Starcraft Environment Manual

20^{th} June, 2016

Contents

Chapter 1

Environment

This section will explain how to set up and start a bot with the starcraft environment using the GOAL programming language.

1.1 Chaoslauncher

In order to make use of all the starcraft brood war plugin, you can make use of the application: the chaoslauncher. With this application several plugins can be used like the: *BWAPI Injector* which is neccesary for using the BWAPI library. It is also recommended to make use of the plugin: *APMAlert*, which shows the current actions per minute of all your units together. When the APM of your bot is suddenly very high, your agents might be using to many actions in a row. At last it is also recommonded to make use of the *W-Mode* plugin. This plugin automatically sets your Starcraft game in windowed mode which makes it easier for debugging.

1.2 Installation

- Download the starcraft environment MSI installer. After running the installer you can find in the installation folder the environment jar and a folder with 3 examples bots (one for each race).
- You can import one of the Example bots in your eclipse IDE. In the mas2g are some specified values, this will be discussed in 1.3, but for now you can run the bot for testing purposes.
- After running the bot, the chaoslauncher should be automatically launched. If this is not the case please read 1.3. When running the

chaoslauncher, please check the BWAPI Injector (and W-Mode for your own convenience) if this is not already done. From here you should be able to click on start which will launch the game. The running GOAL bot will automatically connect to the game and start controlling your units. If the game does not start immediately, it could be generating the map data. If so please wait up to 2 minutes before doing anything, generating the map data only has to happen once for each map you want to use.

1.3 The Mas2g

The starcraft environment offers multiple parameters to be set up in the mas2g. Within the mas2g you can specify which map you want to play, specify your own race, give up the map location of your starcraft game, turn the development tool on or off, enable the automenu script and specify which race you want to play against.

```
use StarcraftEnvironment.jar as environment with
    map="(2)Destination.scx",
    own_race="terran",
    starcraft_location="C:\\Starcraft",
    debug="true",
    auto_menu="Single_Player",
    enemy_race="zerg".
```

1.3.1 Map

It is possible to specify which map the chaoslauncher will automatically load when starting the game. This can be done by inserting the following line: *map = <filename>*, where *<filename>* is the exact filename of the map (with extension). Please note that the environment will only choose maps in the directory: *Starcraft/maps/sscai/*. The installer provides the *mapData* of 1 map, however you can use as many maps as you want. When choosing an other map in the *sscai* folder please note that the first time running the environment will take some time (around 2 minutes) to generate the data of the given map. This only has to happen once, so it won't have to generate more than once.

1.3.2 Own Race

You may also specify the race of your bot in the mas2g. This will automatically launch the chaoslauncher with the specified race. You can do this by inserting the following line: *own_race = <RaceName>*, where *<RaceName>* can either be *zerg*, *protoss*, *terran* or *random*. The option *random* will choose one race with 1/3 of a chance for each race.

1.3.3 Starcraft_Location

It is also possible to specify the location of the source map of the starcraft game. When using the starcraft game provided by the environment installer, this feature will automatically start the chaoslauncher when launching the GOAL bot. When the chaoslauncher is already running it won't start again untill you close it. When the Choaslauncher is automatically started by the environment, an automatic script will be written with all the necessary information to run the GOAL bot (so it is recommended to use this feature). You can use this feature by inserting the line: *starcraft_location = <FilePath>*, where *<FilePath>* is the absolute path to the starcraft source folder.

1.3.4 Debug

The Environment also offers a development tool for debugging purposes. With this development tool you can increase or decrease the game speed, enable cheats and draw unit and map details on screen. More information about the development tool can be found at 1.4. For using the development tool you can insert the following line: *debug=<Boolean>*, where *<Boolean>* will indicate for enabling or disabling the development tool.

1.3.5 Auto Menu

The auto menu parameter can be used to quickly go through the menu of the game when starting your agent. This can be used for single player games and multi player games. For using the auto menu function you can insert the following line: *auto_menu=<MenuChoice>*, where *<MenuChoice>* is either *Single_Player* for a single player game or *Multi_Player* for a multi player game.

1.3.6 Enemy Race

The enemy race parameter can be used for specifying which race you want to play against. When an actual enemy race is chosen like: *zerg*, *protoss* or

terran the *enemyRace* percept will indicate against which race you are playing, while when not specifying an enemy race, so when the option: *random* is chosen, the *enemyRace* percept will be *Unknown* untill the opponent is scouted for the first time. For using the enemy race parameter you can insert the following line: *enemy_race=<RaceName>*, where *<RaceName>* can either be *zerg*, *protoss*, *terran* or *random*. The option *random* will choose one race with 1/3 of a chance for each race.

1.3.7 Defining an agent

When defining an agent it is important that the right type is given to the agent. This has to be the same type of the starcraft unit where the first letter is non-capital. So for example when you want to add a terran SCV agent, this can be done by defining the type of this agent as: *terranSCV*. Note that each unit type first begins with the race of the unit and is followed by the exact name of the unit type.

```
define myAgent as agent {
    use MyAgentInit as init module.
    use MyAgent as main module.
    use MyAgentEvent as event module.
}

launchpolicy {
    when type = terranSCV launch myAgent.
}
```

1.4 The Development Tool

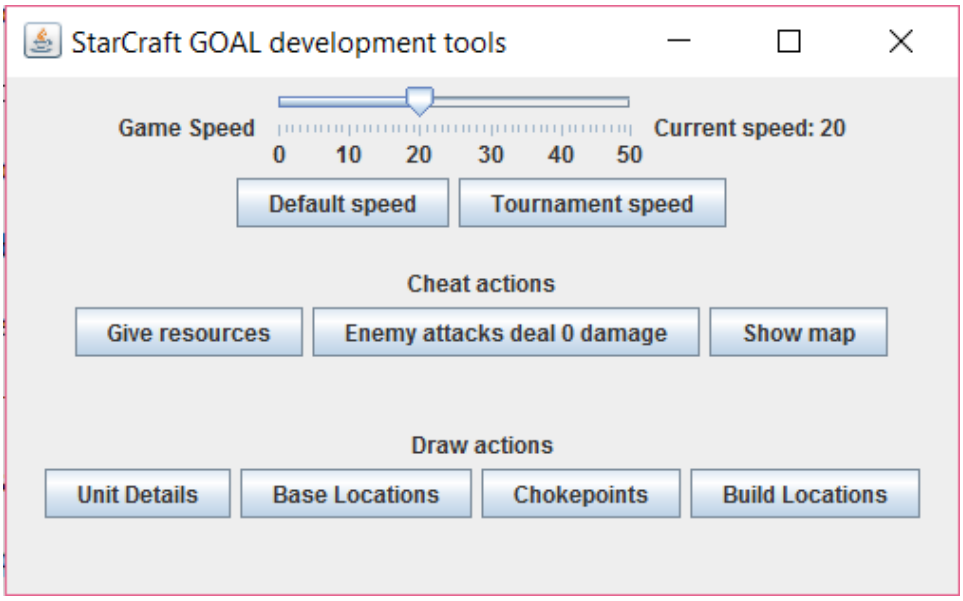

Figure 1.1: Example of the Development Tool

1.4.1 Game Speed

The Game Speed slider can be found at the top of the development tool window. This can be used to quickly change the speed of the game. The initial game speed is set to 20. The fastest game speed is 50 and the slowest game speed is 0. Please note that the agent is supposed to play normally at game speed 30 which is the default game speed for AI tournaments. When the speed is set to 50, the agents can react slower than they would be on the tournament gamespeed. Setting the game speed on 50 should only be used for quick testing purposes.

1.4.2 Cheat Actions

The development tool offers 3 buttons which instantly enable cheats. Note that these cheats should be used for testing purposes only. The first cheat is called: *Give resources* which gives the player 10000 minerals and 10000 gas. The second cheat is called: *Enemy attacks deal 0 damager* which makes the

units of the player immune for damage. The last cheat is called: *Show map* which makes the whole map visible for the player. Note that also all your agents will be perceiving everything on the map.

1.4.3 Map drawing

The development tool can also be used to show map or unit details. There are 4 buttons which can be used. First there is the *Unit Details* button which shows the health and *ID* of every unit. There is also the *Base Locations* button which shows all the starting locations of the map and also all the base locations on the map where players could be expanding to. There is also the *Chokepoints* button which shows all the chokepoints (which are the narrow points where not many units can go through at the same time) on the map. At last there is the *Build Locations* button which shows all the non-obstructed and explored building locations of the map which the worker units perceive with the *constructionSite* percept.

Chapter 2

Percepts

This section will list all the percepts that are usable in the Starcraft environment. The percepts vary per unit, for example: an attacking unit will not perceive the amount of resources available to the player as he does not need them. For the implementation of these percepts in your GOAL code, please refer to the GOAL manual.

2.1 Percepts for all units

These percepts are available to all the units and buildings.

2.1.1 Available Resources

Resources percept

Description	The amount of minerals, gas and supply available to the player. NOTE: supply is multiplied by 2, so 10 supply in game corresponds with 20 supply in the environment.
Type	send on change
Syntax	`resources(<M>, <G>, <CS>, <TS>)`
Example	`resources(350, 100, 25, 41)`

Parameters

`<M>`	The current amount of minerals available to the player.
Type	`Positive Integer`
Range	[0–∞]
`<G>`	The current amount of gas available to the player.
Type	`Positive Integer`
Range	[0–∞]
`<CS>`	The supply of the player which is currently in use.
Type	`Positive Integer`
Range	[0–400]
`<TS>`	The total amount of supply the player can currently use. Note that `<TS>` is always greater or equal to `<CS>`
Type	`Positive Integer`
Range	[0–400]

2.1.2 Unit Information

Self percept

Description	The (unique) *ID* and type of the unit. Also gives information about the maximum health, shield and energy of the unit.
Type	Send once
Syntax	`self(<ID>, <UnitType>, <MaxHealth>, <MaxShield>, <MaxEnergy>)`
Example	`self(21, Terran SCV, 60, 0, 0)`

Parameters

`<ID>`	The (unique) *ID* of the unit.
Type	`Positive Integer`
Range	[0–∞]
`<UnitType>`	The type of the unit. The type of a unit consists of a string with the race of the unit and the name of the unit parted by a space. See Section 6 for the list of all the unit types.
Type	`String`
`<MaxHealth>`	The maximum amount of health of the unit.
Type	`Positive Integer`
Range	[0–2500]
`<MaxShield>`	The maximum amount of shield of the unit.
Type	`Positive Integer`
Range	[0–2500]
`<MaxEnergy>`	The maximum amount of energy of the unit.
Type	`Positive Integer`
Range	[0–2500]

Defensive Matrix percept

Description	Information about how much health the defensive matrix has left on a unit.
Type	Send on change
Syntax	`defensiveMatrix(<health>)`
Example	`defensiveMatrix(200)`

Parameters

`<health>`	The amount of health left of the defensive matrix.
Type	`Positive Integer`
Range	[0–250]

Status percept

Description	The current amount of health, shield and energy of the unit. The `status` percept also shows the conditions of the unit and the current position.
Type	Send on change
Syntax	`status(<Health>, <Shield>, <Energy>, <Cond>, <X>, <Y>)`
Example	`status(250, 0, 0, [moving, carrying], 24, 36)`

Parameters

`<Health>`	The current amount of health of the unit.
Type	`Positive Integer`
Range	[0–`<MaxHealth>`] where `<MaxHealth>` is the maximum health of the given unit.
`<Shield>`	The current amount of shields of the unit.
Type	`Positive Integer`
Range	[0–`<MaxShield>`] where `<MaxShield>` is the maximum shield of the given unit.
`<Energy>`	The current amount of energy of the unit.
Type	`Positive Integer`
Range	[0–`<MaxEnergy>`] where `<MaxEnergy>` is the maximum energy of the given unit.
`<Cond>`	The current condition of the unit. Each unit can have multiple or no conditions depending on the unit and situation. See Section 2.4 for the list of all the conditions.
Type	`List of Strings`
`<X>`	The x-coordinate of the unit in the map.
Type	`Positive Integer`
Range	[0–∞]
`<Y>`	The y-coordinate of the unit in the map.
Type	`Positive Integer`
Range	[0–∞]

2.1.3 Player Percepts

Enemy Race percept

Description	The race of your opponent.
Type	Send once
Syntax	`enemyRace(<Race>)`
Example	`enemyRace(protoss)`

Parameters		
	<Race>	The enemy race which can take the value: protoss, terran, zerg or unknown when the enemy race is not yet known.
	Type	`String`

2.1.4 Map Percepts

Map percept

Description	The width and the height of the map.
Type	Send once
Syntax	`map(<Width>,<Height>)`
Example	`map(96, 128)`

Parameters		
	<Width>	The width of the map.
	Type	`Positive Integer`
	Range	[0–∞]
	<Height>	The height of the map.
	Type	`Positive Integer`
	Range	[0–∞]

Base percept

Description	All the base locations of the map.
Type	Send once
Syntax	`base(<X>,<Y>,<IsStart>,<RegionID>)`
Example	`base(28, 32, true, 8)`

Parameters		
	<X>	The x-coordinate of the base location.
	Type	`Positive Integer`
	Range	[0–∞]
	<Y>	The y-coordinate of the base location.
	Type	`Positive Integer`
	Range	[0–∞]
	<IsStart>	Indicates whether the location is a starting location or not.
	Type	`Boolean (true or false)`
	<RegionID>	The *ID* of the region this location is in. The vespene geyser and all mineral fields will share this region *ID*.
	Type	`Positive Integer`
	Range	[0–∞]

Chokepoint percept

Description	All the chokepoints on the map. These are the narrow points on the map where only a limited amount of units can go through at the same time.
Type	Send once
Syntax	`chokepoint(<X>,<Y>)`
Example	`chokepoint(12, 15)`

Parameters

`<X>`	The x-coordinate of the chokepoint.
Type	`Positive Integer`
Range	[0–∞]
`<Y>`	The y-coordinate of the chokepoint.
Type	`Positive Integer`
Range	[0–∞]

2.1.5 Unit percepts

Attacking percept

Description	Shows the units which are attacking and which units they have targeted.
Type	Send always
Syntax	`attacking(<ID>,<TargetID>,<X>,<Y>)`
Example	`attacking(123, 177, 120, 96)`

Parameters

`<ID>`	The (unique) *ID* of the unit which is attacking.
Type	`Positive Integer`
Range	[0–∞]
`<TargetID>`	The (unique) ID of the targeted unit which is being attacked.
Type	`Positive Integer`
Range	[0–∞]
`<X>`	The x-coordinate of the (attacking) unit.
Type	`Positive Integer`
Range	[0–∞]
`<Y>`	The y-coordinate of the (attacking) unit.
Type	`Positive Integer`
Range	[0–∞]

Unit percept

Description	Shows all units that are currently visible to the player.
Type	Send always
Syntax	`unit(<IsFriendly>,<Type>,<ID>,<Health>,<Shield>,<Condition>)`
Example	`unit(true, Protoss Gateway, 26, 255, 255, [isBeingConstructed])`

Parameters

`<IsFriendly>`	Indicates whether the unit is friendly or not.
Type	`Boolean` (true or false)
`<Type>`	The type of the unit. The type of a unit consists of a string with the race of the unit and the name of the unit parted by a space. See Section 6 for the list of all the unit types.
Type	`String`
`<ID>`	The (unique) *ID* of the unit.
Type	`Positive Integer`
Range	[0–∞]
`<Health>`	The current amount of health of the unit.
Type	`Positive Integer`
Range	[0–`<maxHealth>`] where `<maxHealth>` is the maximum health of the given unit.
`<Shield>`	The current amount of shields of the unit.
Type	`Positive Integer`
Range	[0–`<maxShield>`] where `<maxShield>` is the maximum shield of the given unit.
`<Cond>`	The current condition of the unit. Each unit can have multiple or no conditions depending on the unit and situation. See Section 2.4 for the list of all actual conditions.
Type	`List of Strings`

2.2 Building percepts

These percepts are available to buildings.

2.2.1 Research and Upgrade percepts

HasResearched percept

Description	Indicates which *tech* is already researched. See Section 4 for the list of all actual tech types.
Type	send once
Syntax	`hasResearched(<TechType>)`
Example	`hasResearched(Stim Packs)`

Parameters		
	<TechType>	The *tech* which is currently researched.
	Type	`String`

Upgrading percept

Description	Indicates which *upgrade* is currently being upgraded. See Section 5 for the list of all actual tech types.
Type	Send always
Syntax	`upgrading(<UpgradeType>)`
Example	`upgrading(Stim Packs)`

Parameters		
	<UpgradeType>	The *upgrade* which is currently upgraded.
	Type	`String`

2.2.2 Production Buildings

Queue Size percept

Description	Shows how many units are in queue of the production building.
Type	Send on change
Syntax	`queueSize(<Size>)`
Example	`queueSize(2)`

Parameters		
	<Size>	The size of the current queue.
	Type	`Positive Integer`
	Range	[0–5]

Rally point percept

Description	The exact position of the rallypoint in map coordinates.
Type	Send on change
Syntax	`rallyPoint(<X>,<Y>)`
Example	`rallyPoint(76, 45)`

Parameters

<X>	The x-coordinate of the rallypoint.
Type	`Positive Integer`
Range	[0–∞]
<Y>	The y-coordinate of the rallypoint.
Type	`Positive Integer`
Range	[0–∞]

Rally unit percept

Description	Shows on which unit the rallypoint is set.
Type	Send on change
Syntax	`rallyUnit(<UnitID>)`
Example	`rallyUnit(145)`

Parameters

<UnitID>	The (unique) *ID* the rallypoint points to.
Type	`Positive Integer`
Range	[0–∞]

2.2.3 Loadable Buildings

SpaceProvided percept

Description	Shows how many units are currently loaded in the building and how the maximun amount of units that can be loaded in the building.
Type	Send on change
Syntax	`spaceProvided(<CSize>, <MSize>)`
Example	`spaceProvided(2, 4)`

Parameters		
	<CSize>	The amount of currently loaded units.
	Type	`Positive Integer`
	Range	[0–∞]
	<MSize>	The maximum amount of units that can be loaded.
	Type	`Positive Integer`
	Range	[0–∞]

Unitloaded percept

Description	Shows which unit is loaded inside the given loadable unit.
Type	Send always
Syntax	`unitLoaded(<ID>, <Type>)`
Example	`unitLoaded(154, Terran Marine)`

Parameters		
	<ID>	The (unique) *ID* of the loaded unit.
	Type	`Positive Integer`
	Range	[0–∞]
	<Type>	The type of the loaded unit.
	Type	`String`

2.3 Worker percepts

These percepts are available to worker units.

2.3.1 Worker Management

Worker Activity Percept

Description	Shows the current activity of all friendly workers.
Type	Send always
Syntax	`workerActivity(<ID>, <Activity>)`
Example	`workerActivity(146, gatheringGas)`

Parameters		
	<ID>	The (unique) *ID* of the worker unit.
	Type	`Positive Integer`
	Range	[0–∞]
	<Activity>	The current activity of the worker unit. Can take values: gatheringGas, gatheringMinerals, constructing or idling.
	Type	`String`

Gathering Percept

Description	Shows which mineral or vespene ID the worker is currently gathering from.
Type	Send always
Syntax	`gathering(<ID>)`
Example	`gathering(110)`

Parameters

`<ID>`	The (unique) *ID* of the mineral or vespene geyser.
Type	`Positive Integer`
Range	[0–∞]

2.3.2 Builder Percepts

Vespene Geyser percept

Description	Information about a visible vespene geyser on the map.
Type	Send on change
Syntax	`vespeneGeyser(<ID>,<Resources>,<ResourceGroup>,<X>,<Y>)`
Example	`vespeneGeyser(57, 5000, 6, 22, 32)`

Parameters

`<ID>`	The (unique) *ID* of the vespene geyser.
Type	`Positive Integer`
Range	[0–∞]
`<Resources>`	The amount of resources left in the vespene geyser.
Type	`Positive Integer`
Range	[0–5000]
`<ResourceGroup>`	The resource group of the vespene geyser.
Type	`Positive Integer`
Range	[0–∞]
`<X>`	The x-coordinate of the vespene geyser.
Type	`Positive Integer`
Range	[0–∞]
`<Y>`	The y-coordinate of the vespene geyser.
Type	`Positive Integer`
Range	[0–∞]

ConstructionSite percept

Description	Shows all construction sites on the map, which are explored and not obstructed.
Type	Send always
Syntax	(If Protoss) `constructionSite(<X>,<Y>,<InPylonRange>)` (If Zerg/Terran) `constructionSite(<X>,<Y>)`
Example	`constructionSite(66, 98, false)` `constructionSite(66, 98)`

Parameters

<X>	The x-coordinate of the construction site.
Type	`Positive Integer`
Range	[0–∞]
<Y>	The y-coordinate of the construction site.
Type	`Positive Integer`
Range	[0–∞]
<InPylonRange>	Indicates whether the construction site is in range of a pylon (this is only for protoss)
Type	`Boolean` (true or false)

2.4 Conditions

These are all the possible conditions agents can have in the condition percept.

2.4.1 Worker Conditions

These are the conditions only worker units can have.

`carrying`	Indicates when the worker unit is carrying minerals or vespene gas.
`constructing`	Shows that the worker unit is busy constructing a building.

2.4.2 Building Conditions

These are the conditions only building units can have.

`beingConstructed`	Indicates when a building is being constructed.
`lifted`	Indicates when the building is lifted.
`<addonName>`	Indicates when an addon of the building is present, gives the exact addonname.

2.4.3 Generic Conditions

These are the conditions all units can have.

`idle`	Indicates when the unit is idle (not doing anything).
`cloaked`	Indicates when a unit is cloaked.
`moving`	Shows that a unit is moving.
`following`	Shows that a unit is following an other unit.
`loaded`	Indicates when a unit is loaded.

2.4.4 Zerg Conditions

These are the conditions caused by zerg units.

`burrowed`	Indicates when a zerg unit is burrowed.
`ensnared`	Shows that the unit is ensnared by a Queen unit.
`parasited`	Shows that the unit is parasited by a Queen unit.
`plagued`	Indicates that the unit is plagued by a Defiler unit.
`darkSwarmed`	Indicates that the unit is under a Dark Swarm from a Defiler unit.

2.4.5 Terran Conditions

These are the conditions caused by terran units.

`stimmed`	Indicates when a firebat or marine is stimmed.
`sieged`	Indicates when a siegetank is in siegemode.
`blinded`	Shows when a unit is blinded by a medic.
`lockDowned`	Indicates when a unit is under lockdown by a Ghost unit.
`Irradiated`	Shows when a unit is irradiated by a Science Vessel.

2.4.6 Protoss Conditions

These are the conditions caused by protoss units.

`underStorm`	Shows when a unit is under a storm from a High Templar unit.
`inStatis`	Indicates when a unit is stuck in stasis.
`maelstrommed`	Indicates when a unit is maelstrommed by a Dark Archon.
`disruptionWebbed`	Shows when a unit is in a disruption web from a Corsair.

2.4.7 Unit Percept Conditions

These are the conditions which are visible within the friendly and enemy percept.

`flying`	Indicates whether a unit is flying or not.
`morphing`	Shows when a unit is morphing. (NOTE that sieging and unsieging is also considered morphing)
`cloaked`	Indicates when a unit is cloaked.
`beingConstructed`	Indicates when a unit is being constructed.

Chapter 3

Actions

This section will list all the actions that are usable in the Starcraft environment.

3.1 Attack action

Desription	This action makes a unit which is attack capable, attack the chosen unit.
Syntax	`attack(<TargetID>)`
Parameters	`<TargetID>`: The *ID* of the target that will be attacked.
Pre	The targeted unit is attack capable.
Post	The targeted unit is being attacked by your unit.

3.2 Move action

Desription	Instruct a unit to move to a chosen location.
Syntax	`move(<X>,<Y>)`
Parameters	`<X>`: The x-coordinate of the chosen location `<Y>`: The y-coordinate of the chosen location
Pre	The unit is capable of moving to the chosen location.
Post	The unit moves to the chosen location (ignoring any other unit it might pass by).

3.3 Attack move action

Desription	Go to a location and attack everything you encounter.
Syntax	`attack(<X>,<Y>)`
Parameters	`<X>`: The x-coordinate of the chosen location `<Y>`: The y-coordinate of the chosen location
Pre	The unit is capable of moving to the chosen location.
Post	The unit moves to the chosen locations and attacks any attack capable enemy unit it encounters.

3.4 Upgrade action

Desription	Starts working on the chosen upgrade.
Syntax	`upgrade(<UpgradeName>)`
Parameters	`<UpgradeName>`: The name of the upgrade you want to upgrade.
Pre	The unit is capable of upgrading and has sufficient resources to do so.
Post	The unit starts upgrading the chosen upgrade.

3.5 Build action

Desription	Build a building on a given, not obstructed location.
Syntax	`build(<Type>,<X>,<Y>)`
Parameters	`<Type>`: The Type of the building that has to be built. `<X>`: The x-coordinate of the chosen build location `<Y>`: The y-coordinate of the chosen build location
Pre	The unit is capable of constructing the chosen building and the chosen location is not obstructed.
Post	The unit starts constructing the chosen building at the chosen location.

3.6 Gather action

Desription	Instruct a unit to gather the chosen resource. This can either be minerals or vespene gas.
Syntax	`gather(<ID>)`
Parameters	`<ID>`: The *ID* of the chosen resource.
Pre	The unit is capable of performing the gather action and a valid resource unit is selected.
Post	The unit starts gatering the chosen resource.

3.7 Train action

Desription	Train a chosen unit with a production facility capable of producing the chosen unit.
Syntax	`train(<Type>)`
Parameters	`<Type>`: The type of unit to train.
Pre	The production facility is capable of producing the chosen unit and has sufficient resources to do so.
Post	The production facility starts producing the chosen unit.

3.8 Stop action

Desription	The unit stops performing the action he was busy with.
Syntax	`stop`
Pre	The unit is performing some kind of action.
Post	The unit stops performing the action.

3.9 Ability action

Desription	Use an (researched) ability.
Syntax	`use(<Type>)`
Parameters	`<Type>`: The type of technology to use.
Pre	The chosen tech type is researched and the unit is capable of performing the chosen tech type.
Post	The unit performs the chosen tech ability.

3.10 Ability on target action

Desription	Use an (researched) ability on a target.
Syntax	`use(<Type>, <Target>)`
Parameters	`<Type>`: The type of technology to use. `<Target>`: The target to use the technology on.
Pre	The chosen tech type is researched, the unit is capable of performing the chosen tech type and the chosen target is attack capable.
Post	The unit performs the chosen tech ability on the chosen target.

3.11 Ability on location action

Desription	use an (researched) ability on a location.
Syntax	`use(<Type>, <X>, <Y>)`
Parameters	`<Type>`: The type of technology to use. `<X>`: The x-coordinate of the chosen location `<Y>`: The y-coordinate of the chosen location.
Pre	The chosen tech type is researched, the unit is capable of performing the chosen tech type and the chosen location is valid to perform an action on.
Post	The unit performs the chosen tech ability on the chosen location.

3.12 Research action

Desription	Research a chosen tech type.
Syntax	`research(<Type>)`
Parameters	`<Type>`: The type of tech to research.
Pre	The building is capable of researching the chosen tech type and has sufficient resources to do so.
Post	The building starts researching the chosen tech type.

3.13 Set rally point action

Desription	Set the rally point of a building on a specific location. When the rally point is set, produced units of this production facility will automatically move to this location.
Syntax	`setRallyPoint(<X>, <Y>)`
Parameters	`<X>`: The x-coordinate of the chosen rally location `<Y>`: The y-coordinate of the chosen rally location.
Pre	The building is capable of setting up a rally point and the chosen location is a valid location where units can move to.
Post	The building sets the rally point on the chosen location.

3.14 Set rally point to unit action

Desription	Set the rally point of a building on a unit. When the rally point is set, produced units of this production facility will automatically move to this unit.
Syntax	`setRallyPoint(<Unit>)`
Parameters	`<Unit>`: The unit to set the rally point on.
Pre	The building is capable of setting up a rally point and the chosen unit is on a valid location where units can move to.
Post	The building sets the rally point on the chosen unit.

3.15 Lift action

Desription	Lifts a building which is capable of lifting.
Syntax	`lift`
Pre	The building is capable of flying and is not busy performing any other action.
Post	The building starts flying.
Note	Only for Terran buildings.

3.16 Land action

Desription	Land the unit on a specific, not obstructed location.
Syntax	`land(<X>, <Y>)`
Parameters	`<X>`: The x-coordinate of the chosen land location `<Y>`: The y-coordinate of the chosen land location.
Pre	The unit is currently flying and is capable of landing on the chosen location.
Post	The unit lands on the chosen location.
Note	The location has to be visible.

3.17 Build addon action

Desription	Order a building to build a chosen addon.
Syntax	`buildAddon(<Name>)`
Parameters	`<Name>`: The name of the chosen addon.
Pre	The building is capable of building the addon and does not already have the addon.
Post	The building starts constructing the addon.
Note	Only for Terran buildings.

3.18 Load action

Desription	Order a unit to load into this (loadable) unit.
Syntax	`load(<ID>)`
Parameters	`<ID>`: The *ID* of the unit to load into this (loadable) unit.
Pre	The unit is capable of loading other units inside it and still has enough space prodivded for the targeted unit.
Post	The targeted unit starts walking to the loadable unit and loads into it.

Chapter 4

TechTypes

Here is the list of all tech types that can be researched.

4.1 Terran Units

These are all the Terran tech types.

4.1.1 Battle Cruisers

These are the tech type(s) for Battle Cruiser units.
`Yamato Gun`

4.1.2 Command Centers

These are the tech type(s) for Command Center units.
`Scanner Sweep`

4.1.3 Ghosts

These are the tech type(s) for Ghost units.
`Lockdown`
`Personel Cloaking`
`Nuclear Strike`

4.1.4 Marines and Firebats

These are the tech type(s) for Marine and Firebat units.
`Stim Packs`

4.1.5 Medics

These are the tech type(s) for Medic units.
`Healing`
`Restoration`
`Optical Flare`

4.1.6 Science Vessels

These are the tech type(s) for Science Vessel units.
`Defensive Matrix`
`EMP Shockwave`
`Irradiate`

4.1.7 Siege Tanks

These are the tech type(s) for Siege Tank units.
`Tank Siege Mode`

4.1.8 Vultures

These are the tech type(s) for Vulture units.
`Spider Mines`

4.1.9 Wraith

These are the tech type(s) for Wraith units.
`Cloaking Field`

4.2 Protoss Units

These are all the Protoss tech types.

4.2.1 Arbiters

These are the tech type(s) for Arbiter units.
`Cloaking Field`
`Recall`
`Stasis Field`

4.2.2 Corsairs

These are the tech type(s) for Corsair units.
`Disruption Web`

4.2.3 Dark Archons

These are the tech type(s) for Dark Archon units.
`Feedback`
`Maelstrom`
`Mind Control`

4.2.4 Dark Templars

These are the tech type(s) for Dark Templar units.
`Dark Archon Meld`

4.2.5 High Templars

These are the tech type(s) for High Templar units.
`Archon Warp`
`Psionic Storm`
`Hallucination`

4.3 Zerg Units

These are all the Zerg tech types.

4.3.1 Generic

These are the tech type(s) which all ground units can use.
`Burrowing`

4.3.2 Defilers

These are the tech type(s) for Defilers units.
`Dark Swarm`
`Plague`
`Consume`

4.3.3 Hydralisks

These are the tech type(s) for Hydralisk units.
`Lurker Aspect`

4.3.4 Lurkers

These are the tech type(s) for Lurker units.
`Burrowing` (Can be used without having it researched)

4.3.5 Queens

These are the tech type(s) for Queen units.
`Infestation`
`Parasite`
`Ensnare`
`Spawn Broodlings`

Chapter 5

UpgradeTypes

Here is the list of all upgrade types that can be upgraded.

5.1 Terran Units

These are all the Terran upgrade types for Terran units.

5.1.1 Academy

These are the upgrade type(s) the Academy offers.
`U 238 Shells`
`Caduceus Reactor`

5.1.2 Armory

These are the upgrade type(s) the Armory offers.
`Terran Vehicle Weapons`
`Terran Vehicle Plating`
`Terran Ship Weapons`
`Terran Ship Plating`

5.1.3 Covert Ops

These are the upgrade type(s) the Covert Ops offers.
`Ocular Implants`
`Moebius Reactor`

5.1.4 Engineering Bay

These are the upgrade type(s) the Engineering Bay offers.
`Terran Infantry Weapons`
`Terran Infantry Armor`

5.1.5 Machine Shop

These are the upgrade type(s) the Machine Shop offers.
`Ion Thrusters`
`Charon Boosters`

5.1.6 Physics Lab

These are the upgrade type(s) the Physics Lab offers.
`Colossus Reactor`

5.1.7 Science Facility

These are the upgrade type(s) the Science Facility offers.
`Titan Reactor`

5.1.8 Control Tower

These are the upgrade type(s) the Control Tower offers.
`Apollo Reactor`

5.2 Protoss Units

These are all the Protoss upgrade types for Protoss units.

5.2.1 Arbiter Tribunal

These are the upgrade type(s) the Arbiter Tribunal offers.
`Khaydarin Core`

5.2.2 Citadel of Adun

These are the upgrade type(s) the Citadel of Adun offers.
`Protoss Plasma Shields`

`Leg Enhancements`

5.2.3 Cybernetics Core

These are the upgrade type(s) the Cybernetics Core offers.
`Singularity Charge`
`Protoss Air Weapons`
`Protoss Air Armor`

5.2.4 Fleet Beacon

These are the upgrade type(s) the Fleet Beacon offers.
`Apial Sensors`
`Gravitic Thrusters`
`Argus Jewel`
`Carrier Capacity`

5.2.5 Forge

These are the upgrade type(s) the Forge offers.
`Protoss Plasma Shields`
`Protoss Ground Armor`
`Protoss Ground Weapons`

5.2.6 Observatory

These are the upgrade type(s) the Observatory offers.
`Gravitic Boosters`
`Sensor Array`

5.2.7 Robotics Support Bay

These are the upgrade type(s) the Robotics Support Bay offers.
`Reaver Capacity`
`Scarab Damage`
`Gravitic Drive`

5.2.8 Templar Archives

These are the upgrade type(s) the Templar Archives offers.
`Argus Talisman`
`Khaydarin Amulet`

5.3 Zerg Units

These are all the Zerg upgrade types for Zerg units.

5.3.1 Defiler Mound

These are the upgrade type(s) the Defiler Mound offers.
`Metasynaptic Node`

5.3.2 Evolution Chamber

These are the upgrade type(s) the Evolution Chamber offers.
`Zerg Melee Attacks`
`Zerg Missile Attacks`
`Zerg Carapace`

5.3.3 Hydralisk Den

These are the upgrade type(s) the Hydralisk Den offers.
`Muscular Augments`
`Grooved Spines`

5.3.4 Lair and Hive

These are the upgrade type(s) the Lair and Hive offers.
`Ventral Sacs`
`Antennae`
`Pneumatized Carapace`

5.3.5 Queen's Nest

These are the upgrade type(s) the Queen's Nest offers.
`Gamete Meiosis`

5.3.6 Spawning Pool

These are the upgrade type(s) the Spawning Pool offers.
`Metabolic Boost`
`Adrenal Glands`

5.3.7 (Greater) Spire

These are the upgrade type(s) the (Greater) Spire offers.
`Zerg Flyer Carapace`
`Zerg Flyer Attacks`

5.3.8 Ultralisk Cavern

These are the upgrade type(s) the Ultralisk Cavern offers.
`Chitinous Plating`
`Anabolic Synthesis`

Chapter 6

Unit Types

Here is the list of all unit types you can specify within the mas2g. Note that when you bind your agent to a specific unit type in the mas2g, the first letter of the name unit type should always be non-capital!

6.1 Terran Units

These are all the terran unit types.

6.1.1 Terran Ground Units

These are all the terran ground units.

```
Terran Firebat
Terran Ghost
Terran Goliath
Terran Marine
Terran Medic
Terran SCV
Terran Siege Tank
Terran Vulture
Terran Vulture Spider Mine
```

6.1.2 Terran Air Units

These are all the terran air units.

```
Terran Battlecruiser
Terran Dropship
Terran Science Vessel
Terran Valkyrie
Terran Wraith
```

6.1.3 Terran Building Units

These are all the terran building units.

```
Terran Academy
Terran Armory
Terran Barracks
Terran Bunker
Terran Command Center
Terran Engineering Bay
Terran Factory
Terran Missle Turret
Terran Refinery
Terran Science Facility
Terran Starport
Terran Supply Depot
```

6.1.4 Terran Addons

These are all the terran addon units. Note that terran is the only race capable of making addons.

```
Terran Comsat Station
Terran Control Tower
Terran Covert Ops
Terran Machine Shop
Terran Nuclear Silo
Terran Physics Lab
```

6.2 Protoss Units

These are all the protoss unit types.

6.2.1 Protoss Ground Units

These are all the protoss ground units.

```
Protoss Archon
Protoss Dark Archon
Protoss Dark Templar
Protoss Dragoon
Protoss High Templar
Protoss Probe
Protoss Reaver
Protoss Scarab
Protoss Zealot
```

6.2.2 Protoss Air Units

These are all the protoss air units.

```
Protoss Arbiter
Protoss Carrier
Protoss Corsair
Protoss Interceptor
Protoss Observer
Protoss Scout
Protoss Shuttle
```

6.2.3 Protoss Building Units

These are all the protoss building units.

```
Protoss Arbiter Tribunal
Protoss Assimilator
Protoss Citadel of Adun
Protoss Cybernatics Core
Protoss Fleet Beacon
```

```
Protoss Forge
Protoss Gateway
Protoss Nexus
Protoss Observatory
Protoss Photon Cannon
Protoss Pylon
Protoss Robotics Facility
Protoss Robotics Support Bay
Protoss Shield Battery
Protoss Stargate
Protoss Templar Archives
```

6.3 Zerg Units

These are all the zerg units.

6.3.1 Zerg Ground Units

These are all the zerg ground units.

```
Zerg Broodling
Zerg Defiler
Zerg Drone
Zerg Egg
Zerg Hydralisk
Zerg Infested Terran
Zerg Larva
Zerg Lurker
Zerg Lurker Egg
Zerg Ultralisk
Zerg Zergling
```

6.3.2 Zerg Air Units

These are all the zerg air units.

```
Zerg Cocoon
Zerg Devourer
```

```
Zerg Guardian
Zerg Mutalisk
Zerg Overlord
Zerg Queen
Zerg Scourge
```

6.3.3 Zerg Building Units

These are all the zerg building units.

```
Zerg Creep Colony
Zerg Defiler Mound
Zerg Evolution Chamber
Zerg Extractor
Zerg Greater Spire
Zerg Hatchery
Zerg Hive
Zerg Hydralisk Den
Zerg Infested Command Center
Zerg Lair
Zerg Nydus Canal
Zerg Queens Nest
Zerg Spawning Pool
Zerg Spire
Zerg Spore Colony
Zerg Sunken Colony
Zerg Ultralisk Cavern
```