TurtleField Quick Reference

20-NOV-2021, Y. Hayakawa (CDS, Tohoku Univ.)

Turtle Graphics

Go functions associated with Turtle struct

connect to server	Connect(hostname)	use TCP port 2222
clear screen	Clr()	
home position	Home()	origin is the center of square
pen up	Pu()	
pen down	Pd()	
step forward	Fd(step)	
step back	Bk(step)	
jump	Jump(x, y)	unit of length: pixel
relative jump	Rjump(dx, dy)	
turn left	Lt(angle)	
turn right	Rt(angle)	unit of angle: degree
print string	Say(string)	
line mode	Line()	
brush mode	Brush()	
filling mode	Fill()	max # of vertices in filling mode: 6
display card	Card(type,x,y)	type: 1..53
clear card	ClrCard	
query curr. position	Q_Pos()	returns [x, y]
query heading direction	Q_Dir	returns angle
pen color	Col(red, green, blue)	0.0 < red green, blue < 1.0
background color	Bgc(red, green, blue)	

Callback registration

SetHitByBulletCallback(func(Turtle,int))

SetRunIntoTurtleCallback(func(Turtle,int))

SetRunIntoDonutCallback(func(Turtle,int))

SetRunIntoStoneCallback(func(Turtle,int))

SetRunIntoWallCallback(func(Turtle,int))

SetFoundCoinCallback(func(Turtle,int))

SetDetectedByFinder(func(Turtle,int))

SetDetectedByRadarCallback(func(Turtle,int))

SetGotMessageCallback(func(Turtle,int))

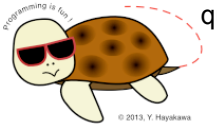
time code is passed in 2nd arg. of the callback

Battles & robot simulations

Regulations/limitations in battle (robot) mode

step size of Fd() & Bk(): up to 2 pixels
turning angle: up to 3 degrees
field is surrounded by rigid walls
only single bullet can exist at a moment in the field for each turtle

	methods	
switch to battle mode	Bmode()	
switch to graphics mode	Gmode()	
shoot a bullet	Fire	
set nickname	Nm(name)	
set my team number	Tm(team_id)	0<=team_id<=4
query number of objects	Q_Nt()	# of turtles and donuts
query to finder	Q_Finder()	returns code
set range of finder	Rf(range)	
query to angular radar	Q_Radar()	returns angle
query to sonar	Q_Sonar()	returns [left, front, right]
query my current score	Q_Score()	returns current score
query team id of turtle met just before	Q_Tm()	returns the id (use this after Q_Ffinder() or Q_Radar() calls)
put a robot turtle in the field	Robot()	
scatter a donut in the field	Donut()	max 10 donuts for each
scatter a coin in the field	Coin()	
borrow 10 coins from the TF owner	BorrowCoin()	max 300 coins in the field
drop a coin at current position	DropCoin()	
pick one coin near current position	PickCoin()	
check number of coin at near distance	Q_Coin()	returns the number
broadcast message to team members	Bcas(message)	
query team message	Q_Bcas()	returns the message



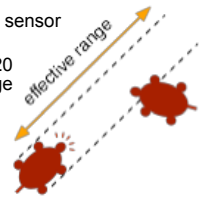
C, Java, Ruby, and Python APIs are available at TurtleField web site:
https://wagtail.cds.tohoku.ac.jp/coda/tfield/index.html

Scoring

initial score of each: 100
point to loose for each shoot: -1
hitting other turtle: +10
hit by bullet:-30
hitting donut within short distance: +10
when score < 0, disconnected

FINDER front sensor

default: 20
use Rf() to change



result code = 0: no objects within effective range
1: turtle 2: donut 3: stone 4: wall found

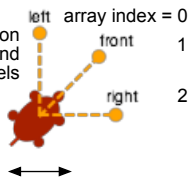
RADAR angular sensor

angle to the nearest object (turtle or donut)

measurement error: ±20 deg

SONAR obstacle detector

for each direction
false: no obstacle, true: obstacle found within 20 pixels



size of turtle: 20 pixels in diameter

Usage of Turtle Field server

right click on screen to pop up menu

press capital **M**: enter maze mode
3: toggle b/w 2D and 3D modes

http port for Turtle Field Live: 2280

clear field	clear drawing
erase zombie	erase turtles that lost TCP connections
toggle graphics/battle mode	in battle mode, background texture changes
toggle private/public mode	in public mode connections from any addr. are accepted. window title bar changes
capture screen	save screen shot in home (UNIX) or desktop (Windows)
kill all turtles	
exit	