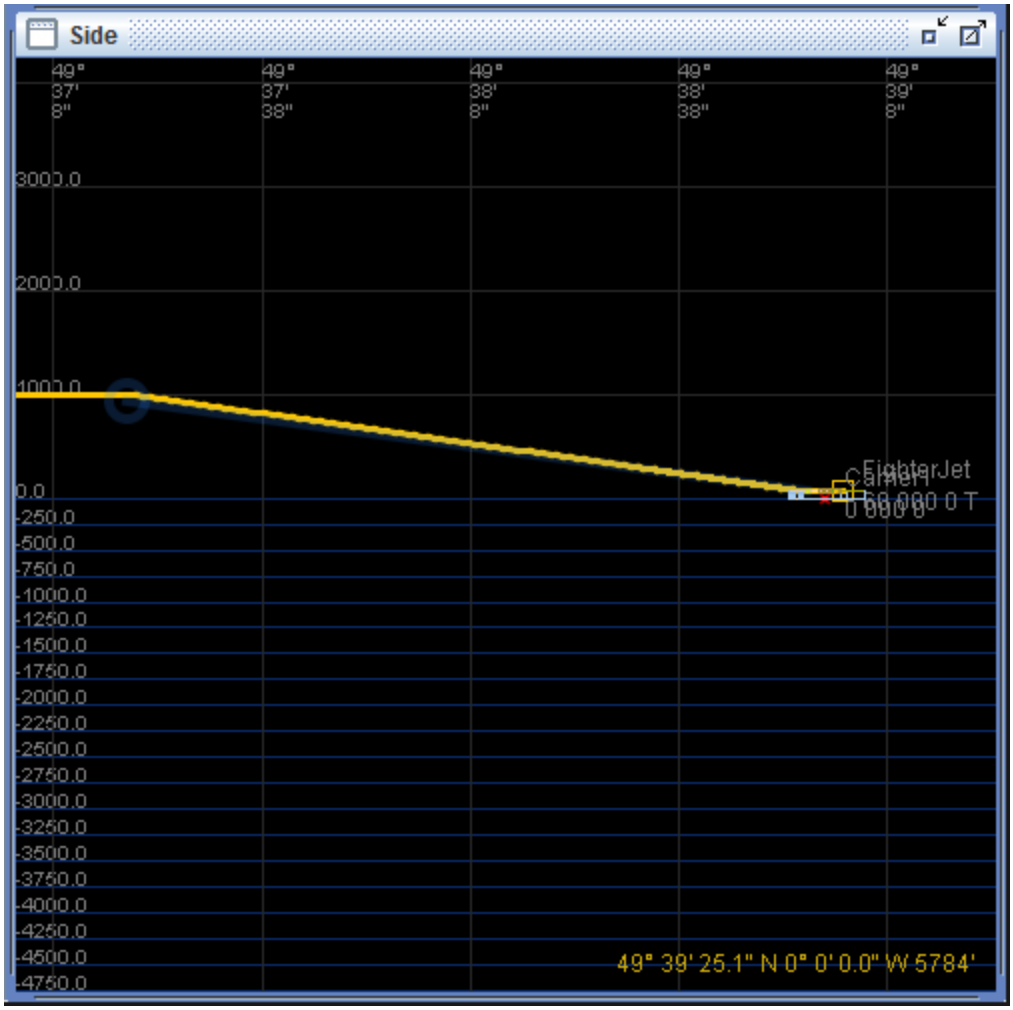
2.
Create a carrier as defined in **F.1.2.** Then place the fighter in front of the carrier heading towards the carrier and have it catch the optical landing systems to land. Place the fighter at coordinates 49*37'20"/117*25'46" with an initial speed of 0.

3.
Create a carrier as in **F.1.3** but do NOT populate the carrier with the fighter jet nor commit the world ...then complete the following commands.

```
POPULATE WORLD WITH FighterJet
COMMIT

@DO FighterJet FORCE COORDINATES 49*37'20"/117*25'46" ALTITUDE 1000 HEADING
000 SPEED 0

DO FighterJet TAILHOOK DOWN
DO FighterJet CAPTURE OLS
```

4.
The fighter jet is expected to perform some sort of latching onto the optical landings system once its pathway intercepts the OLS. Once the fighter jet is given a speed, it should then follow the OLS down until it lands on the plane.

5.
The fighter jet must be placed directly on the OLS circle in order for it to catch it. Once the fighter jet captures the OLS it automatically starts moving and lands on the carrier.



6.
The fighter jet was expected to capture the OLS as long as it was in range of it. It was not expected to have to perfectly intercept the circle at the end of the OLS light. The fighter jet was not expected to start moving once it captured the OLS.

7.
This test could be extended by seeing what happens if the plane is facing the other direction and attempted to capture the OLS. Essentially testing to see if can land backwards.

Test F.4: Recovery Stationary Miss

1.
This test what happens if the fighter misses the trap.

2.
Create a carrier at speed 0 and a fighter in the air. Use the definitions as defined in **F.1.2** but do instead define a trap which misses all the time with the following specifications.

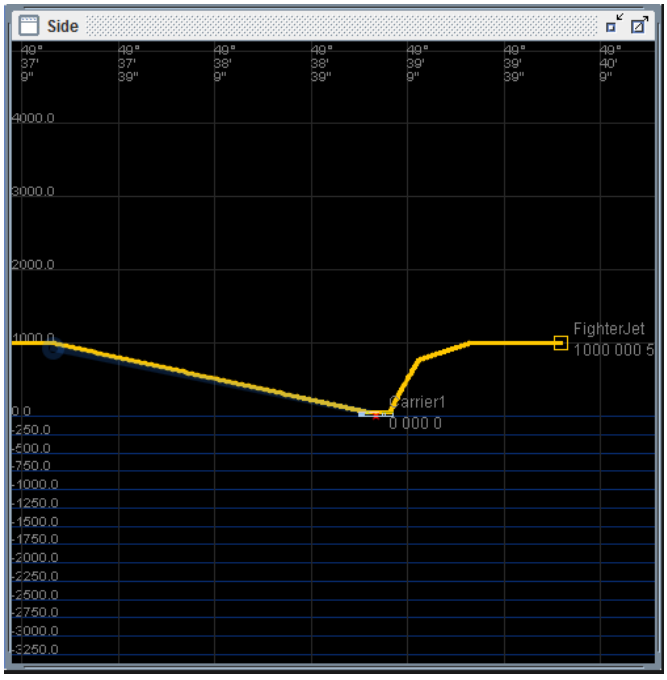
trap: origin **-14.9:+256.3** azimuth **-8.5** width **400** limit weight **50** speed **35** miss **100**

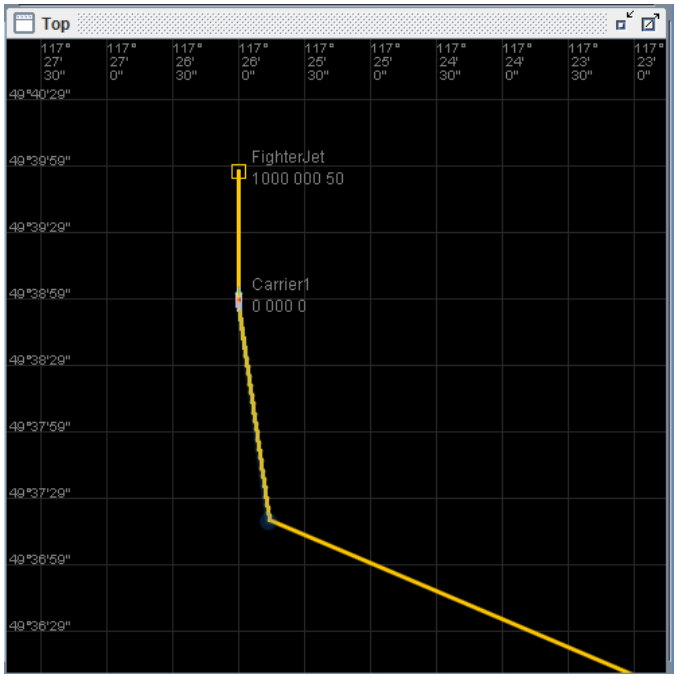
3.
Replace the trap define command in **F.1.2** with the following. Then follow all steps according to **F.3.3**.

```
DEFINE TRAP T_1 ORIGIN -14.9:+256.3 AZIMUTH -8.5 WIDTH 400 LIMIT WEIGHT 50
SPEED 35 MISS 100

//Then enter landing commands from F.3.3
```

4.
This test is expected to try to land the place, once it reaches the carrier and misses the trapit is expected to return to its previous heading, speed, and elevation.

5.
The fighter jet attempted to land as in **F.3**. However once it missed the trap it adopted the ships heading and continued on as if it was being launched from the plane. The fighter climb to its previous altitude but did not return to its old heading.





6.
The actual result differed from the expected in that it did not return to its original heading but instead took appears to have taken on the carriers heading.

7.
This test could be extended by trying to launch and land a plane at the same time, or by trying to land a plane with the barrier in the up position.

Test F.5: Recovery Moving

1.
This tests whether a fighter can land on a carrier which is moving.

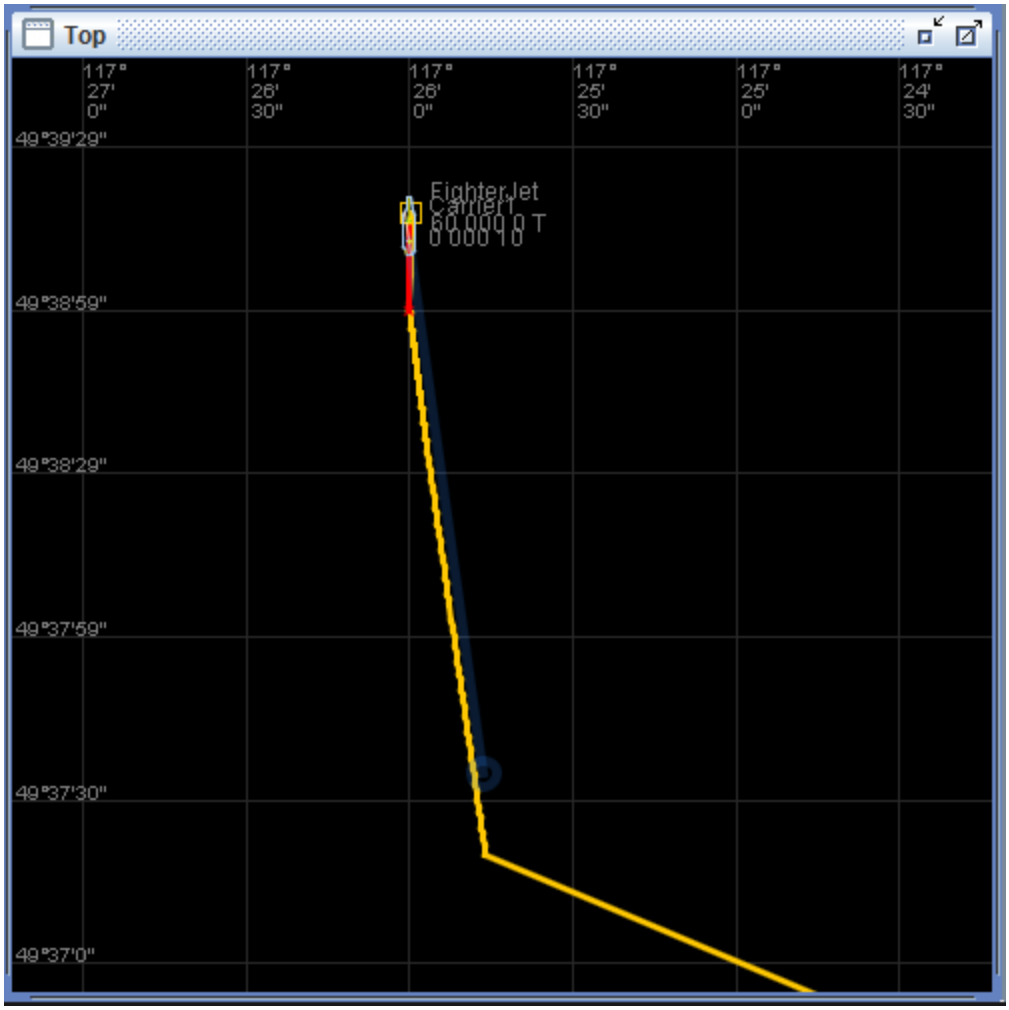
2.
Define a carrier and fighter according to **F.3.2**.

3.
Create a carrier and fighter according the **F.3.3** . But do not attempt to land until the following carrier is set to moving.

```
DO Carrier1 SET SPEED 10

// Now attempt to land the fighter as in F.3.3
```

4.
The fighter is expected to land on the plane as in **F.3.**

5.
The fighter landed on the carrier as in **F.3**



6.
The expected result and the actual result did not differ in the test.

7.
This test could be extend by reversing the flipping the heading or turning the carrier while the fighter is attempting to land on it.

Test F.6: Recovery Catchup

1.
This tests what happens if the carrier is moving at the same speed as the fighter once the OLS is captured.

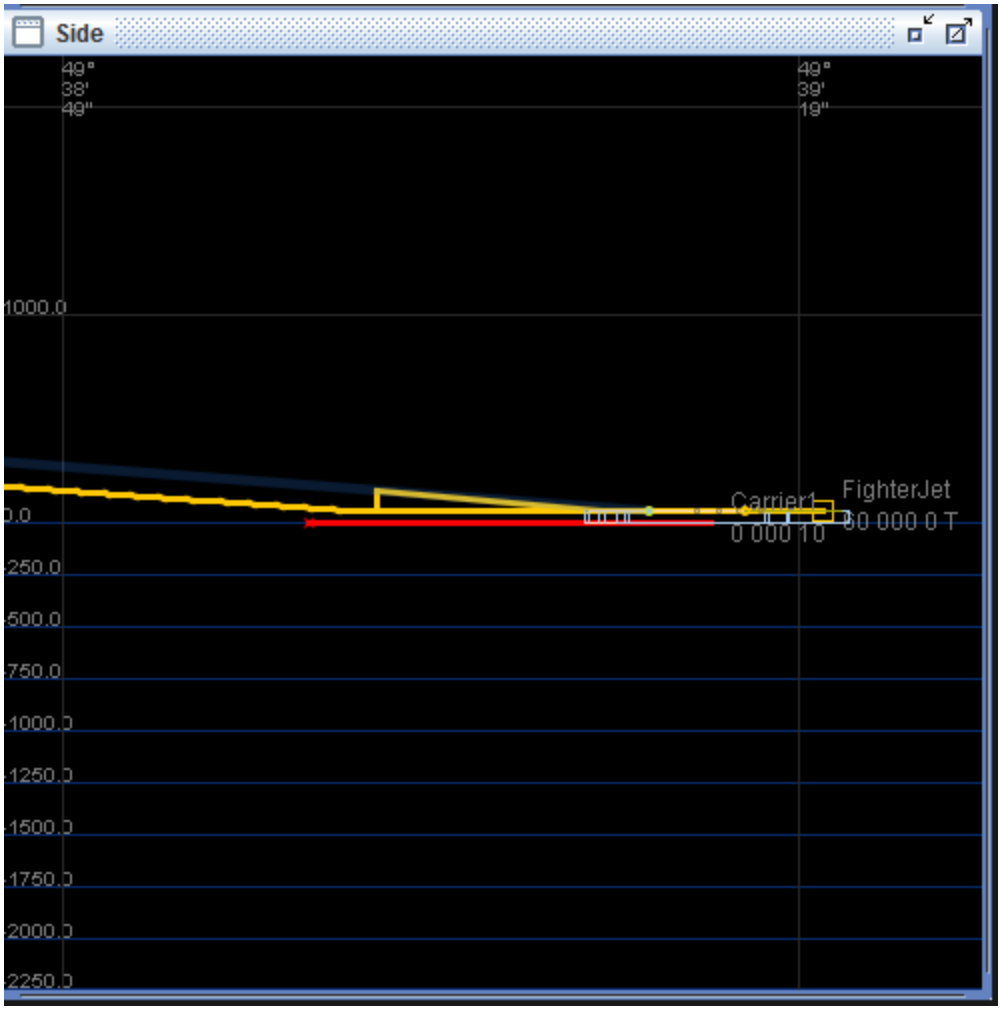
2.
Define a carrier as in **F.3.2** .

3.
Create a carrier as in **F.3.3** but do not attempt to land the fighter with the landing steps in **F.3.3.**

```
@DO FighterJet FORCE COORDINATES 49*37’20”/117*25’46” ALTITUDE 1000 HEADING
000 SPEED 10

DO FighterJet TAILHOOK DOWN
DO FighterJet CAPTURE OLS
DO Carrier1 SET SPEED 10
```

4.
The fighter is expected to be stuck in a game of catchup with the carrier and not be able to land on it.

5.
The fighter was able to land on the carrier, although as soon as it landed there was a strange jump in fighter elevation on the deck and then a smooth settling to the deck again.



6.
Although the fighter was able to land on the fighter, it was not expected to do a jumping maneuver on the flight deck upon touchdown. It appears as if the fighter detected it was too low then jumped to the new OLS position and continued its landing operations following the new OLS configuration.

7.
This test could be extended by trying to accelerate the carrier to a speed faster the the fighter jet is currently traveling.

Test 7: Recovery Moving Turning Early

1.
This tests what happens if the carrier makes a turn left once the fighter has captured the OLS. It then corrects its course.

2.
Define a carrier and fighter as in **F.1.2**.

3.
Create a carrier as defined in **F.1.3** but do not execute any of the landing commands and do not populate the carrier with the fighter.

```
POPULATE WORLD WITH FighterJet
COMMIT

@DO FighterJet FORCE COORDINATES 49*37’20”/117*25’46” ALTITUDE 1000 HEADING
000 SPEED 0
DO Carrier1 SET SPEED 10
DO FighterJet TAILHOOK DOWN
DO Carrier1 SET HEADING 090 LEFT
DO FighterJet CAPTURE OLS
```

4.
The fighter is expected to follow the carrier’s OLS and land on the carrier.

5.
The fighter seemed to be able to land on the carrier. Although it does a strange jump to the side to catch up with the carrier.

6.
The test differed from the expected in that the test was not expected to perform the jog as it did in the above picture.

7.
This test could be extended by attempting to course correct the carrier before the fighter jet has landed.

Test F.8: Recovery Moving Turning Late

1. This test whether the plane can follow the OLS even though the carrier is performing a turn.

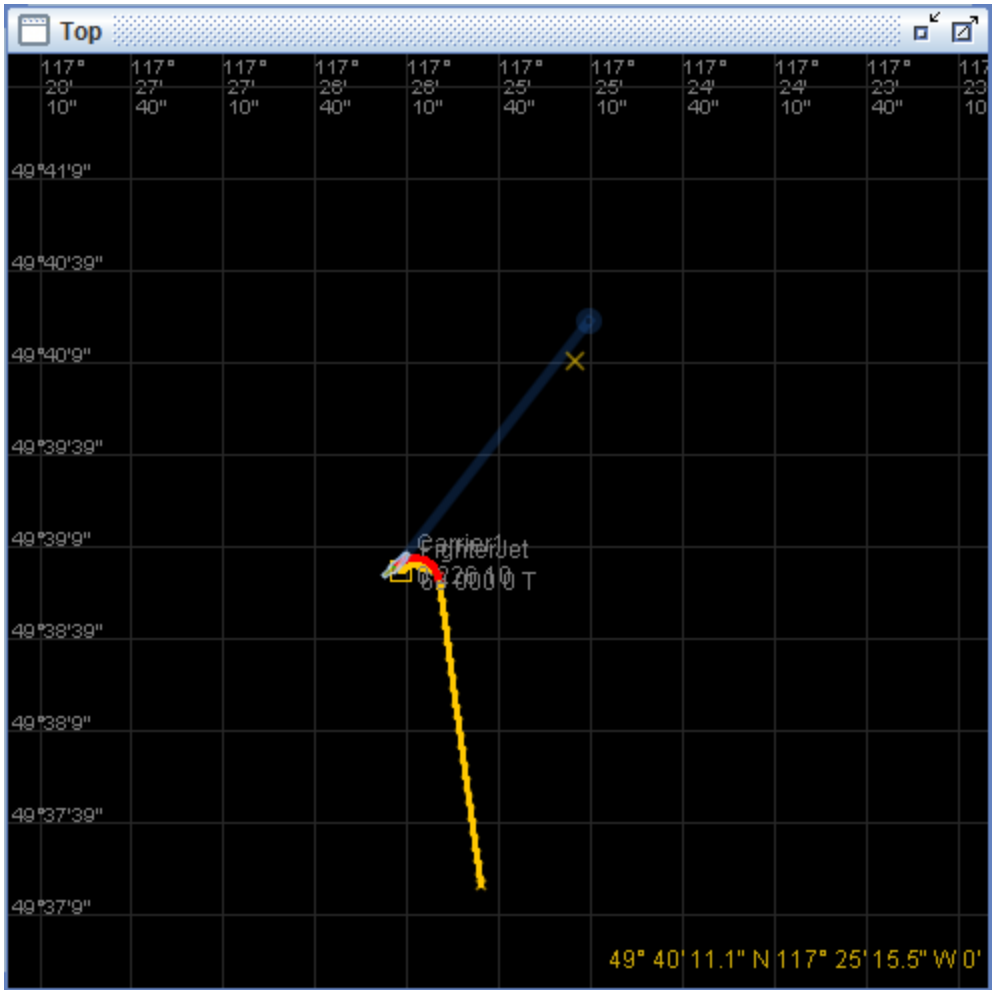
2. Define a carrier and fighter as in **F.1.2**.

3. Create a carrier as in **F.7.3** but instead make the carrier turn after the OLS has been captured.

```
DO FighterJet CAPTURE OLS
DO Carrier1 SET HEADING 090 LEFT
```

4.
This test is expected to cause the plane to continue to follow the OLS light and follow the carrier to a safe landing.

5.
The test resulted in the carrier turning once the OLS was captured, the fighter then was able to still land on the carrier.



6.
The test results did not differ from the expected results.

7.
This test could be extended making the carrier turn again halfway through the fighters landing sequence.

G. Tanker-Tanker Tests

Test G.1: Refueling Attempt

1.
In this test we're trying to refuel a tanker from another tanker.

2.
We define two tankers facing opposite direction on the same elevation and almost same coordinates but with 0 speed.

3.

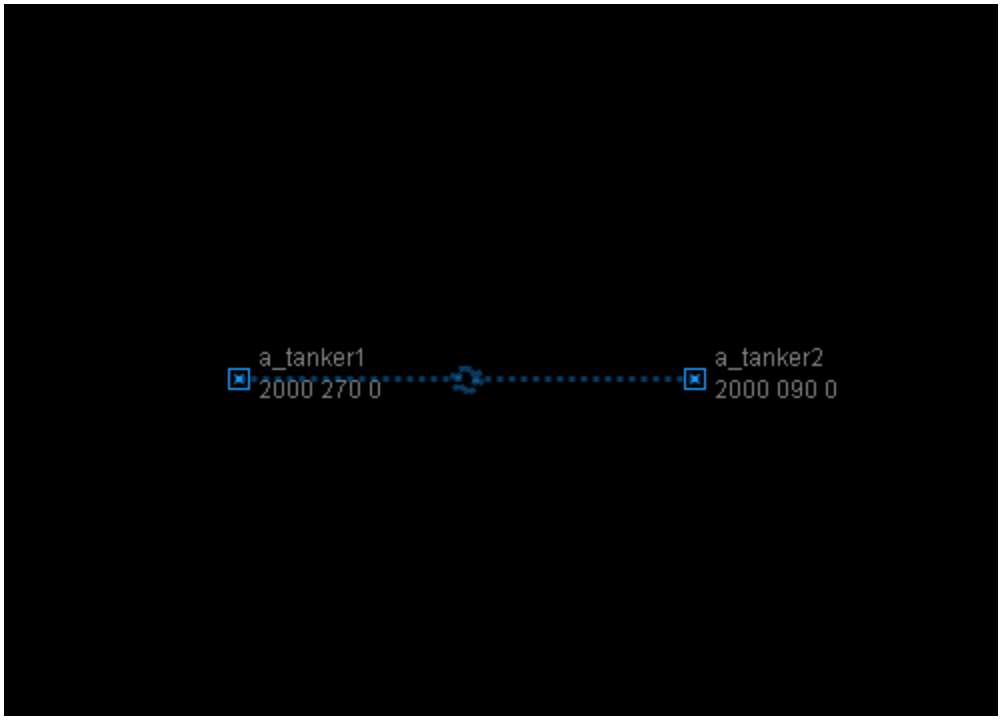
```
DEFINE TANKER t_tanker1 SPEED MIN 10 MAX 20 DELTA INCREASE 30 DECREASE 40
TURN 50 CLIMB 50 DESCENT 60 TANK 70
DEFINE BOOM FEMALE tboomf1 LENGTH 200 DIAMETER 20 ELEVATION 20 FLOW 50
CREATE BOOM a_boomf1 FROM tboomf1
CREATE BOOM a_boomf2 FROM tboomf1
CREATE TANKER a_tanker1 FROM t_tanker1 WITH BOOM a_boomf1 AT COORDINATES
49*39'50"/117*25'40" ALTITUDe 2000 HEADING 270 SPEED 0
CREATE TANKER a_tanker2 FROM t_tanker1 WITH BOOM a_boomf2 AT COORDINATES
49*39'50"/117*25'36" altitude 2000 heading 090 speed 0
POPULATE WORLD WITH a_tanker1 a_tanker2
COMMIT
DO a_tanker1 BOOM EXTEND
DO a_tanker2 BOOM EXTEND
DO a_tanker1 TRANSFER START
DO a_tanker2 START
```

4.

We should expect anything in result since both booms are female thus no coupling will occur.

5.

The booms were touching but no coupling occurred.



6.
Results didn't differ from expected.

7.
We could try to create a male boom on one of the tankers, if that works then redo the whole test.

H. Tanker-Carrier Tests

Test H.1: Landing Attempt

1.
In this test we're trying to land a tanker on a carrier.

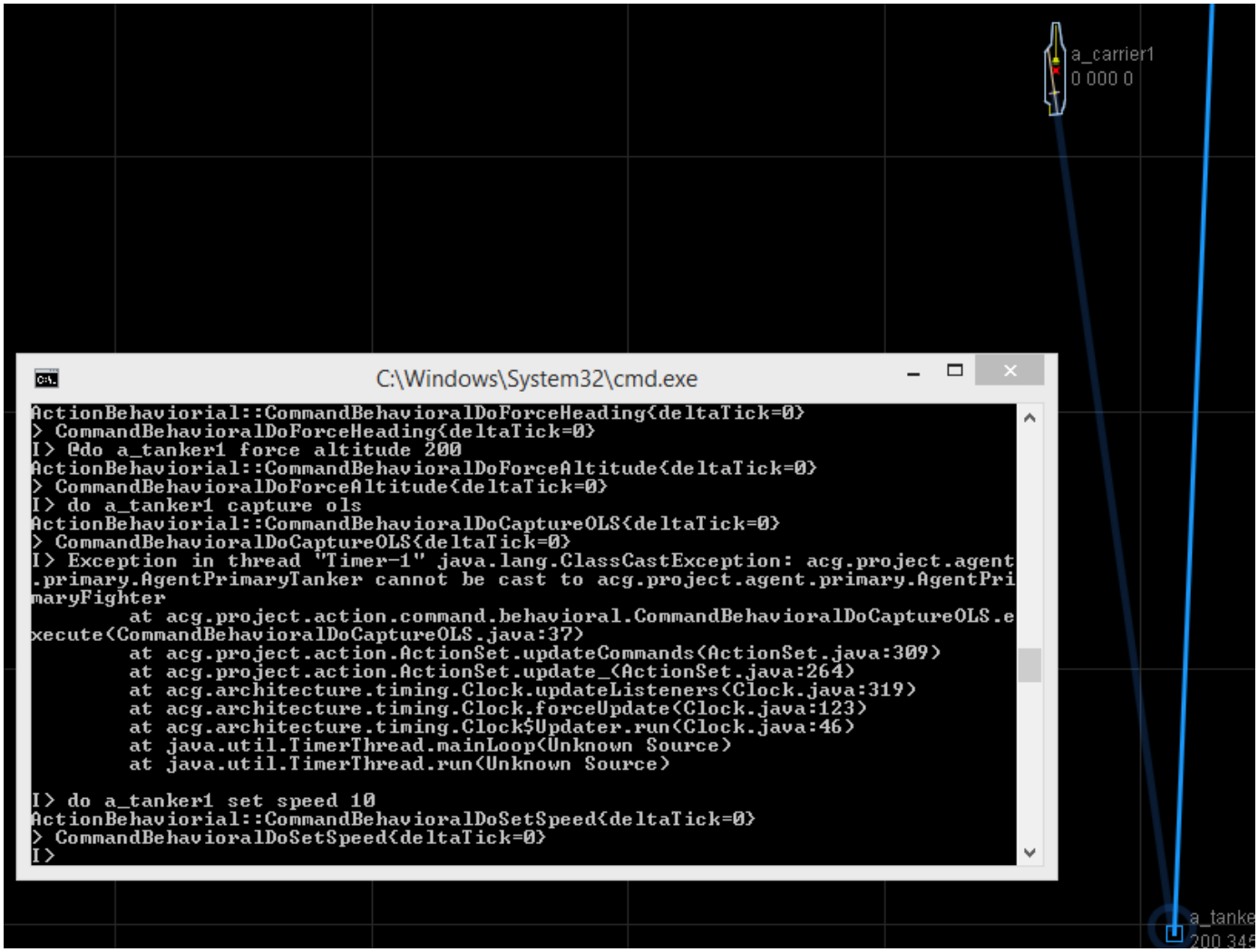
2.
We use the carrier previously defined in C1 and we create a tanker then place it on the ols circle, we then try to make the tanker capture the ols

3.
We use the carrier definition from test **C.1** and we add the following:

```
DEFINE TANKER t_tanker1 SPEED MIN 10 MAX 20 DELTA INCREASE 30 DECREASE 40
TURN 50 CLIMB 50 DESCENT 60 TANK 70; DEFINE BOOM FEMALE tboomf1 LENGTH 200
DIAMETER 20 ELEVATION 20 FLOW 50
CREATE BOOM a_boomf1 FROM tboomf1
CREATE TANKER a_tanker1 FROM t_tanker1 WITH BOOM a_boomf1 AT COORDINATES
49*39'50"/117*25'40" altitude 2000 heading 270 speed 0
POPULATE WORLD WITH a_tanker1 a_carrier1
COMMIT
@DO a_tanker1 FORCE COORDINATES 49*37'19"/117*25'46"
@DO a_tanker1 FORCE HEADING 345 right
@DO a_tanker1 FORCE HEADING 345
@DO a_tanker1 force ALTITUDE 200
DO a_tanker1 CAPTURE ols
```

4.
The test should fail because the tanker doesn't have an ols receiver or a tailhook.

5.
We received an exception error when we tried to capture the ols with the tanker.



6.
The results didn't differ from expected.

7.
We can try to implement a tailhook and an OLS receiver on the tanker, if that works we try this landing test again.

I. Carrier-Carrier Tests

Test I.1: Collision

1.
In this test we try to collide two carriers head on at slow speed.

2.
We create two carriers facing each other and we place them on the same trajectory the we move them.

3.

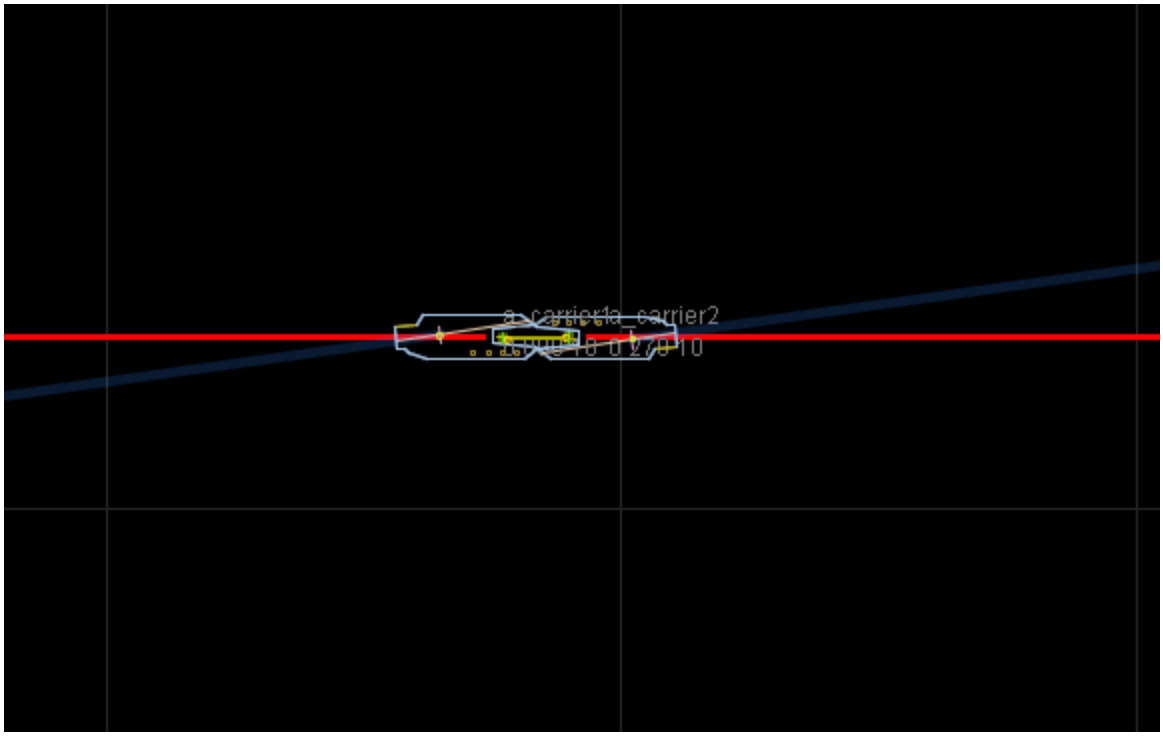
```
DEFINE CARRIER t_carrier1 SPEED MAX 10 DELTA INCREASE 20 DECREASE 30 TURN 40
LAYOUT 'carrier.acg'
DEFINE CATAPULT t_cat1 ORIGIN +4.4:-141 AZIMUTH 0 LENGTH 400 ACCELERATION 20
LIMIT WEIGHT 50 SPEED 45 RESET 20
DEFINE BARRIER t_bar1 ORIGIN +2.4:-119.5 AZIMUTH 0 WIDTH 60 TIME 50
DEFINE TRAP t_trap1 ORIGIN -14.9:+256.3 AZIMUTH -8.5 WIDTH 400 LIMIT WEIGHT
50 SPEED 35 MISS 30
DEFINE OLS_XMT t_ols1 ORIGIN -14.9:+261.3 AZIMUTH 172 ELEVATION 5 RANGE
10000 DIAMETER 500
CREATE CATAPULT a_cat1 FROM t_cat1
CREATE BARRIER a_bar1 FROM t_bar1
CREATE TRAP a_trap1 FROM t_trap1
CREATE OLS_XMT a_ols1 FROM t_ols1
CREATE CATAPULT a_cat2 FROM t_cat1
CREATE BARRIER a_bar2 FROM t_bar1
CREATE TRAP a_trap2 FROM t_trap1
CREATE OLS_XMT a_ols2 FROMt_ols1
CREATE CARRIER a_carrier1 FROM t_carrier1 WITH CATAPULT a_cat1 BARRIER
a_bar1 TRAP a_trap1 OLS a_ols1 AT COORDINATES 49*39'00"/117*26'50" HEADING
090 SPEED 10
CREATE CARRIER a_carrier2 FROM t_carrier1 WITH CATAPULT a_cat2 BARRIER
a_bar2 TRAP a_trap2 OLS a_ols2 AT COORDINATES 49*39'00"/117*25'00" HEADING
270 SPEED 10
```

4.

Our simulator does not check for overlapping or collision therefore there should be no collision.

5.

The carrier just overlapped each other completely ignoring collision



6.
The results didn't differ from expected.

7.
We can expand this test by trying to have two carriers on top of each other facing the same direction therefore their ols systems are overlapping as well, then we try to have a fighter capture the ols and land.

J. Operational Tests

Test J.1: Once Around the Pattern

1.
This test creates a fighter aboard an aircraft carrier, has it launch, get refueled by a tanker, land back on the carrier, and then launch again.

2
Define a fighter on a carrier as in **F.1.2** and a tanker as in **B.1.2.**

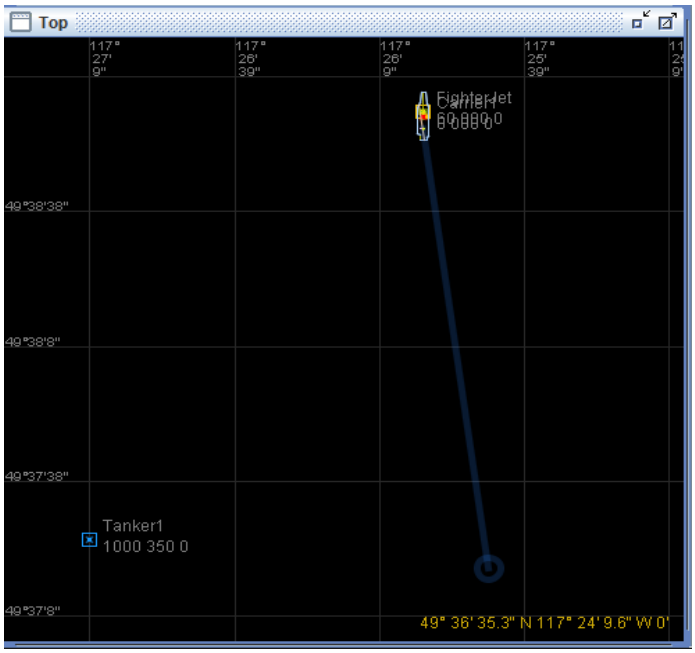
3.

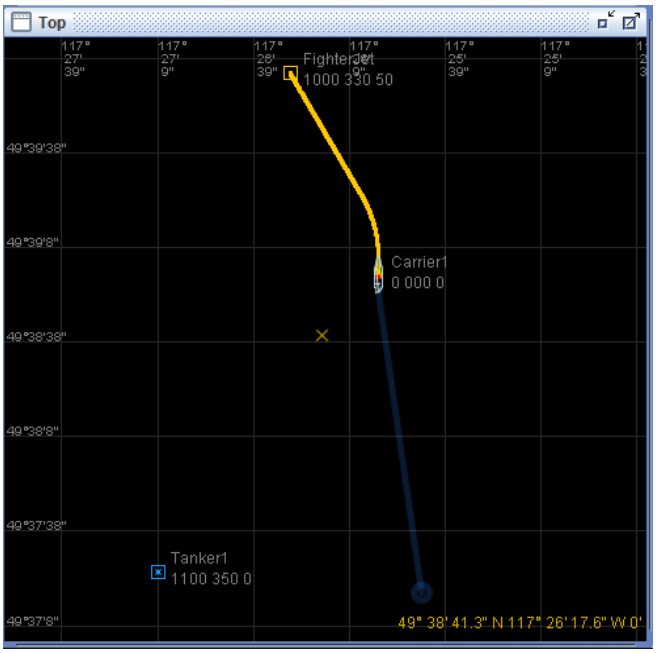
Create a fighter on a carrier as in **F.1.3** and a tanker**.** Launch the fighter from the carrier as in **F.1.3**. Then force it into range of the tanker and refuel it. Then force it into range of the OLS and land the fighter, after that, launch the fighter again.

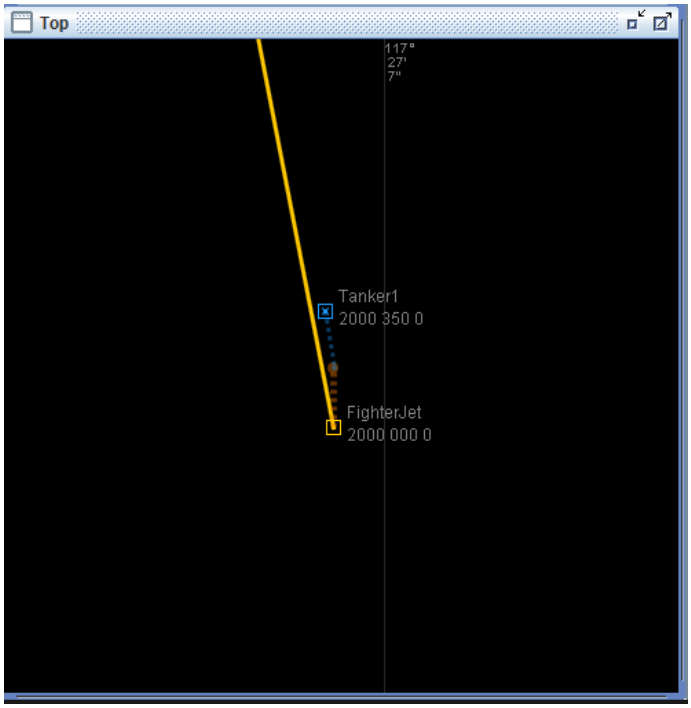
```
// Create a tanker
DEFINE TANKER TA_1 SPEED MIN 10 MAX 20 DELTA INCREASE 30 DECREASE 40 TURN 50
CLIMB 50 DESCENT 60 TANK 70
DEFINE BOOM FEMALE tboomf1 LENGTH 200 DIAMETER 20 ELEVATION 20 FLOW 50

CREATE BOOM FemaleBoom FROM tboomf1
CREATE TANKER Tanker1 FROM TA_1 WITH BOOM FemaleBoom AT COORDINATES
49*37’26”/117*27’9” ALTITUDE 2000 HEADING 350 SPEED 0

POPULATE WORLD WITH Tanker


// Create a fighter and carrier as in F.1.3 and launch the fighter
plane...etc.

@DO FighterJet FORCE COORDINATES 49*37'22.1"/117*27'8.7" ALTITUDE 2000
HEADING 000 SPEED 0

DO Tanker1 BOOM EXTEND
DO FighterJet BOOM EXTEND
DO Tanker1 TRANSFER START
DO Tanker1 TRANSFER STOP

@DO FighterJet FORCE COORDINATES 49*37'20"/117*25'46" ALTITUDE 1000 HEADING
000 SPEED 0

DO Tanker1 BOOM RETRACT
DO FighterJet BOOM RETRACT

DO FighterJet TAILHOOK DOWN
DO FighterJet CAPTURE OLS

DO FighterJet POSITION
DO Carrier1 BARRIER UP
DO Carrier1 CATAPULT LAUNCH WITH SPEED 5
```

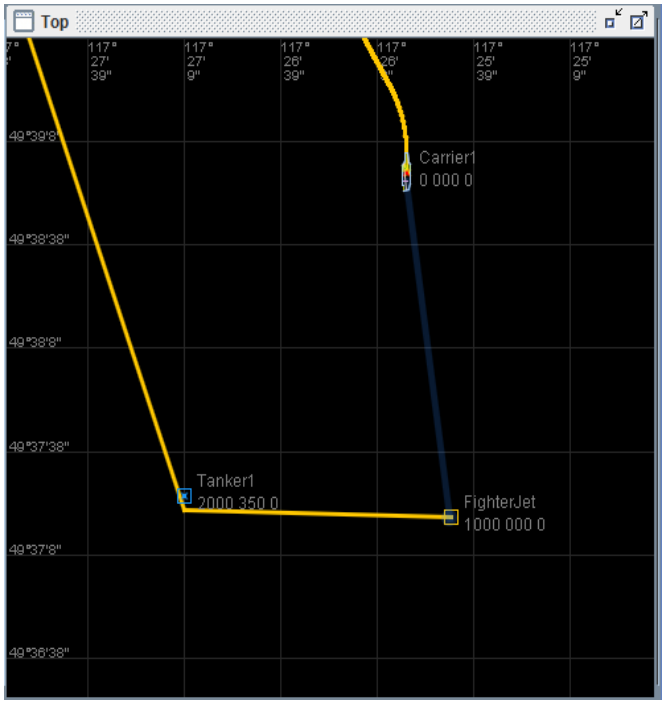
4.
All creations are expected to operate as normal. The fighter jet is expected to launch from the carrier, be refueled by the tanker, land on the carrier, and then be launched again.

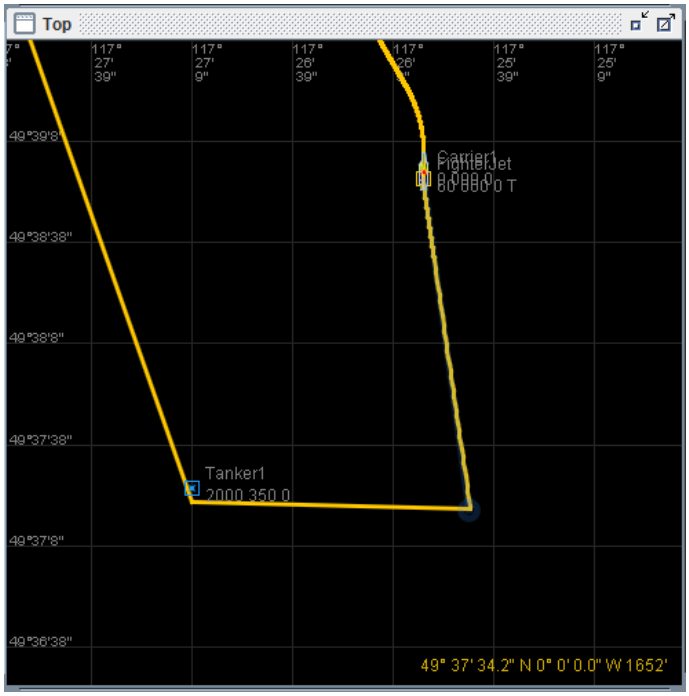
5.
The plane was created on the carrier and the tanker in the air. Once the fighter was launched, it was able to go to the correct coordinates and then both the tanker and fighter extended there booms, connected, and refueled. The fighter then broke away and went into position to catch the OLS from the carrier. Once the fighter capture the OLS it proceeded to land successfully on the carrier. Once landed the fighter moved into launch position. The barrier was raised and then the fighter launched from the carrier again.

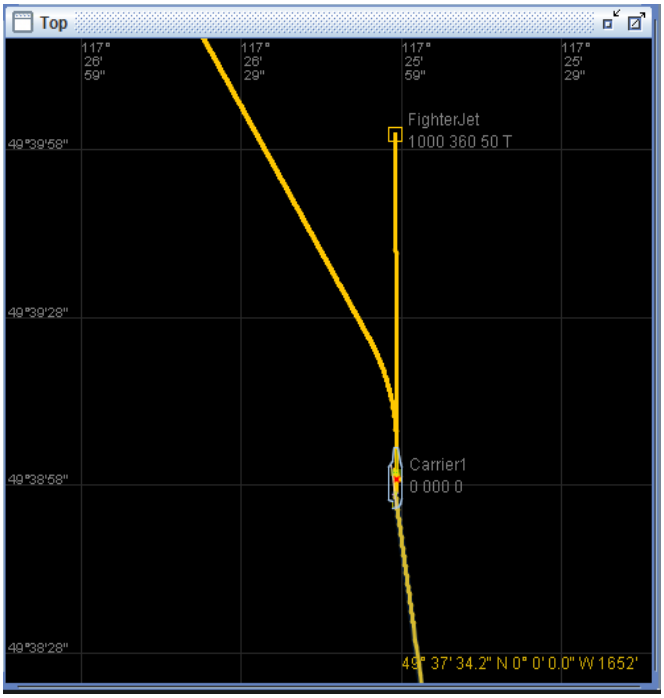












6.

The actual results did not different from the expected results. In the process, the the test revealed that a fighter can be landed on a carrier and moved to launch position even with the carrier's barrier in the up position.

7.
This test could be extended by attempting to land with the tailhook up, and the boom extended instead. It could also be extended by adding more airplanes into the circuit.

Test 2: Kamikaze

1.
This test adds two carriers (one named Godzilla), two fighters (one initially on board the other carrier, the other airborne), and two tankers. It then crashes all the airplanes into Godzilla at the same time.

2.
Define a 2 carriers and a fighter as in **C.1.2**, 2 tankers as in **B.1.2**. Name one carrier Godzilla and populate it with one of the fighter jets.

Initial coordinates Godzilla: **49*39'59"/117*25'00"** Altitude: **2000** Heading: **000** Speed: **0**

Initial coordinates FighterJet: **-** on Carrier1
Initial coordinates Carrier1: **49*39'30"/117*25'00"** Heading: **180** Speed: **5**
Initial coordinates Tanker1: **49*40'30"/117*25'00"** Altitude: **1** Heading: **180** Speed: **5**
Initial coordinates Tanker2: **49*39'59"/117*24'30"** Altitude: **1** Heading: **270** Speed: **5**
Initial coordinates FighterJet2 **49*39'59"/117*25'30"** Altitude: **1** Heading: **090** Speed: **5**

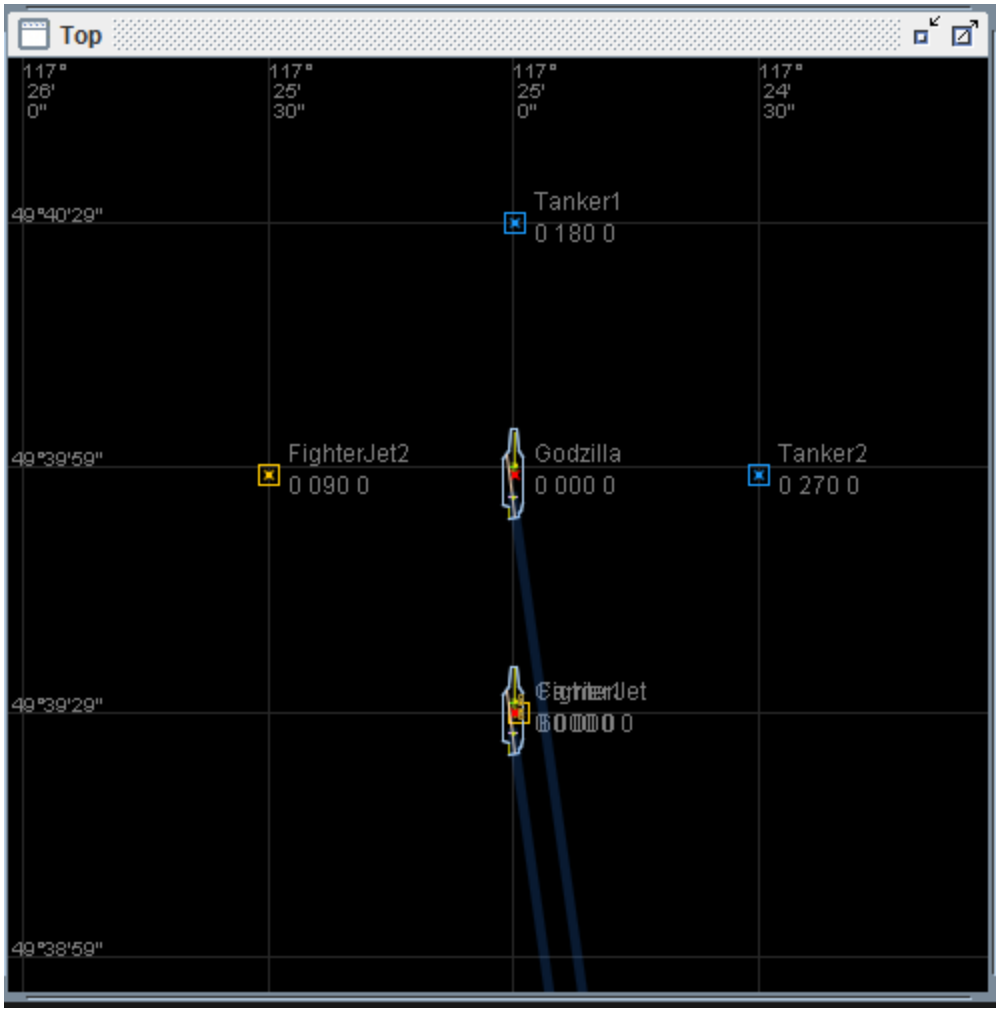
3.

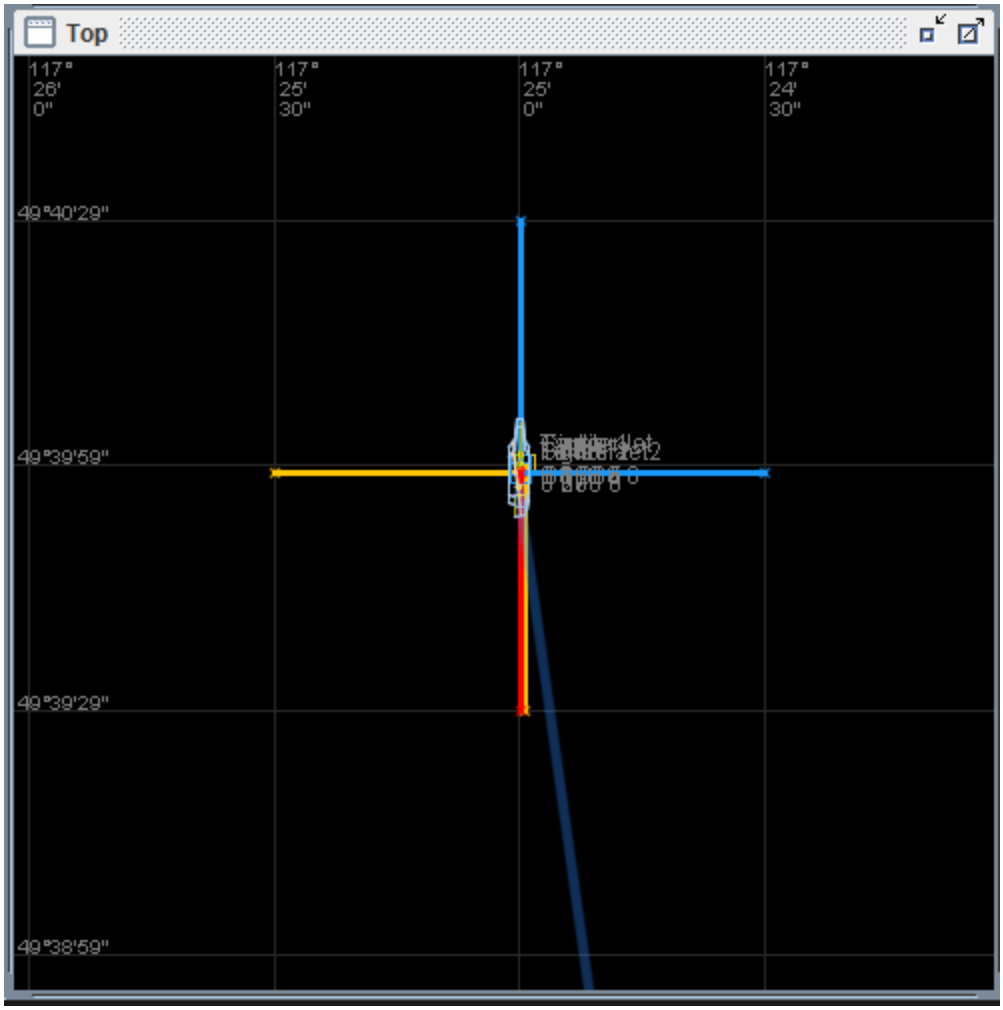
In all referenced commands do not COMMIT the world!

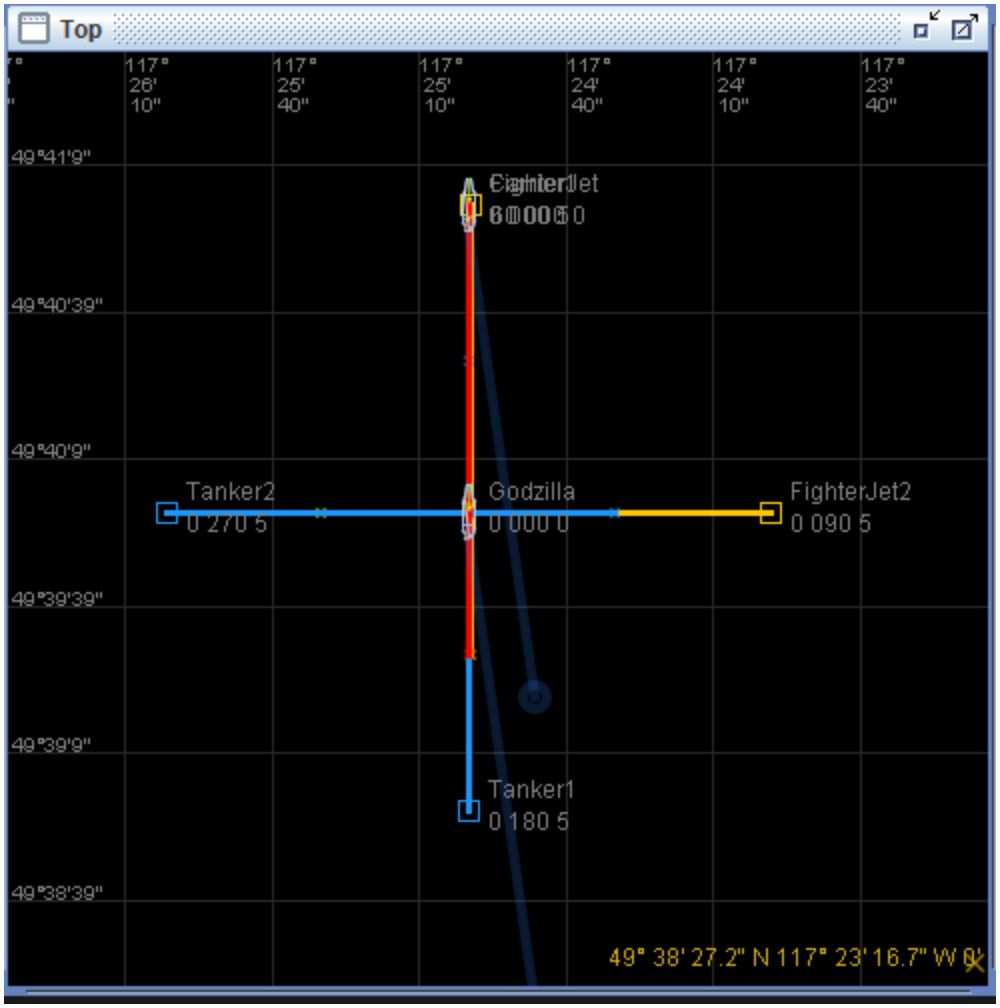
```
// Create Carrier1 populated with fighterjet from  C.1.2 from then force it
to be on the correct coordinates.
// Now create Godzilla
CREATE CATAPULT Catapult2 FROM CT_1
CREATE BARRIER Barrier2 FROM B_1
CREATE TRAP Trap2 FROM T_1
CREATE OLS_XMT OLS2 FROM OLS_1
CREATE CARRIER Godzilla FROM C_1 WITH CATAPULT Catapult2 BARRIER Barrier2
TRAP Trap2 OLS OLS2 AT COORDINATES 49*39'59"/117*25'00" HEADING 000 SPEED 0
Populate WORLD WITH Godzilla
// Now create FighterJet2
CREATE OLS_RCV Receiver2 FROM RCV_1
CREATE BOOM BoomMale2 FROM BM_1
CREATE TAILHOOK THook2 FROM TH_1
CREATE AUX_TANK FighterTank2 FROM AT_1
CREATE FIGHTER FighterJet2 FROM F_1 WITH OLS Receiver2 BOOM BoomMale2
TAILHOOK THook2 TANKS FighterTank2 AT COORDINATES 49*39'59"/117*25'30"  ALTITUDE
1 HEADING 090 SPEED 0
POPULATE WORLD WITH FighterJet2
// Now Create  tanker1
DEFINE TANKER TA_1 SPEED MIN 10 MAX 20 DELTA INCREASE 30 DECREASE 40 TURN 50
CLIMB 50 DESCENT 60 TANK 70
DEFINE BOOM FEMALE tboomf1 LENGTH 200 DIAMETER 20 ELEVATION 20 FLOW 50
CREATE BOOM FemaleBoom FROM tboomf1
CREATE TANKER Tanker1 FROM TA_1 WITH BOOM FemaleBoom AT COORDINATES
49*40'30"/117*25'00"  ALTITUDE 1 HEADING 180 SPEED 0
POPULATE WORLD WITH Tanker
// Create tanker2 as follows
CREATE BOOM FemaleBoom2 FROM tboomf1
CREATE TANKER Tanker2 FROM TA_1 WITH BOOM FemaleBoom2 AT COORDINATES
49*39'59"/117*24'30" ALTITUDE 1 HEADING 270 SPEED 0
POPULATE WORLD WITH Tanker2
// Commit the world
COMMIT
// Arrange agents in correct space
@DO Carrier1 FORCE COORDINATES 49*39'30"/117*25'00" HEADING 180 SPEED 0
// Now put the agents in motion
@DO Carrier1 FORCE SPEED 5
@DO FighterJet2 FORCE SPEED 5
@DO Tanker1 FORCE SPEED 5
@DO Tanker2 FORCE SPEED 5
```

4.
It is expected that all of the agents will pass through each other and continue on their way.

5.
All the agents are created surrounding Godzilla. Once all agents are given a speed, they travel right on through Godzilla as if it didn't exist and then continue on their way.




Top
117°
26'
0"
117°
25'
30"
117°
25'
0"
117°
24'
30"
49°40'29"
49°39'59"
49°39'29"
49°38'59"



6.
The actual results did not differ from the expected results.

7.
This test could be extended by starting all of the agents on top of each other to check once again for collision success.