The VSS (Virtual Ship Simulator) Specification

Jinseok Seo
jsseo@postech.ac.kr

November 15, 2004

Chapter 1

Specification Level 1

1.1 New Features

- Basic objects and functions/behavior
- Keyboard based interaction

1.2 Requirements List

- The Virtual Ship Simulator (called ***ShipSimulator***) shall help users (called ***User***) operate a vessel (called ***MyShip***) and practice docking.
- ***ShipSimulator*** shall control several number of vessels (called ***OtherShip***) navigating automatically.
- ***ShipSimulator*** shall display the interior view of the bridge including a steering wheel (called ***SteeringWheel***) and an engine lever (called ***EngineTelegraph***).
- ***ShipSimulator*** shall display static environments containing the sky (called ***Sky***), the land (called ***Land***), and the sea surface (called ***Sea***).
- ***ShipSimulator*** shall accept input from the keyboard to control ***MyShip***.
- ***OtherShip*** shall its initial position and direction randomly.
- ***OtherShip*** shall change its speed and course every 10 seconds.
- ***User*** shall look around the interior view of the bridge.
- ***User*** shall change the cource of ***MyShip***.
- ***User*** shall change the velocity of ***MyShip***.

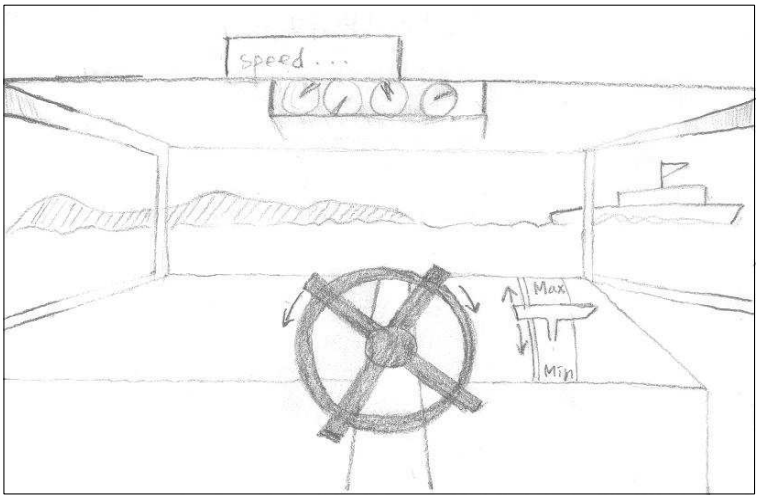

Figure 1.1: The initial sceen.

1.3 Storyboards

- Scene 1.1: When ***ShipSimulator*** starts, ***User*** can see the scene like Figure 1.1. In this scene, ***User*** can look around the interior view of the bridge.
- Scene 1.2: The over view of Scene 1.1.

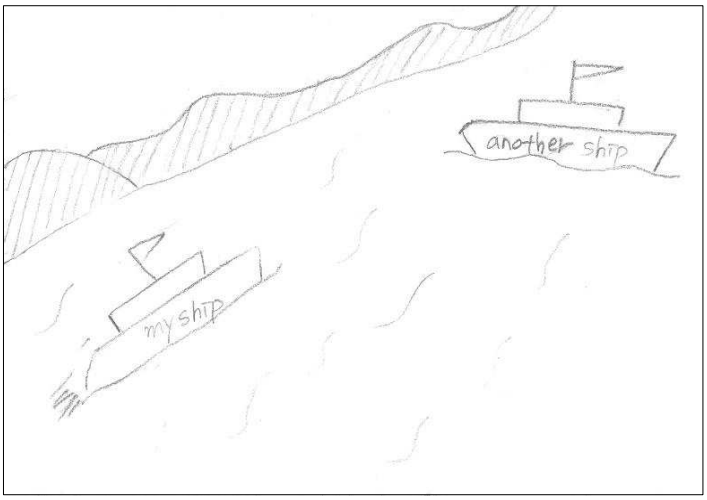

Figure 1.2: The over view of Scene 1.1.

- Scene 1.3: ***User*** can change ***MyShip***'s course by manipulating /4 in, dy = 0-1/8 in***SteeringWheel***.

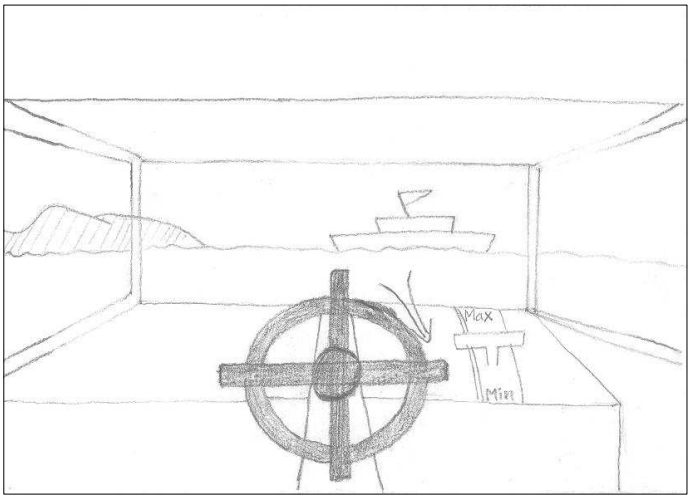

Figure 1.3: ***MyShip*** is changing its course.

- Scene 1.4: The over view of Scene 1.3.

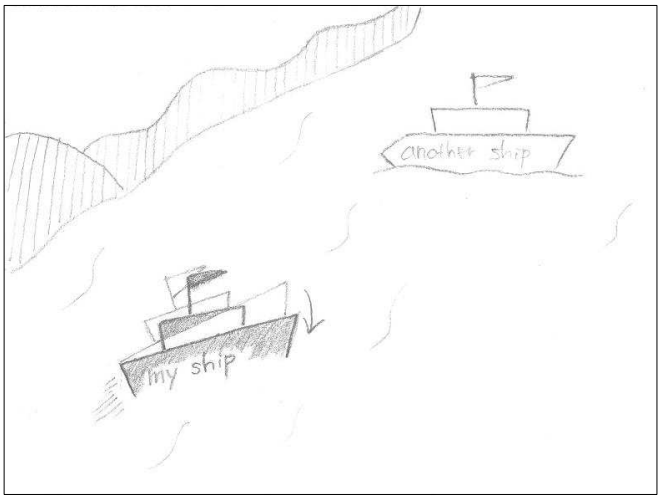

Figure 1.4: The over view of Scene 1.3.

- Scene 1.5: ***User*** can change ***MyShip***'s speed by manipulating ***EngineTelegraph***.

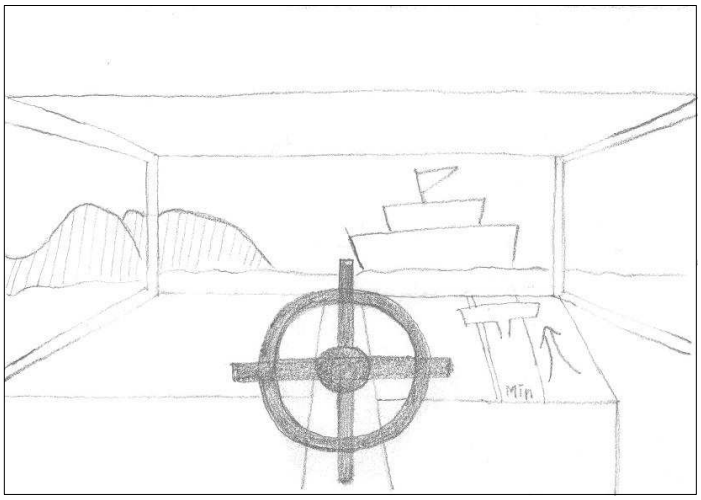

Figure 1.5: ***MyShip*** is changing its speed.

- Scene 1.6: The over view of Scene 1.5

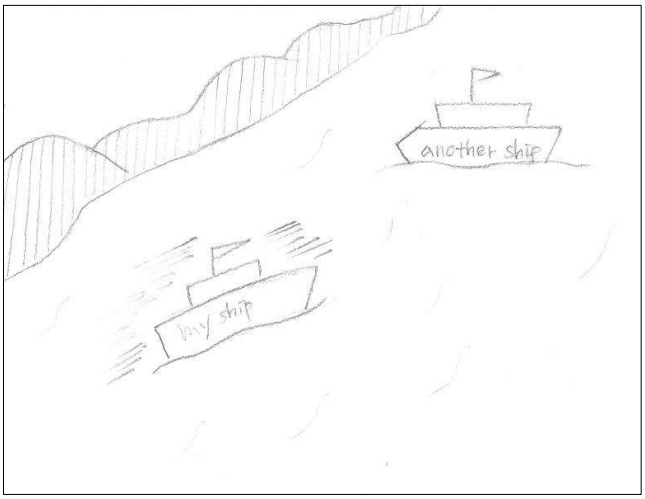

Figure 1.6: The over view of Scene 1.5

1.4 Message sequence diagrams

- MSD 1.1 : **To look around.** ***User*** can look left and right by pressing the "z" and "c" keys, respectively. Pressing the "x" key returns ***Camera*** to the initial direction.

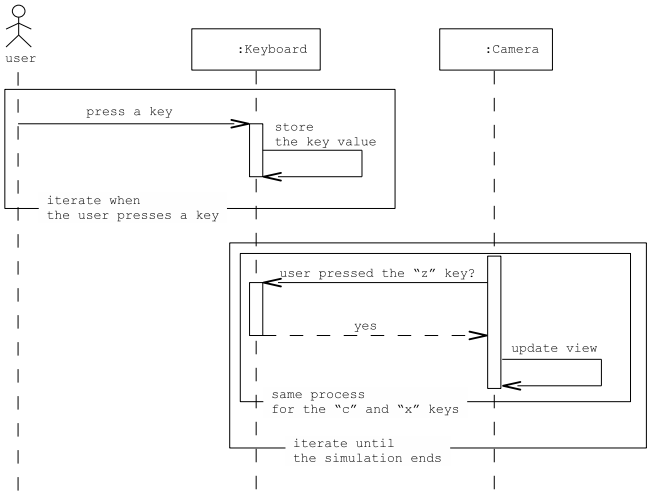

Figure 1.7: To look around.

- MSD 1.2 : **To change speed.** ***User*** can change the speed of ***MyShip*** by pressing the "up arrow" (to increase speed) and "down arrow" (to decrease speed) keys.

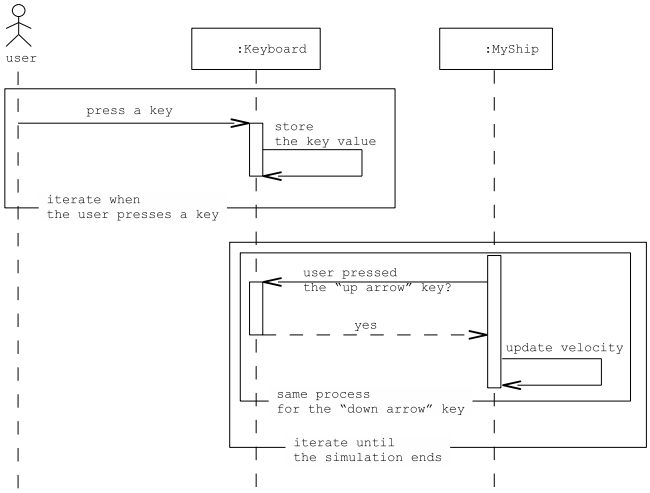

Figure 1.8: To change speed.

- MSD 1.3 : **To change course.** ***User*** can change the course of ***MyShip*** by pressing the "left arrow" (to turn left) and "right arrow" (to turn right) keys.

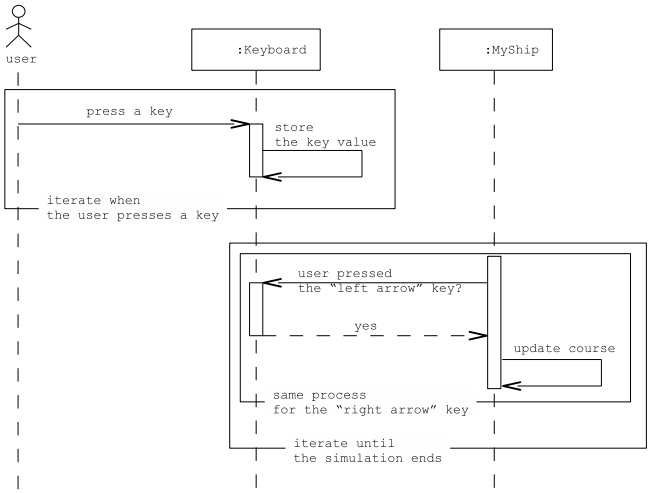

Figure 1.9: To change course.

- MSD 1.4 : **To navigate automatically.** ***OtherShip*** determines its initial position and direction randomly and changes its speed and course every 10 seconds.

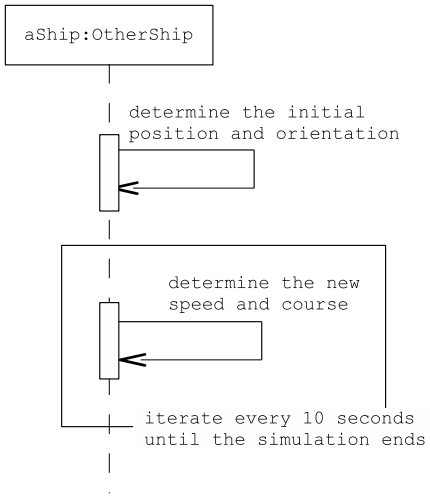

Figure 1.10: To navigate automatically.

1.5 Scenegraph

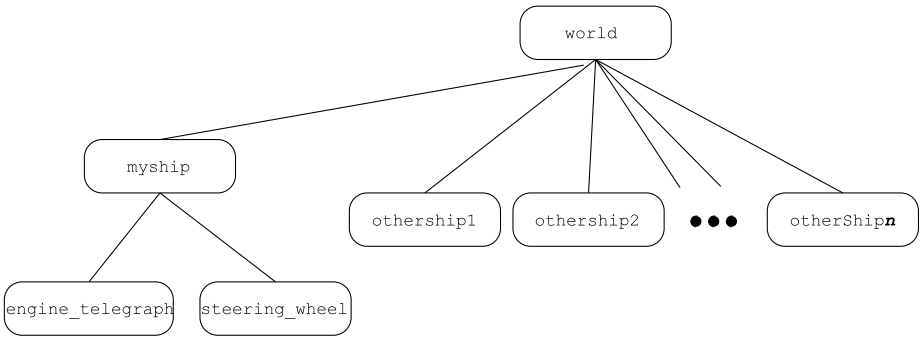

Figure 1.11: Scenegraph level 1.

1.6 Statecharts

1.6.1 Statecharts 1.1: *MyShip* level 1

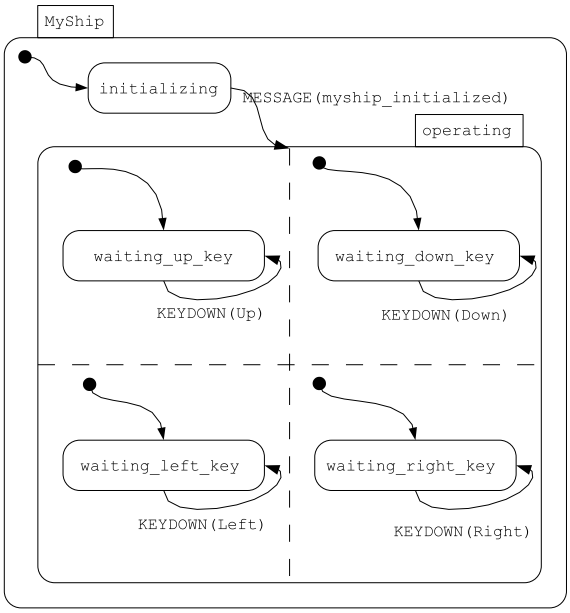

Figure 1.12: Statecharts of ***MyShip*** level 1.

- State specification 1.1: ***MyShip.initializing***
  - OnEnter

    ```
    self.velocity = 0
    ```

```
self.target_velocity = 0
self.acceleration = 0.1
self.velocity_step = 1

self.angular_velocity = 0
self.target_angular_velocity = 0
self.angular_acceleration = 0.1
self.angular_velocity_step = 1
```

– OnDuring

```
self.generate_message('myship_initialized')
```

- State specification 1.2: ***MyShip.operating***

  – OnDuring

```
diff_velocity = self.target_velocity - self.velocity
if abs(diff_velocity) < self.acceleration:
    self.velocity = self.target_velocity
else:
    if diff_velocity > 0:
        self.velocity += self.acceleration
    else:
        self.velocity -= self.acceleration

if abs(self.velocity) < 0.0001:
    self.velocity = 0

diff_angular_velocity = self.target_angular_velocity - self.angular_velocity
if abs(diff_angular_velocity) < self.angular_acceleration*abs(self.velocity):
    self.angular_velocity = self.target_angular_velocity
else:
    if diff_angular_velocity > 0:
        self.angular_velocity += self.angular_acceleration*abs(self.velocity)
    else:
        self.angular_velocity -= self.angular_acceleration*abs(self.velocity)

if abs(self.angular_velocity) < 0.0001:
    self.angular_velocity = 0
```

- State specification 1.3: ***MyShip.operating.waiting_up_key***

  – OnExit

```
if self.target_velocity < self.velocity_step*7.5:
    self.target_velocity += self.velocity_step
    self.engine.p -= 1
```

- State specification 1.4: ***MyShip.operating.waiting_down_key***

  – OnExit

```
if self.target_velocity > self.velocity_step*-7.5:
    self.target_velocity -= self.velocity_step
    self.engine.p += 1
```

- State specification 1.5: ***MyShip.operating.waiting_left_key***
  - OnExit

```
if self.target_angular_velocity < self.angular_velocity_step*7.5:
    self.target_angular_velocity += self.angular_velocity_step
    self.steering.r -= 10
```

- State specification 1.6: ***MyShip.operating.waiting_right_key***
  - OnExit

```
if self.target_angular_velocity < self.angular_velocity_step*7.5:
    self.target_angular_velocity += self.angular_velocity_step
    self.steering.r -= 10
```

1.6.2 Statecharts 1.2: *OtherShip* level 1

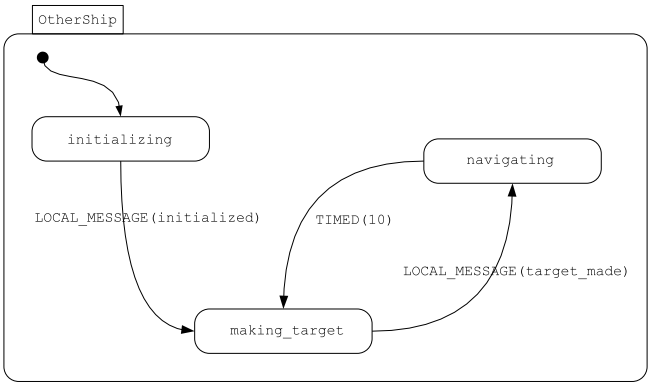

Figure 1.13: Statecharts of ***OtherShip*** level 1.

- State specification 1.7: ***OtherShip.initializing***
  - OnEnter

```
import random
self.x = (random.random() - 0.5) * 1000
self.y = (random.random() - 0.5) * 1000
```

  - OnDuring

```
self.generate_local_message('initialized')
```

- State specification 1.8: ***OtherShip.making_target***
  - OnEnter

```
import random
self.velocity = (random.random() + 0.2) * 5
self.angular_velocity = (random.random() - 0.5) * 10
```

– OnDuring

```
self.generate_local_message('target_made')
```

- State specification 1.9: ***OtherShip.navigating***

– OnEnter

```
self.vx = self.velocity
self.vh = self.angular_velocity
```

1.6.3 Statecharts 1.3: *Camera* level 1

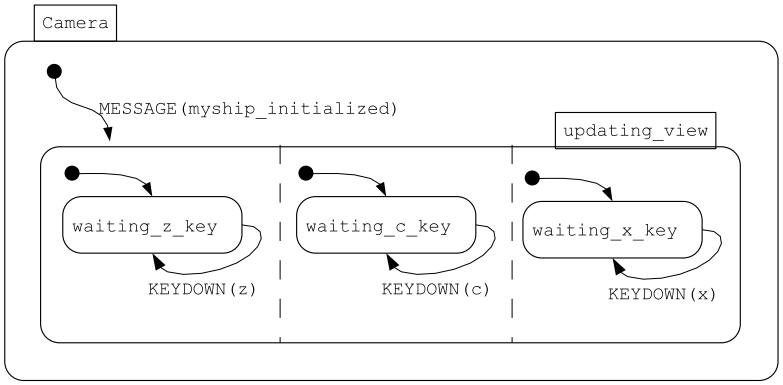

Figure 1.14: Statecharts of ***Camera*** level 1.

- State specification 1.10: ***Camera.updating_view***

– OnDuring

```
self.vy = self.myship.velocity
self.vh = self.myship.angular_velocity
```

- State specification 1.11: ***Camera.updating_view.waiting_z_key***

– OnExit

```
if self.myship.h < 6:
    self.myship.h += 1
```

- State specification 1.12: ***Camera.updating_view.waiting_c_key***

– OnExit

```
if self.myship.h > -6:
    self.myship.h -= 1
```

- State specification 1.13: ***Camera.updating_view.waiting_x_key***

– OnExit

```
self.myship.h = 0
```

Chapter 2

Specification Level 2

2.1 New Features

- Refinement of vessels' behavior (rolling, pitching, and docking)
- Simple special effects (fog, moving texture for wave)
- Collision detection (hierarchical BBound)
- 2D Text (information on voyage)

2.2 Requirements List

- ***ShipSimulator*** shall express wave and fog effects.
- ***MyShip*** and ***OtherShip*** shall express rolling and pitching movement.
- ***ShipSimulator*** shall detects collision between objects including ***MyShip***, ***OtherShip***, and ***Land***.
- ***ShipSimulator*** shall inform ***User*** of ***MyShip***'s collision with ***OtherShip***, ***Land***, and the boundary of the virtual world, then stops ***MyShip*** for 3 seconds.
- ***OtherShip*** shall resolve collision by moving backward for 5 seconds and continue their navigation.
- ***ShipSimulator*** shall inform ***User*** the result, when the docking is successful.
- ***ShipSimulator*** shall display the information on voyage including the velocity, the position, and the target position.

2.3 Storyboards

- Scene 2.1: When ***MyShip*** collided with ***OtherShip***s, ***ShipSimulator*** informs ***User*** the collision and stops ***MyShip***. ***OtherShip***s move backward for 5 seconds, then change their course and continue their navigation..

Figure 2.1: The collision between vessels.

- Scene 2.2: The over view of Scene 2.1.

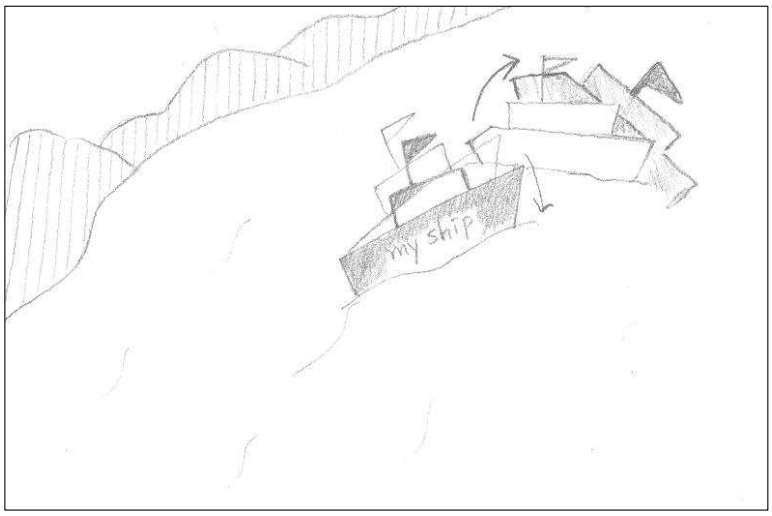

Figure 2.2: The over view of Scene 2.1.

- Scene 2.3: When the docking is successful, ***ShipSimulator***informs ***User*** the results.

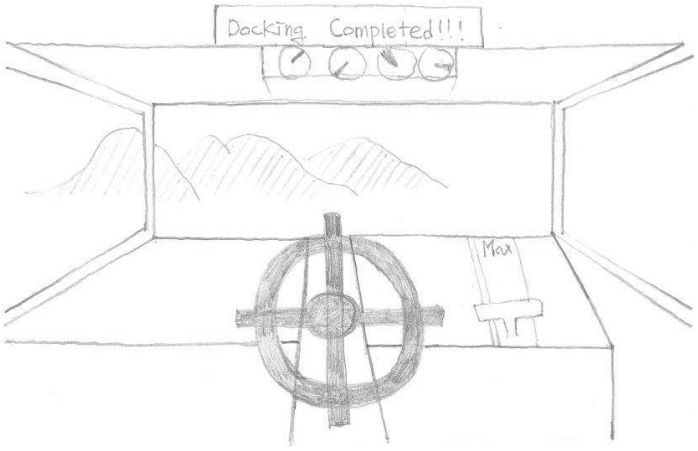

Figure 2.3: The successful docking.

- Scene 2.4: The over view of Scene 2.3.

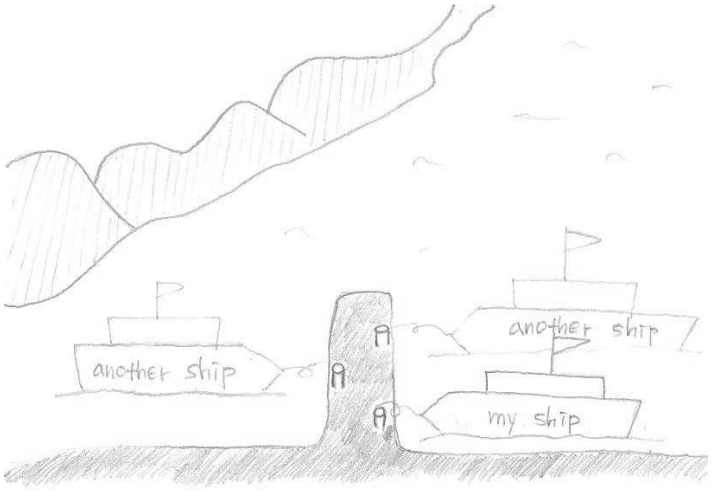

Figure 2.4: The over view of Scene 2.3.

2.4 Message sequence diagrams

- MSD 2.1 : **To resolve the collision.** When collision is detected, ***MyShip*** stops for 3 seconds. ***OtherShip*** moves backward for some 5 seconds, then changes its course and continues its navigation.

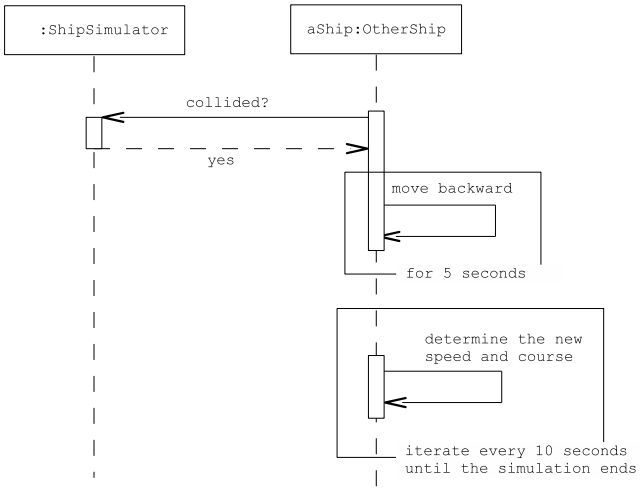

Figure 2.5: To resolve the collision.

- MSD 2.2 : **To dock *MyShip* successfully.** When ***MyShip*** arrives at the defined position successfully, the mission is completed.

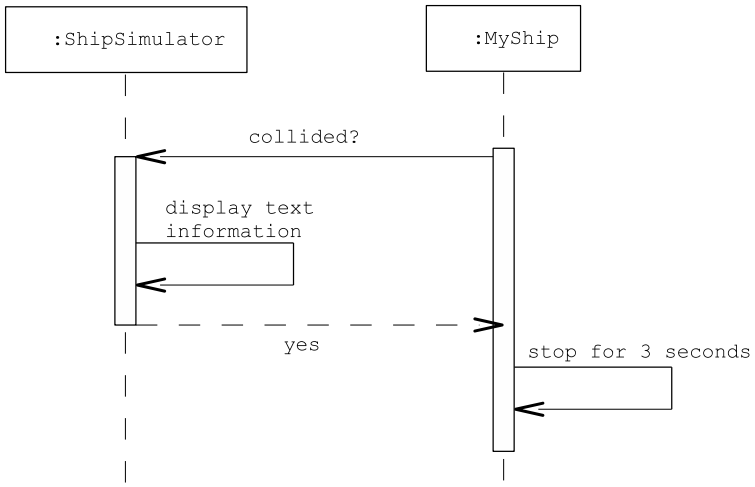

Figure 2.6: To dock ***MyShip*** successfully.

2.5 Scenegraph

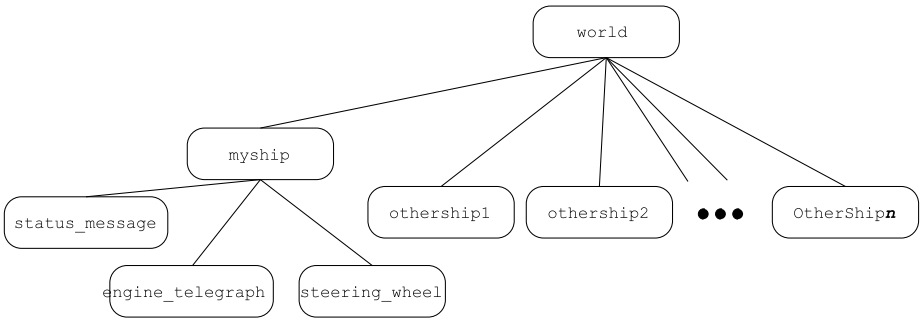

Figure 2.7: Scenegraph level 2.

2.6 Statecharts

2.6.1 Statecharts 2.1: *MyShip* level 2

- State specification 2.1: ***MyShip.initializing***
  - OnEnter

```
self.velocity = 0
self.target_velocity = 0
self.acceleration = 0.1
self.velocity_step = 1

self.angular_velocity = 0
self.target_angular_velocity = 0
self.angular_acceleration = 0.1
self.angular_velocity_step = 1

self.status_message.string = "status: initializing"
```

- State specification 2.2: ***MyShip.operating***
  - OnEnter

```
self.status_message.string = "status: operating"
```

- State specification 2.3: ***MyShip.colliding***
  - OnEnter

```
self.status_message.string = "status: colliding"
```

- State specification 2.5: ***MyShip.docking***
  - OnEnter

```
self.status_message.string = "status: docking"
```

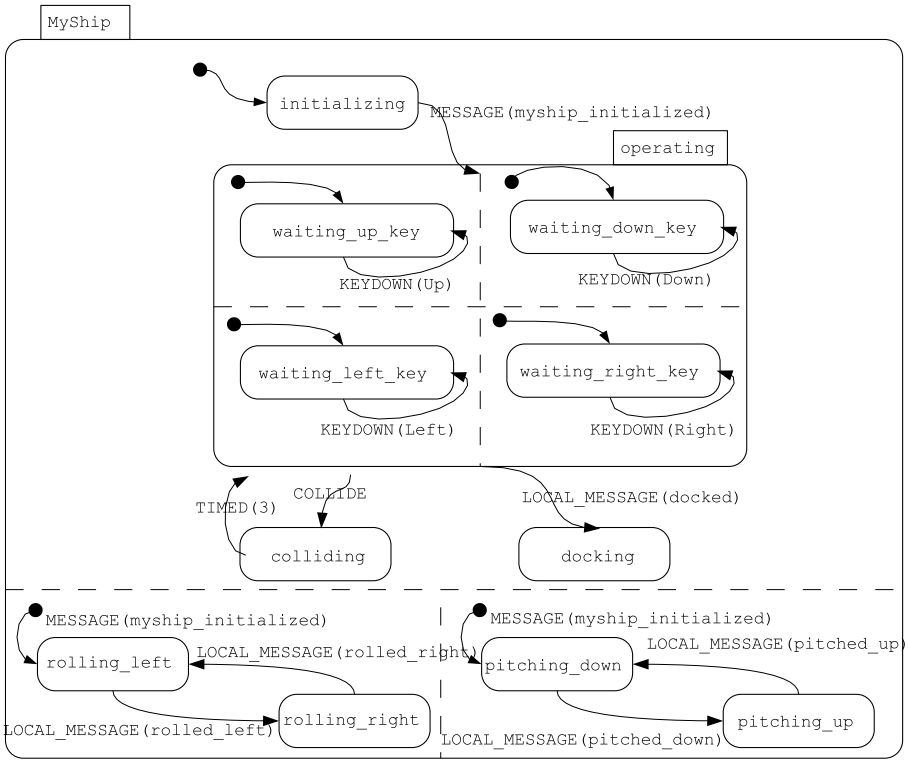

Figure 2.8: Statecharts of ***MyShip*** level 2.

- State specification 2.6: ***MyShip.rolling_left***
  - OnEnter

    ```
    self.vr = -1
    ```
  - OnDuring

    ```
    if self.r < -0.5:
        self.generate_local_message("rolled_left")
    ```
- State specification 2.7: ***MyShip.rolling_right***
  - OnEnter

    ```
    self.vr = 1
    ```
  - OnDuring

    ```
    if self.r > 0.5:
        self.generate_local_message("rolled_right")
    ```
- State specification 2.8: ***MyShip.pitching_down***
  - OnEnter

    ```
    self.vp = -0.5
    ```

– OnDuring

```
if self.p < -0.2:
    self.generate_local_message("pitched_down")
```

- State specification 2.9: ***MyShip.pitching_up***
  – OnEnter

```
self.vp = 0.5
```

  – OnDuring

```
if self.p > 0.2:
    self.generate_local_message("pitched_up")
```

2.6.2 Statecharts 2.2: *OtherShip* level 2

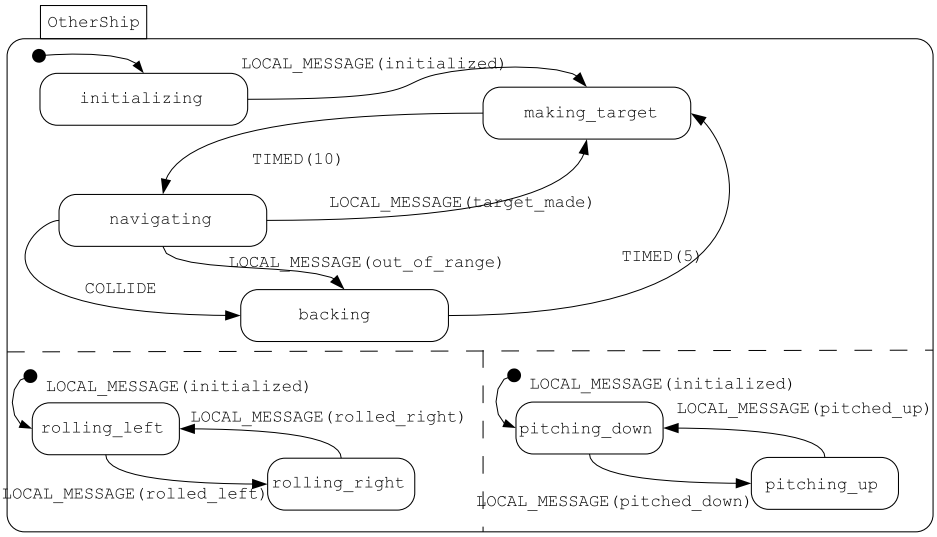

Figure 2.9: Statecharts of ***OtherShip*** level 2.

- State specification 2.10: ***OtherShip.navigating***
  – OnDuring

```
if self.x > 600 or self.x <-600:
    generate_local_message("out_of_range")
if self.y > 600 or self.y < -600:
    generate_local_message("out_of_range")
```

- State specification 2.11: ***OtherShip.backing***
  – OnEnter

```
self.vx = -self.velocity
self.vh = -self.angular_velocity
```

- State specification 2.12: ***OtherShip.rolling_left***
  - OnEnter

    ```
    self.form1.vp = -10
    ```

  - OnDuring

    ```
    if self.form1.p < -3:
        self.generate_local_message("rolled_left")
    ```

- State specification 2.13: ***OtherShip.rolling_right***
  - OnEnter

    ```
    self.form1.vp = 10
    ```

  - OnDuring

    ```
    if self.form1.p > 3:
        self.generate_local_message("rolled_right")
    ```

- State specification 2.14: ***OtherShip.pitching_down***
  - OnEnter

    ```
    self.form1.vr = 5
    ```

  - OnDuring

    ```
    if self.form1.r > 1:
        self.generate_local_message("pitched_down")
    ```

- State specification 2.15: ***OtherShip.pitching_up***
  - OnEnter

    ```
    self.form1.vr = -5
    ```

  - OnDuring

    ```
    if self.form1.r < -1:
        self.generate_local_message("pitched_up")
    ```