Wall-in

February 12, 2014

1 What we want

We want a pluggable module to decide how to make our wall, according to user's situation (position, chokes, ...) and desires (low cost, fast to build, no or few gaps for small units, ...).

Question: Is wall strength interesting? Because buildings composing a wall are more or less the sames, and a wall strength is more likely to correspond to the building with the less HP rather than the sum of buildings' HP

1.1 Library functionnality

One easy and straight forward way to proceed is to propose a library with one function only:
Function **makeWall**
Inputs:

- a chokepoint (from BWTA::Chokepoint),
- a boolean telling if one wants a wall inside or outside the user's region,
- what to optimize (cost, time, gaps). Start with one objective at the same time. Multi-objective optimization problems are quite tricky,
- an option for getting outputs as a list of tuples (building, position, timing) or as a list of BWAPI::UnitCommand.

Output: Depending of input options, a list of tuples (building, position, timing) or as a list of BWAPI::UnitCommand.

1.2 Inside makeWall

For BWTA, a chokepoint can have different aspects, and I think it is good to not limit ourselves to chokes linking low- and high-grounds only. See Fig. 1.

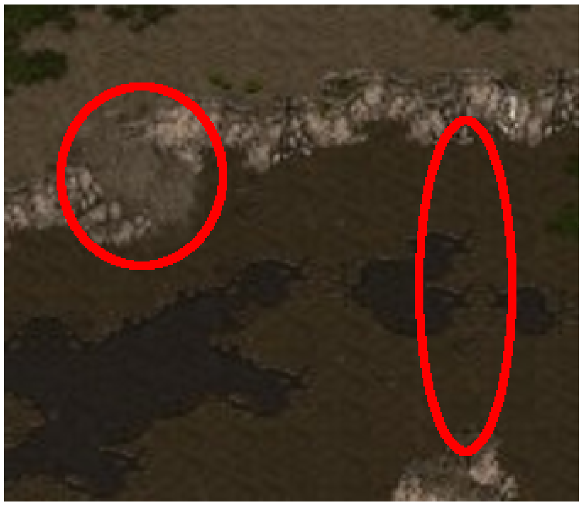

Figure 1: Two different kind of chokes, circled in red.

Let consider one wants to make an inside-wall from choke in Fig. 2.

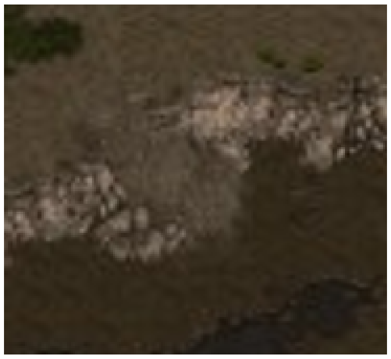

Figure 2: Yeah, this one.

In order to make easier the optimization process, we could model the space around a choke through a 16 × 16 grid of tiles, like Michal proposed in his paper. See Fig. 3.

Then, we mark which cells correspond to unbuildable tiles, and marks also a source and target cell/tile. See Fig. 4. Remark that target cell let a

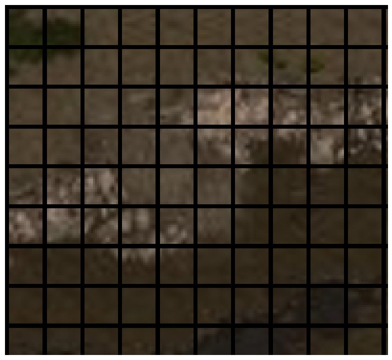
Figure 3: Choke abstracted via a grid.

“door” for an entrance/exit.

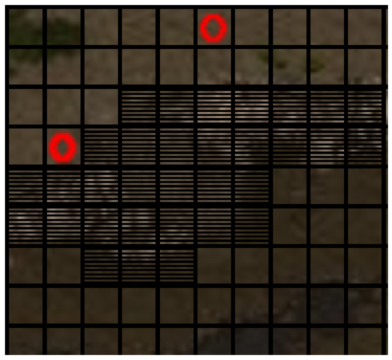
Figure 4: Remove unbuildable tiles and fix a starting and aimed tile.

May be clearer if I remove the background. See Fig. 5.

Finally, knowing if the user wants to build a wall inside or outside his region, we can simplify the grid. Even if it is not shown in Fig. 6, we could just keep 4 tiles to the left/right of the starting tile and 3 tiles above/below the goal, since hugest buildings are 4×3 tiles big.

2 CSP model

Let’s focus an Terran first. A CSP is composed of a set V of variables, a domain D (i.e., values that variables can take) and a set C of constraints.

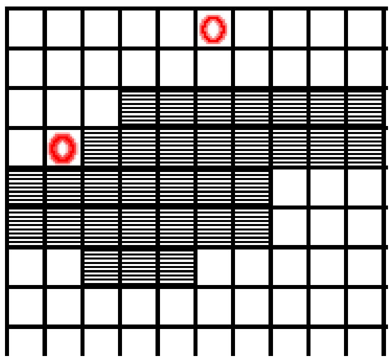

Figure 5: Better indeed.

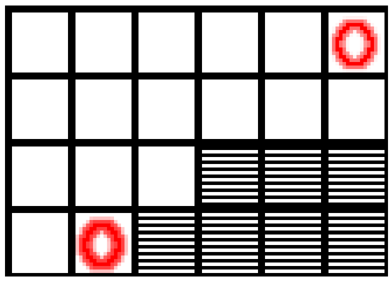

Figure 6: Simplify the grid.

2.1 Informal

Our **variables** are buildings. Thus, V could be composed of:

- an academy,
- barracks,
- an engineering bay,
- a turret,
- a factory,
- **two** bunkers,
- and **two** supply depots.

I think allowing two bunkers and supply depots extends greatly our possibilities (especially for having few gaps in the wall). Why twice these buildings and not the others? Good question; we could indeed double each buildings.

The **domain** is the set of positions (i.e., cells) in the grid. A building position is characterized by its upper-left tile.

Constraints are:

- *Can I build here?* That is, if I place a given building to a given position, will it overlap some marked cells?
- *Non-overlapping buildings*, i.e., if I place a given building to a given position, will it overlap some cells booked for other buildings?
- *No tile-sized gaps between buildings*, that is, our wall must by composed of side-by-side placed buildings.

2.2 More formal

$$\begin{aligned} V &= \{A, B, E, T, F, U_1, U_2, S_1, S_2\} \\ D &= \mathit{Positions} \end{aligned}$$

Let c_{ij} be the cell at row i column j in the grid.

Let t_{ij} be a cell/tile filled by a building b, such that $t_{11} = b$ (recall: the value of b is the upper-left tile of the building), with $1 \leq i \leq b.length$ and $1 \leq j \leq b.height$.

2.2.1 Predicates

- A cell c_{ij} is buildable if the predicate $isBuildable(c_{ij})$ is true.
- A path from the source cell to the target cell exists if the predicate $path(c_{source}, c_{target})$ is true.

2.2.2 Constraints

Can I build here:

$$\forall b \in V, \forall t_{ij} s.t. t_{11} = b, 1 \leq i \leq b.length, 1 \leq j \leq b.height, isBuildable(t_{ij})$$

Non-overlapping buildings:

$$\begin{aligned} &\forall b_1, b_2 \in V, \\ &\forall t^1_{ij} \ s.t.\ t^1_{11} = b_1, 1 \leq i \leq b_1.length, 1 \leq j \leq b_1.height, \\ &\forall t^2_{i'j'} \ s.t.\ t^2_{11} = b_2, 1 \leq i' \leq b_2.length, 1 \leq j' \leq b_2.height, \\ &t^1_{ij} \neq t^2_{i'j'} \end{aligned}$$

No tile-sized gaps between buildings:

$$path(c_{source}, c_{target})$$

2.2.3 About the path

We could run a A*-like algorithm to determine if such a path exists.

If it does not exist, then we need to know how many tiles are missing to make a path (i.e., how many tiles are forming tile-sized gaps). This is necessary for making an objective function in order to allow efficient optimization of this constraint.

Knowing there is no path between the source and the target cell, we could make successive runs of an A*-like algorithm allowing to mark one cell, the two cells if we still have no path by adding one cell, then three cells and so on.