Introduction

We have two kinds of triggers, normal triggers and trip wire triggers. A normal trigger looks something like this:

0 norm 1 (c>20)
reinforce 1 64 64 8 40 10 10 0 0 0 0 0 0
ai 1 3
end

The first line,

0 norm 1 (c>20)

Describes the trigger

0 is the trigger number
norm means it's a normal trigger
1 means the trigger gets has one ``life''
(c>20) is the condition the trigger is activated on. In this case, the trigger goes off when **c**, the clock, is larger than 20 seconds.

The next two lines,

reinforce 1 64 64 8 10 10 5 0 0 0 0 0 0
ai 1 3

describe what happens when the trigger goes off.

reinforce 1 64 64

says “put reinforcements for team one at map location (64,64)”

The rest of the line, **8 10 10 5 0 0 0 0 0 0**, says “put 10 troops of type 8 (gray infantry) and 5 troops of type 10 (scythe deamon)

The second line, **ai 1 3**, tells the computer to switch team 1’s ai to type 3, builder.

A trip wire trigger looks similar to a normal trigger:

32 trip 1 (S==1)
reinforce 1 64 64 8 40 10 10 0 0 0 0 0 0
ai 1 3
end

This trigger says goes off when a unit crosses trip wire 32. The condition checks to see if the unit’s team (side) is equal to 1. If it is, the reinforcement event and the ai event occur.

Trigger Conditions

In general, trigger conditions are like **C** conditions. The computer takes an arithmetic expression and evaluates it. If the result is 0 the condition is **false**, otherwise, the condition is **true**. In the introductory example, the variable **c** in **c>20** represents the time, in seconds, since the start of the game. The operator > returns **true** (a none zero number) if the

current time is greater than 20, and **false** (0) otherwise. The following variables and functions currently have special trigger:

t holds the type of a unit that activates a trip wire trigger, only valid for trip wire triggers
S holds the side of a unit that activates a trip wire trigger, only valid for trip wire triggers
c holds the time, in seconds, since the game began
u(*t*) returns the number of units of type ***t*** that have been captured
b(*p*,*t*) returns 0 if player ***p*** has no building of type ***t***. Valid types are:

0	Exo Center
1	Barracks
2	Robot Factory
3	Science Pod
4	Research Pod

s(*p*,*s*) returns a statistic from the statistic tracker. ***p*** is the player, and ***s*** is the statistic "slot." Valid slots are:

1	Money mined by the player, to date
2	Player's kills, to date
3	Player units killed, to date
5	# of mines the player has
6	# of troops the player has
8	# of shots the player's troops have fired
9	# of hit points the player's healers have healed.
10	0 for night, 1 for day (***p*** must equal 0)

s(*p*,*s*,*unit*) returns a statistic from the statistic tracker. ***p*** is the player, ***s*** is the statistic slot, and ***unit*** is the unit number.

0	How many units of that type have been killed
1	Current number of troops for a player
2	Array slot. Set ***p=0***, ***unit*** is the array element

v(*p*,*x*,*y*) returns true of the point **(*x*,*y*)** is visible to player ***p***
r returns a psudio random number.

Trigger Actions

Here are the actions that have, to date, been coded.

ai ***team ai***	Change team ***team***'s ai to ***ai***
die	Program shuts down
reinforce ***team x y type num***	Send reinforcements for team ***team*** to (***x,y***).
type number type num	The 5 ***type num*** entries describe the type of and number
type number type num	of re-inforcing units
bail ***team***	Exit scenario, team ***team*** won.
aimsg ***team msgsize d0 d1 ...***	Sends a message to the ai on team ***team***. ***msgsize*** is the number of items in the message ***d0, d1...*** are the items in the message. See the next section for a description of ai messages.
newrate ***rate x y***	Changes the rate of the vent at (***x y***) to rate.
setlifes ***trigger equation***	Sets the number of lifes in trigger ***trigger*** using equation ***equation***.
setarray ***element equation***	Sets array location ***element*** using equation ***equation***. Array values can be looked up with **s(0,2,*element*)**
waypoint ***x y num x0 y0 x1 y1 ...***	Sets a waypoint for the troop at ***x,y***. ***num*** is the number of waypoints. ***x0 y0*** is the first waypoint, ***x1 y1*** is the second way point (etc)
ally ***team allyteam state***	Either sets or breaks an alliance. If ***state*** is 1 then ***team*** is an ally of ***allytream***. If ***state*** is 0 then ***team*** is not an ally of ***allyteam***.
dfiddle ***team dnum state***	Sets or resets a ***player*** team's ability to build the object described by item ***dnum*** of the depend.txt file. If ***state*** is 0 then the player can't build that item. If ***state*** is 1 then the player can build that item.

AI Messages

These are the messages you can send the builder ai (ai 3).

msgsize	***d0***	***d1***	
2	0	***attack***	Sets the percent of regular units (***attack***) the ai will allocate for attacking.
2	1	***troop_p***	Sets the infantry priority (***troop_p***). The lower the number, the more infantry. The default value is 1.
2	2	***attack_p***	Sets the mech/ scyth deamon priority (***attack_p***). The lower the number, the more mechs. The default value is 1.
2	3	***artillery_p***	Sets the artillery priority (***artillery_p***). The lower the number, the more artillery. The default value is 2.
2	4	***commando_p***	Sets the commando priority (***commando_p***). The lower the number, the more commandos. The default value is 16.
2	5	***troop_p***	Sets the scout priority (***scout_p***). The lower the number, the more scouts. The default value is 2.
3	6	***xd yd***	Sets up a defense point at (***xd,yd***).
3	7	***xd yd***	Deletes a previously set up defense point at (***xd,yd***)
4	8	***goal type arg***	Change what the krusty AI wants to get built. More on this later.
3	9	***xd yd***	Tells the AI to attack a point if there are enemy troops nearby. Note that the AI will not attack if outnumbered.
3	10	***xd yd***	Deletes a previously set "attack point"
3	11	***xd yd***	Tells the AI there's something of interest in the zone. The AI will very likely launch an attack. Note that, if there is nothing of interest, the AI will likely back off.
3	12	***xd,yd***	Delete a previously set interest point

A note on priorities. If you want more of a unit with a default priority of one you should try raising the priority of all of the other units. It's all relative.

Changing what the krusty AI wants to build

The krusty ai has a list of building goals it wants to satisfy. It goes through it's goals, checks each one to see if it has been satisfied and, if it has, goes on to the next one. Here's the list of default goals:

goal	Type	Arg
0	Exploiters	1
1	Building	Barracks
2	Troops	5
3	Building	Robot 1
4	Building	Science 1
5	Troops	10
6	Exploiters	2
7	Troops	15
8	Building	Science 2
9	Building	Robot 2
10	Troops	20
11	Building	Mech Upgrade Attack
12	Building	Mech Upgrade Defense
13	Troops	200

The krusty_ai starts by checking to see if it has an exploiter or LAR. If it does, it goes on to the next goal, build a barracks. If it has a barracks, it builds 5 troops. When that goal is satisfied, it builds a robot factory. etc. The change build goals AI message allows you to change these goals. The ***goal*** parameter in the AI message corresponds to the goal

column in the above table. Similarly, the ***type*** parameter in the AI message corresponds to the kind of thing you want to see build. Valid types are:

type	number
exploiter	0
building	1
troops	2
money	3

The ***arg*** parameter gives some extra information. In the case of the exploiter type it describes the number of exploiters to be built before the goal is satisfied. In the case of the troops type it describes the number of troops to be built before the goal is satisfied. In the case of the money goal it describes the amount of money the computer must have saved before the goal is satisfied. In the case of the building type it describe the kind of building that has to be build. Valid buildings are:

Barracks	0
Robot 1	1
Robot 2	2
Science 1	3
Science 2	4
Research	5
Mech Defense Upgrade	6
Mech Attack Upgrade	7

Additional upgrades can be programmed on request.

So, the following action would make player 1's krusty AI build two exploiters:

aimsg 1 4 8 0 0 2

aimsg 1 4 8 0 0 2
^ ai 1 ^ 4 items ^ build change message ^ goal 0 ^ build exploiter ^ number

If you want to do serious changes you'll probably have to over-write the entire building goal table. Sorry. Note that the maximum number of goals is currently 32. I can increase this on request.

Macros

I've created a macro library to make the triggers more readable. The macros were written using a package called **M4**. Here is a description of the current macros in the library:

Macro: win_trig_num
Evaluates to: 0
Description: Standard number for the "player wins the game" trigger.

Macro: lost_trig_num
Evaluates to: 1
Description: Standard number for the "player loses the game" trigger.

Macro: exo_center(*p*)
Evaluates to: b(*p*,0)
Description: exo_center(*p*) returns true if player *p* has an exo center.

Macro: barracks(*p*)
Evaluates to: b(*p*,1)
Description: barracks(*p*) returns true if player p has a barracks.

Macro: robot_factory(*p*)
Evaluates to: b(*p*,2)
Description: robot_factory(*p*) returns true if player p has a robot factory.

Macro: research_pod(*p*)
Evaluates to: b(*p*,3)
Description: research_pod(*p*) returns true if player p has a research pod.

Macro: money_mined(*p*)
Evaluates to: s(*p*,1)
Description: Returns the amount of money player *p* has mined

Macro: player_kills(*p*)
Evaluates to: s(*p*,2)
Description: Returns the number of units player *p* has killed.

Macro: player_killed(*p*)
Evaluates to: s(*p*,3)
Description: Returns the number of units player *p* has had killed

Macro: player_mines(*p*)
Evaluates to: s(*p*,5)
Description: Returns the number of mines player *p* has

Macro: player_current_troops(*p*)
Evaluates to: s(*p*,6)
Description: Returns the number of troops player *p* currently has

Macro: player_shots_fired(*p*)
Evaluates to: s(*p*,7)
Description: Returns the number of shots player *p*'s troops have fired

Macro: player_points_healed(*p*)
Evaluates to: s(*p*,8)
Description: Returns the number of hit points player *p* has healed.

Macro: is_day
Evaluates to: s(0,9)
Description: Returns true if it is day

Macro: time_of_day
Evaluates to: s(1,9)
Description: Returns a number from 0 to 256.
if it's day, 0 = sunrise, 256 = sunset
if it's night, 0 = sunset, 256 = sunrise.

Macros: player_zero, player_one, player_two, player_three, player_four, player_five, player_six, player_seven
Evaluates to: 0, 1, 2, 3, 4, 5, 6, 7
Description: Player number "short" forms.

Macro: set_attack_priority(*p* , *value*)
Evaluates to: aimsg *p* 2 0 *value*
Description: Sets the attack priority for player *p*. *value* = 0 for all defense, 256 for all attack.

Macro: set_troop_building_priority(*p*, *value*)
Evaluates to: aimsg *p* 2 1 *value*
Description: Set the emphasis on building troops. See AI messages section for better description

Macro: set_robot_building_priority(*p*, *value*)
Evaluates to: aimsg *p* 2 2 *value*
Description: Set the emphasis on building mechs. See AI messages section for better description

Macro: set_artil_building_priority(*p*, *value*)
Evaluates to: aimsg *p* 2 3 *value*
Description: Set the emphasis on building artillery. See AI messages section for better description

Macro: set_cybor_building_priority(*p*, *value*)
Evaluates to: aimsg *p* 2 4 *value*
Description: Set the emphasis on building cyborgs. See AI messages section for better description

Macro: set_troop_building_priority(*p*, *value*)
Evaluates to: aimsg *p* 2 5 *value*
Description: Set the emphasis on building troops. See AI messages section for better description

Macro: set_defense_point(*p*,*x*,*y*)
Evaluates to: aimsg *p* 3 6 *x y*
Description: Sets up a defensive point for player *p* at location <*x*,*y*>

Macro: delete_defense_point(*p*,*x*,*y*)
Evaluates to: aimsg *p* 3 7 *x y*
Description: Deletes a previously set up defensive point for player *p* at location <*x*,*y*>

Macro: set_build_goal(*p*, *goal_num*, *goal_type*, *arg*)
Evaluates to: aimsg *p* 4 8 *goal_num goal_type arg*
Description: Sets a goal for the ai to build. See the AI messages and building goal section for a better description.

Macro: g_exploiter
Evaluates to: 0
Description: Macro for the build an exploiter goal type. Used with set_build_goal macro

Macro: g_troops
Evaluates to: 2
Description: Macro for the build troops goal type. Used with set_build_goals macro

Macro: g_building
Evaluates to: 1
Description: Macro for the build a building goal type. Used with set_build_goal macro

Macro: b_barracks
Evaluates to: 0
Description: Barracks building goal. Used with set_build_goal where *goal_type* is g_building

Macro: b_robot1
Evaluates to: 1
Description: Robot factory level 1 building goal. Used with set_build_goal where *goal_type* is g_building.

Macro: b_robot2
Evaluates to: 2
Description: Robot factory level 2 building goal. Used with set_build_goal where *goal_type* is g_building.

Macro: b_science1
Evaluates to: 3

Description: Science Pod Level 1 building goal. Used with set_build_goal where *goal_type* is g_building

Macro: b_science
Evaluates to: 4
Description: Science Pod Level 2 building goal. Used with set_build_goal where *goal_type* is g_building

Macro: b_research
Evaluates to: 5
Description: Research center building goal. Used with set_build_goal where *goal_type* is g_building

Macro: b_mech_def_up
Evaluates to: 6
Description: Upgrades the mech's defensive armour to level 1. Used with set_build_goal where *goal_type* is g_building

Macro: b_mech_att_up
Evaluates to: 7
Description: Upgrades the mech's attack strength to level 1. Used with set_build_goal where *goal_type* is g_building

Macro: time_trigger(*t, time*)
Evaluates to: *t* norm 1 (c>*time*)
Description: Creates an "alarm clock" trigger, *t*, than goes off when the time passes *time*

Macro: switch_ai(*p*, *ai_type*)
Evaluates to: ai *p ai_type*
Description: Change the ai of player *p* to *ai_type*

Macro: builder_ai
Evaluates to: 3
Description: The ai number of the builder (or krusty) ai. Used with the switch_ai macro

Macro: passive_ai
Evaluates to: 4
Description: The ai number of the passive ai. Used with the switch_ai macro

Macro: captured
Evaluates to: c
Description: Returns the number of units that have been captured.

Macro: spotted(*p*,*x*,*y*)
Evaluates to: v(*p*, *x*, *y*)
Description: Returns true if location <*x*,*y*> is visible to player *p*

Macro: standard_scenario_win_trigger
Evaluates to: 0 norm 1 (b(1,0)==0 && b(2,0) ==0 && b(3,0) ==0 && b(4,0) ==0 && b(5,0) ==0 && b(6,0)==0 && b(7,0)==0)
bail 0
end
Description: Player 0 wins if everybody else's exo center has been destroyed

Macro: standard_scenario_lost_trigger
Evaluates to: 0 norm 1 (b(0,0)==0)
bail 1
end

Description: Player 1 wins if player 0's exo center has been destroyed

Getting the macro stuff to work.

1) Get a copy of m4.
2) If foo is the name of your scenario, and foo.trg is where you have your trigger file, run...
m4 < foo.trg > foo.tro

Note that if you want the standard macro library in your trg file you need to put the line...

include (stdlib.trg)

At the top of your file

An example macro file

Here's an example file that uses macros:

```
include(stdlib.trg)

standard_scenario_win_trigger
standard_scenario_lost_trigger

time_trigger(10,4)
set_attack_priority(player_one,200)

set_troop_building_priority( player_one,4)
set_robot_building_priority( player_one,4)
set_artil_building_priority( player_one,4)
set_cybor_building_priority( player_one,16)
set_scout_building_priority( player_one,1)

switch_ai(player_one,builder_ai)
end
```

standard_scenario_win_trigger makes sure the player wins when he destroys the enemy's exo centers. **standard_scenario_lost_trigger** makes sure he losses when his exo center gets destroyed. The **time_trigger** creates a trigger (trigger 10) that goess off after 4 seconds. It sets the attack priority for player 1 to 200, and then re-writes the building priorities so most of what the ai builds will be scouts. It then changes the ai to the builder_ai type.

Here's what this file looks like after it's been through m4:

```
0 norm 1 (b(1,0)==0 && b(2,0)==0 && b(3,0)==0 && b(4,0)==0 && b(5,0)==0 && b(6,0)==0 && b(7,0)==0
bail 0
end

1 norm 1 (b(0,0)==0)
bail 1
end


10 norm 1 (c>4)
aimsg 1 2 0 200

aimsg 1 2 1 4
```

aimsg 1 2 2 4
aimsg 1 2 3 4
aimsg 1 2 4 16
aimsg 1 2 5 1

ai 1 3
end

The first is, IMHO, more readable.