Summarized API for playing Moria

This short document briefly presents the main types, classes and methods that you may need to program your player.

```
// Enum to encode directions.
enum Dir {
  Bottom, BR, Right, RT, Top, TL, Left, LB,
  None,
  DirSize
};

// Defines the type of a cell.
enum CellType {
  Outside, Cave, Abyss, Granite, Rock,
  CellTypeSize
};

// Defines the type of a unit.
enum UnitType {
  Dwarf, Wizard, Orc, Troll, Balrog,
  UnitTypeSize
};

// Simple struct to handle positions.
struct Pos {
  int i, j;
};

Pos::Pos (int i, int j);
// Example: Pos p(3, 6);

ostream& operator<< (ostream& os, const Pos& p);
// Example: cerr << p << endl;

bool operator== (const Pos& a, const Pos& b);
// Example: if (p == Pos(3, 2)) ...

bool operator!= (const Pos& a, const Pos& b);
// Example: if (p != Pos(3, 2)) ...
```

```
// Compares using lexicographical order (first by i, then by j).
// If needed, you can sort vectors of positions or build sets of positions.
bool operator< (const Pos& a, const Pos& b);
// Example: if (p < Pos(3, 2)) ...

Pos& operator+= (Dir d);
// Example: p += Right;

Pos operator+ (Dir d);
// Example: Pos p2 = p + Left;

Pos& operator+= (Pos p);
// Example: p += Pos(3, 2);

Pos operator+ (Pos p);
// Example: p2 = p + Pos(3, 2);

// Returns whether (i, j) is a position inside the board.
bool pos_ok (int i, int j);
// Example: if (pos_ok(i + 1, j - 1)) ...

// Returns whether p is a position inside the board.
bool pos_ok (Pos p);
// Example: if (pos_ok(p1 + Bottom)) ...

// Describes a cell in the board.
struct Cell {
  CellType type; // The kind of cell.
  int owner;     // The player that owns it, otherwise -1.
  int id;        // The id of a unit if present, or -1 otherwise.
  int turns;     // For a rock, times it has to be attacked to destroy it.
  bool treasure; // For a cave, if it has a treasure or not.
};

// Returns a copy of the cell at p.
Cell cell (Pos p);
// Example: Cell c2 = cell(p);

// Returns a copy of the cell at (i, j).
Cell cell (int i, int j);
// Example: Cell c3 = cell(3, 6);

// Describes a unit on the board and its properties.
struct Unit {
  UnitType type; // The kind of unit.
  int id;        // The id for this unit (new orcs may repeat old ids).
  int player;    // The player that owns this unit.
  int health;    // For the Balrog, anything. For the rest, the current health.
  Pos pos;       // The position inside the board.
};
```

```
// Returns a copy of the unit with identifier id.
Unit unit (int id);
// Example: Unit u2 = unit(23);

// Identifier of your player, between 0 and 3.
int me ();

// Returns the identifiers of all the dwarves of a player.
vector<int> dwarves (int player);
// Example: vector<int> d = dwarves(3);

// Returns the identifiers of all the wizards of a player.
vector<int> wizards (int player);
// Example: vector<int> w = wizards(0);

// Returns the identifiers of all the orcs currently alive.
vector<int> orcs ();
// Example: vector<int> v = orcs();

// Returns the identifiers of all the trolls.
vector<int> trolls ();
// Example: vector<int> t = trolls();

// Returns the identifier of the Balrog.
int balrog_id ();
// Example: int bal = balrog_id();

// Returns the current round.
int round ();

// Returns the current number of cells owned by a player.
int nb_cells (int player);

// Returns the number of treasures already accumulated by a player.
int nb_treasures (int player);

// Returns the percentage of cpu time used up to the last round by a player.
// It is in the range [0..1], or -1 if this player is dead.
// Note that this method only works when executed in the judge.
double status (int player);

// Returns a random integer in [l..u]. u - l + 1 must be between 1 and 10^6.
int random (int l, int u);
// Example: if (random(0, 4) < 2) whatever();
// This code executes whatever() with probability 2/5.

// Returns a random permutation of [0..n-1]. n must be between 0 and 10^6.
vector<int> random_permutation (int n);

// A movement is defined by a unit identifier and a direction.
void command (int id, Dir dir);
// Example: command(23, Bottom);
```