National University of Singapore
School of Computing
CS1101S: Programming Methodology (Scheme)
Semester I, 2011/2012

Deathcube Contest:
Destroying the Generators

Issued: 31 October 2011

Due: 3 November 2011, before 23:59

Readings:

- SICP: Chapter 3, Section 3.1, 3.2, 3.3
- Concrete Abstractions, Chapter 14
- Lecture notes on Object-Oriented Programming

Urgent Brief

The Cube of Many Faces

Mission Brief by XO Zi Han

Good day, everyone. I hope you've had enough rest last night. We have had updates from our Scouts watching the Death Cube. It appears to exist in many dimensions, and what we see now is only one of its many areas. Each of its areas has its own generator, and it is these that you will be targeting at dawn tomorrow.

We will be sending all of you out in batches. The groupings will be decided by me later today, and bear no indication towards personal ability, for I believe you are all well prepared and ready to do us all proud.

I do have a piece of good news. It seems that, in order to conserve resources, the cloning of the drones has been limited per area of the Death Cube. This, in turn, should prove beneficial to us, it now being easier to take them down.

Your task, then, in these final hours, is to analyse your strategy once more. Sometimes sheer luck can be the deciding factor in a battle; nevertheless, some form of plan often helps.

May the Force guide us all.

This contest consists of **one** task.

The final touches: To identify your character on the planning board more easily, you may customize it by specifying a few optional variables.

Name	Default	Description
`shortname`	First 3 letters of your name	A name not exceeding three characters. e.g. `(define shortname "PAT")`
`sabercolour`	"Red"	A colour for lightsaber attacks. e.g. `(define sabercolour "Blue")`
`lasercolour`	"MediumBlue"	A colour for laser beam attakcs. e.g. `(define lasercolour "Red")`
`spellcolour`	"MediumVioletRed"	A colour for lightning attakcs. e.g. `(define spellcolour "Green")`
`bombcolour`	"OrangeRed"	A colour for bomb explosions. e.g. `(define bombcolour "Orchid")`

Possible weapon colour names can be obtained from:
`http://docs.racket-lang.org/draw/color-database___.html`
(Colour names are not case-sensitive)

Please refrain from using the colour `Gold` as that is reserved for the drones.

NOTE: Your submission should only contain definitions for `make-player` and the above variables at the top level. All other definitions (like helper procedures) should be nested **inside** your definition of `make-player`. The definition of `make-player` should take in two parameters, `name` and `birthplace`.

Damage Range querying

Damage Range: Most weapons have some form of damage range, dealing a random amount of damage between its minimum and maximum. You are now able to find out this minimum and maximum, if you wish to.

`(ask wpn 'min-damage target)` will return you the minimum possible damage, while `(ask wpn 'max-damage target)` will return the maximum. When using these queries, note that the parameter is a `target`, even if the weapon itself usually takes another parameter, like a `direction`, in the case of spells. If `max-damage` (and `min-damage`) return 0, this weapon does not affect the chosen target.

Scoring

The scoring for this contest remains the same as Mission 19. Bots are worth 10 points, while all other kills are worth 5 points. Destroying the generator is worth 100 points, and ends the game. Suicides do not add to points, but does add to both kill and death counters.

When running the simulation (see below for more details), you will notice a text output window to the side. Each line will be in the format of `(name kills deaths points generators-destroyed)`. This will also be shown on the live run.

Ranking will be done based on the following **per-round** statistics: number of points (higher is better), generators destroyed (higher is better), total kills (higher is better), total deaths (lower is better).

Round Progression

The brackets will be selected at random, in groups of 4-6. Each round will end when the generator is destroyed, or when 150 seconds are up. At least two people from each round will progress to the next bracket. More people will progress to the next bracket if there is a tie for second place (i.e. all four per-round statistics are equal).

Equipment and Movement

Your equipment consists of: 2 lightsabers, 1 laser rifle, 2 lightning spellbooks, and 1 bomb. Each turn, the order in which the players move will be randomized, slightly favouring players with higher levels. The players will move first, followed by the drones, followed by the bots.

There may be different maps used for the contest, however, all rounds within a single bracket will share the same map.

Your level in the battle will be your JFDI Academy level as of 3 November 2011, 23:59.

Testing your code

To test your code, place `showdown-<yourname>.rkt` in the `submissions` folder. Your name cannot contain spaces. Next, edit `players0.rkt` in the `brackets` folder, replacing `me` with your name. Lastly, run the code in `contest.rkt`.

As some people required it, `(require racket)` is included by default. This causes issues if you use `()` to specify the empty list. **Please use `'()` (with the quote) or `null` instead.**

Instead of writing to screen, you might notice that the mission log is written to a text file in the `logs` folder. Errors are also written to that text file instead of stopping the code, and each `act` method call is limited to 2 mb of memory and 3 seconds of running time (to prevent infinite loops).

May the Force guide your sword.