

# Turtle Field Quick Reference

19-JUN-2017, Y. Hayakawa (CITE, Tohoku Univ.)

## Python methods in Turtle class

Turtle Graphics

connect to server	Turtle(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)	

Methods need to be overridden to detect events

- hit\_by\_bullet(self,time)
  - run\_into\_turtle(self,time)
  - run\_into\_donut(self,time)
  - run\_into\_stone(self,time)
  - run\_into\_wall(self,time)
  - found\_coin(self,time)
  - detected\_by\_finder(self,time)
  - detected\_by\_radar(self,time)
  - got\_message(self,time)
- time code is passed

## Scoring

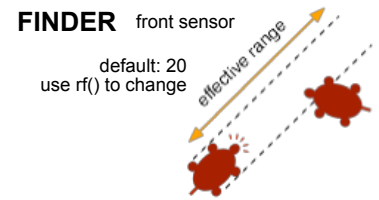
- initial score of each: 100
- point to loose for each shoot: -1
- hitting other turtle: +10
- hit by bullet: -30
- hitting donut within short distance: +10
- when score < 0, disconnected

## Regulations/limitations in battle (robot) mode

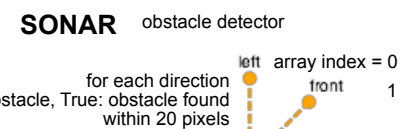
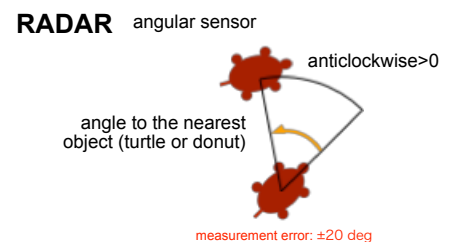
- step size of FD & BK: up to 2 pixels
- turning angle: up to 3 degrees
- field is surrounded by rigid walls
- only single bullet can exist at a moment in the field for each turtle

Battles & robot simulations

	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_FINDER 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() max 300 coins in the field
borrow 10 coins from the TF owner	borrowcoin()
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



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



size of turtle: 20 pixels in diameter



C, Java, and Python APIs are available at TurtleField web site:  
<http://seaotter.cite.tohoku.ac.jp/coda/tfield/index.html>

## Usage of Turtle Field server

right click on screen to pop up menu

- 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

press capital **M**: enter maze mode

**3**: toggle b/w 2D and 3D modes

http port for Turtle Field Live: 2280