Scalatron Game Rules

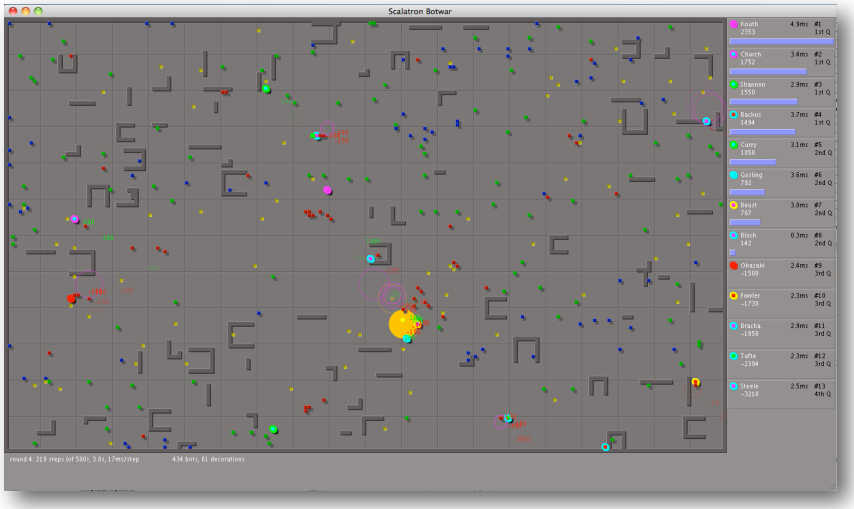
About Scalatron

Scalatron is an educational resource for groups of programmers that want to learn more about the Scala programming language or want to hone their Scala programming skills. It is based on Scalatron BotWar, a competitive multi-player programming game in which coders pit bot programs (written in Scala) against each other.

The documentation, tutorial and source code are intended as a community resource and are in the public domain. Feel free to use, copy, and improve them!

Overview

The behavior of each bot is determined by an associated computer program fragment that implements a control function mapping inputs (what the bot sees) to outputs (how the bot responds). The game server controls the behavior of a collection of non-player characters ("beasts").



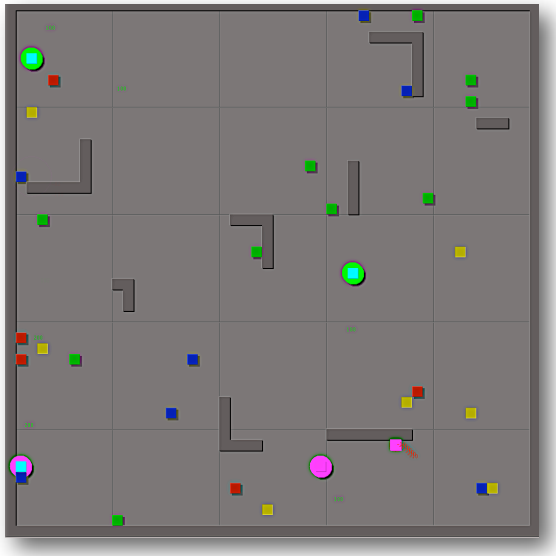
Players' bots must collect energy units ("EUs") by avoiding harmful beasts, hunting edible beasts, harvesting edible plants and steering clear of harmful plants. Walls provide obstacles that bots and beasts must navigate around. Winner is the player whose bot has accumulated the highest energy level at the end of a game round.

The Scalatron BotWar game server runs continuously, playing round after round. Each round lasts for a configurable number of steps (generally about 10,000 steps, or about 3 minutes). At the beginning of each round, a set of bot control functions is created via factories loaded from Java .jar archive files that reside in a set of plug-in directories below a common, shared directory. Once the game started, each player's control function is invoked once per cycle for each entity it controls. The input consists of a string that contains the entity name, an ASCII encoding of the section of the game environment that the bot can see, plus the energy available to the bot. The format is very simple.

After doing some hopefully clever calculations, the control function responds by returning a new string containing its appropriately encoded instructions to the server. The instructions can request various things, such as moving a bot one step to the left or spawning a new mini-bot. After asking all control functions for their responses for all associated bots, the game server updates the game state and displays the new state on the screen.

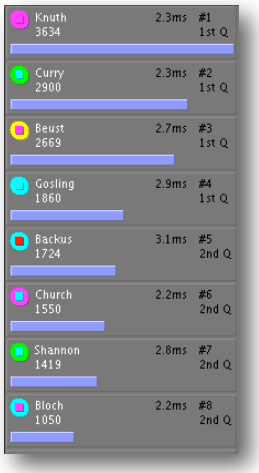
Arena and Entities

The Arena



- The arena consists of a finite, rectangular grid of cells. The size of the arena is configurable at start-up.
- The absolute size of the arena is not known to the player, but is guaranteed to be larger than any bot's view.
- Game entities perceive the arena as infinitely large, since it toroidally wraps back onto itself at the edges (i.e. walking out on one side takes an entity back in on the other side; looking out of the arena on one side, an entity really looks into the arena at the other side, etc.).
- No two entities can occupy the same cell at the same time, i.e. all movements are constrained by collisions with other entities. This constraint allows for an extremely simple encoding of the game world (using one character per cell) when it is passed to the player's control function.
- The origin of the coordinate system is at top left.

The Players



- Each player is represented in the game via a control function, which is instantiated via a factory loaded from a player-supplied plug-in residing in a player-specific sub-directory below a common, network shared directory.
- Each player's control function controls exactly one master bot as well as zero or more mini-bots.
- The commands available to players are listed and described in detail in the section "Control Function Protocol" below.

The Entity Types

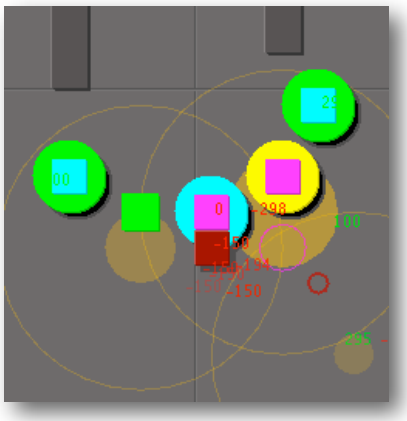
The game contains the following types of entities, each of which is described in more detail in the sections immediately following:

- Bot -- the players' master entity
- Mini-Bot -- slave entities controlled by the players
- Fluppet -- a nice beast (mobile and edible, computer-controlled)
- Snorg -- an evil beast (mobile and predatory, computer-controlled)
- Zugar -- an edible plant (immobile and edible)
- Toxifera -- a poisonous plant (immobile and harmful)
- Wall -- an immobile obstacle

THIS MATERIAL IS IN THE PUBLIC DOMAIN

Entity Details

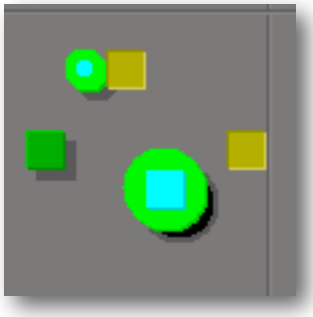
(Master) Bots



- Master bots start each round with an initial energy budget of 1000 EU.
- A master bot sees a limited portion of the game environment. The view of a bot (its "horizon") currently contains the following cells:
  - the cell where the bot resides
  - all adjacent cells within a range of 15 steps in each direction
- This results in a total view size of 31x31 cells. The cell occupied by the bot entity always resides in the center of the view, i.e. at position (15,15).
- The exact size of the view may change in variants or future versions of the game, but the contract that will always be honored is that the view will always be square and its edges will always span an odd number of cells. This makes it possible to equip bots with a resolution-independent view parser.
- The name of the bot that is displayed on the screen is the name of the plug-in directory from which it was loaded. This is to ensure that every bot can reliably be associated with the player responsible for that plug-in directory.
- The `generation` parameter passed to the control function of a bot is always zero.
- Bots can store arbitrary named state properties. This allows control functions to remain stateless by delegating all state maintenance to the server.
- Bots can perform the following operations:
  - `Move()` -- move one step in some direction
  - `Spawn()` -- spawn a mini-bot (slave entity) next to the bot
  - `Say()` -- drop a text 'breadcrumb' into the arena
  - `Status()` -- set a status message to be displayed above the bot
  - `Set()` -- set a state parameter maintained for the bot
  - `Log()` -- write a log message to a debug state parameter

  The commands are documented in more detail in the section "Control Function Protocol".
- Master bots do not consume energy to live or move and can never die.
- On the screen, master bots are big, colorful blobs, with a unique combination of lively colors for every bot.

Mini-Bots



- A mini-bot is spawned by a master bot, which it always remains associated with. The master bot transfers some configurable portion of its energy to the mini-bot when it is created, but at least 100 EU.
- A mini-bot also sees a limited portion of the game environment, one that is slightly smaller than that of a master bot. The view of a mini-bot

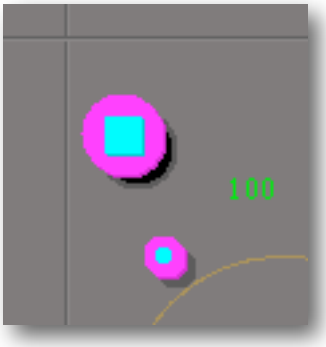
THIS MATERIAL IS IN THE PUBLIC DOMAIN

currently contains the following cells:

- the cell where the mini-bot resides
- all adjacent cells within a range of 10 steps in each direction

  This results in a total view size of 21x21 cells. The cell occupied by the mini-bot entity always resides in the center of the view, i.e. at position (10,10).

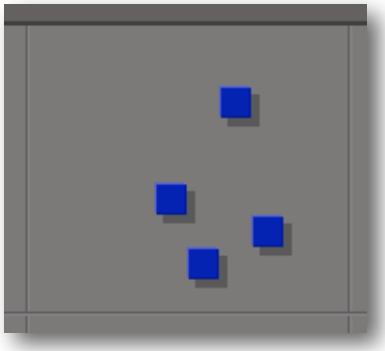
- In addition to observing its surroundings, a mini-bot can also sense the relative position of its master bot. It is provided as part of the control function input. This allows a mini-bot to travel home to its parent bot, generally to be re-absorbed and transfer its energy (for example from plants and beasts it harvested).



- When a bot spawns a new mini-bot, it can provide a name for the new slave entity. This name is passed back to the control function with every request. The control function can also change the name of a slave at any time.
- Mini-bots have `generation` values that are always one higher than the `generation` of the entity that spawned them. Mini-bots spawned by the master have `generation` 1 (one), mini-bots spawned by those have `generation` 2 (two) and so on.
- Like master bots, mini-bots can store arbitrary named state properties. This allows control functions to remain stateless by delegating all state maintenance to the server. The state properties of a mini-bot can also be initialized by the spawning entity to pass information about the desired behavior.
- Mini-bots can perform the following operations:
  - `Move()` – move one step in some direction
  - `Spawn()` – spawn a child mini-bot (Nth-generation slave entity)
  - `Explode()` – convert the energy of the mini-bot into an explosion
  - `Set()` – change the custom state of a bot or mini-bot
  - `Say()` – drop a text 'breadcrumb' into the arena
  - `Status()` – set a status message to be displayed above the bot
  - `Log()` – log debug information into a bot-specific container
- Mini-bots operate at twice the cycle rate of bots, so they can be sent on errands or used to target other bots, which would otherwise be out of reach.
- Unlike bots, mini-bots consume energy just to stay alive: one EU every fourth cycle. When their energy is used up, they disappear. Example: a mini-bot spawned with the default energy of 100 EU will disappear after 400 steps of the game.
- Unlike bots, mini-bots can blow themselves up, for example in order to harm enemy bots, enemy mini-bots or beasts. The blast radius of the explosion can be determined by the mini-bot when it goes, and the damage dealt out by the explosion is credited to the mini-bot's master bot.
- On the screen, mini-bots appear in the same lively color combinations as their master bots, but are much smaller.

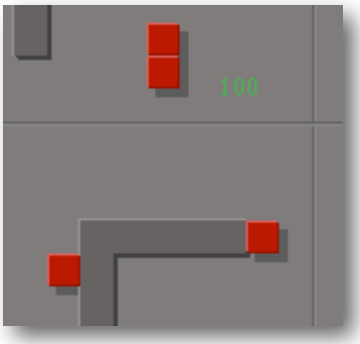
THIS MATERIAL IS IN THE PUBLIC DOMAIN

Fluppets ("nice beasts")



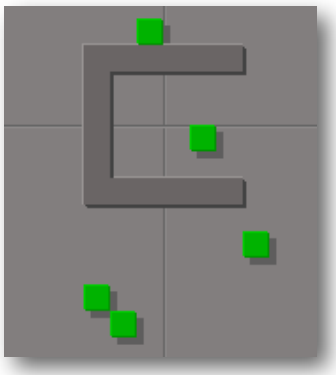
- Fluppets are fearful but nutritious little creatures that roam the arena at half the speed of the bots. They can be hunted by bots for an energy reward of 200 EU each. The energy is transferred during a collision, i.e. in order to consume a Fluppet the bot has to step onto a cell containing it.
- Fluppets are shy and nervous. They run away from approaching bots as soon as they see them. They have a limited horizon, however, and are slower than bots, so they are relatively reliable prey.
- Fortunately, Fluppets are always plentiful, since a new one is born at a random location whenever one was hunted down.
- Fluppets detect and evade bots by their smell, which gets stronger as bots become more successful. The worst players (4th quartile) are perceived only when they are already very close to a Fluppet (6 cells). The best players (1st quartile) are perceived from vast distances (up to 80 cells).
- On the screen, Fluppets appear as blue dots.

Snorgs ("evil beasts")



- Snorgs are rather dumb but highly determined predators that roam the arena searching for bot meat. They will pursue bots as soon as they perceive them and try to bite them, which results in a penalty of 150 EU for the bitten bot.
- Snorgs hunt in packs, but they are slower than bots. By using timely and appropriate evasive maneuvers a bot can generally escape hunting Snorgs unscathed.
- Snorgs are powerful, but their life energy is finite. After seven bites, a Snorg perishes. Nevertheless, Snorgs are also always plentiful, since a new one is born at a random location whenever one expires.
- Snorgs locate and hunt bots by their smell, which gets stronger as bots become more successful. The worst players (4th quartile) are perceived only when the Snorg is very close (6 cells). The best players (1st quartile) are perceived from vast distances (up to 80 cells).
- On the screen, Snorgs appear as red dots.

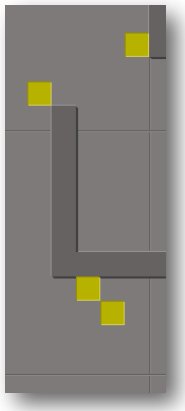
Zugars



- Zugars are nutricious plants that grow all around the arena and are always plentiful, since they are replenished at random locations whenever they are harvested. They are healthy little treats for roaming bots and mini-bots: harvesting a plant (by stepping on it) delivers a 100 EU energy boost.
- As is customary for plants, they do not move around but rather remain where they were seeded until they are eaten.
- On the screen, Zugars appear as green dots.

THIS MATERIAL IS IN THE PUBLIC DOMAIN

Toxifera



- Toxifera are vile, foul-smelling, poisonous growths that sting roaming bots. Stepping on Toxifera is painful for a bot or mini-bot, which loses 100 EU of energy.
- As is customary for plants, they do not move around but rather remain where they were seeded.
- On the screen, Toxifera appear as yellowish dots.

Walls

Walls exist since the beginning of time, are completely inert and immobile and will continue to exist long after the last bot has theft the arena. Running into a wall comes with a 10 EU energy penalty.

Explosions

Mini-bots can detonate themselves to damage other entities with the effects of the resulting explosion. The effects of an explosion are computed as follows:

- energy = the current energy level of the exploding bot
- blastRadius = the blast radius requested via the `Explode` opcode, constrained to the range 2 <= x <= 10
- blastArea = blastRadius * blastRadius * Pi
- energyPerArea = energy / blastArea
- damageAtCenter = energyPerArea * (-200 EU)
- distanceFromCenter = Pythagorean distance of affected entity from exploding mini-bot
- damageForEntity = damageAtEpicenter * (1 - (distanceFromCenter / blastRadius))
- actualDamage = damageForEntity constrained to actually available energy of affected entity

The damages for all affected entities are summed up and the resulting total damage is credited to the master bot that spawned the mini-bot that caused the explosion. Explosions do not damage any of the entities spawned by the master bot, or the master bot itself.

Example A

A mini-bot with an energy of 100 EU detonates with a blast radius of 5 cells:

- blastArea = 5*5*3.14 = 78.5
- energyPerArea = 100/78.5 = 1.274
- damageAtEpicenter = -200*1.274 = 254.7

Another player's bot resides at a (pythagorean) distance of 3 cells:

- damageForEntity = 254.7*(1-3/5)) = 101.9

The damaged bot only has 80 EU left:

- actualDamage = 80 EU

 THIS MATERIAL IS IN THE PUBLIC DOMAIN

Example B

A mini-bot detonates and affects 4 nearby entites:

- another master bot for 80 EU
- another master's mini-bot for 50 EU
- a nearby Snorg for 40 EU
- a more distant Snorg for 30 EU

The master bot that spawned the mini-bot is credited 200 EU.

- size: an integer value 2 < x < 10 indicating the desired blast radius

Collisions

Collision between entities drive the action in the game. They are tested as moves are executed. Move ordering is (from a player perspective) non-deterministic. Should two entities attempt to move to the same cell at the same time, the one whose move is processed first gets to move first, and the laggard's move attempt fails.

Collision Details

Note: in the table below, "bonk" means entities collide and the attempt to move will fail.

- master bot collides with...
  - ...one of its own mini-bots => mini-bot disappears, energy added to bot
  - ...another player's master bot => bonk
  - ...another player's mini-bot => +150, mini-bot disappears
  - ...a Zugar => +100, plant disappears
  - ...a Toxifera => -100, plant disappears
  - ...a Fluppet => +200, beast disappears
  - ...a Snorg => -150, bonk, beast also damaged
  - ...a wall => bonk, stunned for 4 cycles, loses 10 EU
- mini-bot collides with.. .
  - ...a sibling mini-bot => bonk
  - ...its master bot => mini-bot disappears, energy added to bot
  - ...another player's master bot => mini-bot disappears, energy budget lost
  - ...another player's mini-bot => both disappear, energy budgets lost
  - ...a Zugar => +100, plant disappears
  - ...a Toxifera => -100, plant disappears
  - ...a Fluppet => +200, beast disappears
  - ...a Snorg => -150, bonk, beast also damaged
  - ...a wall => bonk, stunned for 4 cycles, loses 10 EU
- Fluppet collides with...

THIS MATERIAL IS IN THE PUBLIC DOMAIN

- ...some player's master bot => beast disappears, bot gets +200
- ...some player's mini-bot => beast disappears, mini-bot gets +200
- ...a Zugar => bonk
- ...a Toxifera => bonk
- ...a Fluppet => bonk
- ...a Snorg => bonk
- ...a wall => bonk

- Snork collides with...
  - ...some player's master bot => beast is hurt, bot gets -150
  - ...some player's mini-bot => beast is hurt, mini-bot gets -150
  - ...a Zugar => bonk
  - ...a Toxifera => bonk
  - ...a Fluppet => bonk
  - ...a Snorg => bonk
  - ...a wall => bonk

THIS MATERIAL IS IN THE PUBLIC DOMAIN

Strategies

Although the game rules are simple and limited, Scalatron bots can use a variety of complex strategies to succeed in the game arena. To get your imagination into gear, here are a few examples for bot strategy and tactics.

Missile Defense

Upon detecting another player's bot in its view, a bot uses the `Spawn()` command to launch a mini-bot that will act as a missile. The mini-bot receives the relative position of the detected opposing bot as a parameter (conveniently encoded within the mini-bot's entity name, which is always provided as a parameter to its control function invocation) and begins to head toward that position (by issuing `Move()` commands that step towards the target position). Once the proximity of an opposing bot is detected within the mini-bot's view it issues an `Explode()` command, converting its stored energy into blast energy. If an opposing bot is damaged by the explosion, the damage points will be credited to the master bot's energy score.

Food Fetcher

Upon detecting a food item (Fluppet or Zugar) in its view, a bot uses the `Spawn()` command to launch a mini-bot that will act as a harvesting drone. The mini-bot receives the relative position of the detected food item as its initial target position and proceeds to head toward it, from then on approaching and ingesting any edible item in its path. Once a certain threshold is reached (say, 1000 EU harvested), the mini-bot uses the relative position of its mast bot (which is passed to its control function) to return home to it and thus to transfer the harvested energy to the master bot's energy score.

Wall Planter

A master bot can also "plant" rows mini-bots as obstacles for other players by spawning them in quick succession at appropriate locations.

Mine Layer

A master bot can lay "mines", mini-bots that remain in position as traps for other players and that detonate as soon as another player's bot approaches within a certain radius.

Fluppet Herder

A master bot can deploy mini-bots to circle around a swarm of Fluppets and to "herd" them toward it. The Fluppets will flee the mini-bots and can then be harvested by the master bot.

Snorg Tempter

A master bot can deploy one or more mini-bots to attract roaming Snorgs and to draw them away from it and towards another player's master bot.

THIS MATERIAL IS IN THE PUBLIC DOMAIN