Universidad Nacional de Quilmes

Técnicas Avanzadas de Programación – 2010

Extrae agua del pozo antes de que sientas sed – Proverbio chino

Práctica n° 2

Ejercicio 1

El mundo de robots de esta práctica ya no tiene un robot al comenzar. El mismo debe agregarse explícitamente por medio del botón etiquetado "Agregar Robot". Al agregar un robot debe indicar el nombre de la variable con la que lo referencia (en este caso sugerimos utilizar la variable robotech) y la posición inicial del robot en el mundo.

Extienda el comportamiento del robot para que sea capaz de entender los siguientes mensajes. Compruebe que el robot robotech reacciona correctamente.

a) `squareOfSize: aSize`. Realiza un cuadrado con una esquina en su posición actual y de lado aSize.

B) `squareAtHomeOfSize: aSize`. Realiza un cuadrado con una esquina en el origen y de lado aSize.

c) `rotateClockwise: someDegrees`. Rota los grados indicados en sentido de las agujas del reloj

d) `squareOfSize: aSize rotatedBy: someDegrees`. Realiza un cuadrado con una esquina en su posición actual, de lado aSize, rotado someDegrees

Ejercicio 2

Agregue otro robot al que conocerá por medio de la variable "afrodita". Compruebe que la robot afrodita entiende los mensajes definidos en el ejercicio 1 y reacciona de igual forma que el robot "robotech".

¿Qué sucede si modifica uno de los métodos? ¿Siguen comportándose ambos robots de igual manera? ¿Por qué? ¿Qué es lo distinto entre los dos robots? Discuta sus conclusiones con un ayudante.

Ejercicio 3

Hay dos formas de recargar la batería de robotech

a.robotech battery chargeYourself

b.robotech refillBattery

Indique cuál le parece mejor desde el punto de vista del encapsulamiento, y qué cambio en la implementación de los robots podría provocar que una siga andando mientras que la otra no.

Ejercicio 4

Para las siguientes expresiones Smalltalk describa el orden en que se envían los mensajes. Reduzca las expresiones evaluando mensaje por mensaje hasta llegar a un objeto.

a) 3 factorial
b) 7 <= 5
e) #('object' 'oriented') size
f) 4+8 factorial

c) 'objetos' includes: ('Metodologías' at:2)
d) 8 between: 2 and: 60
g) (4+8) factorial
h) 4+8 factorial between: 3+4*10 and: 'hello' size * 8

Ejercicio 5

Ahora que tenemos varios robots en nuestro mundo de robots podemos compararlos. Agregar los siguientes métodos a los robots:

•isNorthFrom: aRobot "true si el receptor está más al norte que aRobot, false en caso contrario"

•isNear: aRobot.

Decimos que dos robots están cerca si su distancia es menor a 10, observar que los puntos (p.ej. el resultado de robotech position) entienden el mensaje dist:, probar

30@40 dist: 27@44.

Ejercicio 6

La forma adecuada para lograr que afrodita se mueva 50 puntos hacia al sur es

afrodita south; move: 50

Una persona pícara podría ver que al moverse, el robot actualiza el valor de su variable position (aunque eso no es tan directo para ver, no es fácil bucear en los métodos). Sabiendo que moverse hacia el sur equivale a sumar en la dirección y, podría hacer

afrodita position: (afrodita position + (0@50))

Ahora bien, un robot se mueve sólo si tiene energía suficiente, y además si está brushDown marca su estela al moverse.

Sabiendo esto ¿qué opina de la forma alternativa de lograr que afrodita se mueva? Relacione su respuesta con el concepto de encapsulamiento.

Ejercicio 7

Agregar los siguientes métodos a los robots:

•goToPosition: aPosition

•move: aDistance northFrom: aRobot

Vale implementar el goToPosition: con dos movimientos, uno en dirección norte-sur y otro en dirección este-oeste. Tener en cuenta que apuntar hacia el norte y moverse una distancia negativa es lo mismo que apuntar para el sur y moverse esa misma distancia positiva, o sea que las dos siguientes expresiones tienen el mismo comportamiento

robotech north; move: -40.

robotech south: move: 40.

move: aDistance northFrom: aRobot mueve al receptor poniéndolo aDistance puntos al norte de aRobot. P.ej. las expresiones

afrodita move: 20 northFrom: robotech

afrodita move: -30 northFrom: robotech

deben hacer que afrodita se mueva hasta ponerse 20 puntos al norte / 30 puntos al sur de robotech respectivamente.

Tener en cuenta que los puntos se saben sumar y restar, probar p.ej.

(20@70)-(0@30)

Ejercicio 8

Una DanceCouple (pareja de baile) esta formada por dos robots; uno conocido como "he" y el otro conocido como "she". Luego de crear una instancia de DanceCouple, se le deben pasar los dos robots como parámetros a un mensaje `#he:she:`.

a) Defina la clase DanceCouple, cree una instancia en el workspace (la parte del ambiente del robot donde se evalúan expresiones), y haga que robotech y afrodita sean los bailarines. Inspeccione el resultado.

Nota: Para agregar una clase, válgase del botón etiquetado ‘‘Agregar Clase’’.

b) agregue los siguientes métodos a DanceCouple:

>>reset

‘‘Ambos bailarines saltan a una posición a distancia 20 del home, enfrentados (uno mira al south y el otro al north).”

>>tangoStep

‘‘He hace un rombo de lado 100, she hace un cuadrado de lado 50.”

>> doTheTango

‘‘Los bailarines repiten tangoStep 5 veces, pero luego de cada una se corren 10 hacia el EAST.”

Ejercicio 9

Una analizador de persecución (PursuitAnalyzer) es un objeto que ayuda a un robot (al que llamaremos *follower*) a alcanzar a otro robot (a que llamaremos *hunted*), proveyendo información útil para quien controla al follower.

a)

Implementar la clase PursuitAnalyzer, cuyas instancias tienen que entender los siguientes mensajes

- follower:hunted: "le decimos cuáles son los robots perseguidor y perseguido"
- isGotcha "true si el follower y el hunted están en la misma posición, false en caso contrario"
- isNear "true si la distancia en línea recta entre follower y hunted es menor o igual a 10, false en caso contrario"
- isUsefulToGoNorth "true si moverse hacia el norte puede ayudar al follower a acercarse al hunter, false en caso contrario".
- straightDistance "la distancia entre follower y hunted en línea recta"
- latitudeDistance "la distancia entre follower y hunted en el eje norte-sur, positiva si el follower está al norte del hunted"
- longitudeDistance "la distancia entre follower y hunted en el eje este-oeste, positiva si el follower está al

este del hunted”

P.ej. si el follower está en 0@0 (el origen) y el hunted está en 30@-40 (30 para el este, 40 para el norte del origen) entonces: la straightDistance es 50 (que es la distancia en línea recta entre el origen y 30@-40), la latitudeDistance es -40 (porque el follower está 40 al sur del hunted), y la longitudeDistance es -30 (porque el follower está 40 al oeste del hunted). Al follower le conviene moverse hacia el norte para acercarse al hunted, por lo tanto isUsefulToGoNorth tiene que devolver true.

Si fuera al revés (follower en 30@-40, hunted en 0@0) entonces sería: straightDistance = 50, latitudeDistance = 40, longitudeDistance = 30, isUsefulToGoNorth = false (si el follower se mueve hacia el norte se aleja del hunted).

b)

Probar con un PursuitAnalyzer que tenga a afrodita como hunted y a robotech como follower. Se recomienda verificar que los mensajes que se le envían al analyzer responden lo que tienen que responder tanto con afrodita en 0@0 y robotech en 30@-40, como con afrodita en 30@-40 y robotech en 0@0.

c)

Cambiar la implementación de los PursuitAnalyzer para que el follower y el hunted se le pasen al analyzer en el momento de crearlo.

Ejercicio 10

Evalúe las siguientes expresiones en el Transcript, e indique cuál es el comportamiento y resultado:

a. 2 + 1
b. #(1 2 3)
c. #(1 2 3) at: 1
d. 'hola, mundo'
e. 'hola', 'mundo'
f. X := 3
g. | n| n := n + 1
h. | n m | n := 1. m:=1. ^n+1
i. 1 + 2 * 3
j. Date today.
k. Time now.
l. 1 > 3
m. 5 between: 1 and: 9
n. 'unString' inspect
o. 1 halt
p. 'otro halt' halt: 'ventana de un debugger'
q. 1* 3; yourself

Ejercicio 11

Suponga que se quiere definir un método como el que sigue:

```
Object>> swap: anObject with: anotherObject
        | temp |
        temp := anObject.
```

anObject := anotherObject.
anotherObject := temp.

Describa los motivos por los cuales no se puede definir este tipo de métodos. Analice el pasaje de parámetros en Smalltalk.

Ejercicio 12

Una persecución activa quedándose al norte (NorthActivePursuit) es una persecución que ayuda activamente a un *follower* a mantenerse al norte de un *hunted.* ¿Cuánto al norte? Una cantidad de unidades que es distinta para cada pursuit.

a)

Implementar la clase NorthActivePursuit, cuyas instancias deben entender los siguientes mensajes

- follower:hunted: "le decimos cuáles son los robots perseguidor y perseguido"
- stayNorth: "le decimos cuánto tiene que estar el follower al norte del hunted"
- moveFollower "hace que el follower se posicione al norte del hunter, a la distancia indicada"

b)

Probar con un pursuit que haga que robotech esté 30 unidades al norte de afrodita.

Ejercicio 13

Los robots pueden informarle de sus movimientos a un objeto; cada vez que un robot se mueve, le avisa a un determinado objeto (ya veremos a cuál) que se movió una cantidad de unidades en una determinada dirección.

¿Cómo avisa el robot a su objeto "amigo"? Obviamente, mandándole un mensaje. El mensaje es moved:inDirection: con la magnitud y la dirección del movimiento como parámetros.

¿Cómo decirle a un robot a qué objeto le tiene que avisar de sus movimientos? Mandándole el mensaje notifyMovesTo:, con el objeto al que se le tiene que avisar como argumento.

a)

Definir la clase SimpleMovementRecorder, cuyas instancias van a saber registrar los movimientos de un robot, y también decidir si un movimiento es corto o largo.

Los SimpleMovementRecorder consideran que un movimiento es "corto" si es de menos de *n* unidades, y largo en caso contrario. El *n* es distinto para cada instancia de SimpleMovementRecorder, se le setea al crearla.

Las instancias de SimpleMovementRecorder tienen que poder entender estos mensajes

- moved:inDirection: "necesario para saber registrar los movimientos de un robot"
- movements "la cantidad de eventos de movimiento registrados"
- maxMovement "la magnitud del más grande de los movimientos registrados"
- totalMovement "el total de unidades de distancia de los movimientos registrados"
- lastMoveWasLittle "true si en el último movimiento se movió menos que una cantidad de referencia, que se

le pasa al recorder al crearlo”

●lastMoveWasBig “true si en el último movimiento se movió al menos la misma cantidad de referencia que para lastMoveWasLittle”

b)

Pasarle a robotech un SimpleMovementRecorder que considere que un movimiento de hasta 20 unidades es chico, y a afrodita otro que considere que un movimiento de hasta 30 unidades es chico.

Los detalles de la relación entre un robot y el objeto al que le informa sus movimientos están en la categoría de métodos notify moves.

Probar moviendo a robotech y a afrodita, y después preguntándole a los recorders.

c)

Pasarle a afrodita el mismo SimpleMovementRecorder que está usando robotech, mover a los dos robots, ver qué responde el recorder.

d)

Implementar un NorthMovementRecorder, al que le interesan los movimientos que fueron exactamente hacia el norte (o sea con direction 0). Tiene que poder decir cuántos hubo de estos, y cuántos hubo seguidos (0 si el último movimiento que le llegó no fue hacia el norte). Probarlo con algún robot que tenga por ahí.

e)

Lograr que los ActivePursuit muevan automáticamente al follower cuando el hunted se mueve, enganchando al pursuit para que sea el objeto que se entere de los movimientos del hunted.