Summarized API for playing Mad Max

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 if a cell is empty or it has any special feature on it.
enum CellType {
    Desert, Road, City, Water, Station, Wall,
    CellTypeSize
};

// Defines the type of a unit.
enum UnitType {
    Warrior, Car,
    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);

// Describes a cell in the board, and its contents.
struct Cell {
  CellType type;  // The kind of cell.
  int owner;       // If a city cell, the player that owns it, otherwise -1.
  int id;          // The id of a unit if present, or -1 otherwise.
};

Cell :: Cell (CellType type, int owner, int id);
// Exampe: Cell c(Road, 2, 23);
// A cell on a road, owned by player 2, with unit 23 on it.

// Describes a unit on the board and its properties.
struct Unit {
  UnitType type;  // The kind of unit.
  int id;          // The unique id for this unit during the game.
  int player;      // The player that owns this unit.
  int food;        // For warriors, the current food. For cars, the current fuel.
  int water;       // For warriors, the current water. For cars, nothing.
  Pos pos;         // The position inside the board.
};

Unit :: Unit (UnitType type, int id, int player, int food, int water, Pos pos);
// Example: Unit u(Warrior, 23, 2, 30, 20, Pos(3, 6));

// 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);
```

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

// Tells if a unit can move at this round.
bool *can_move* (**int** *id*);
// Example: if (can_move(23)) ...

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

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

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

// Returns the current number of cities owned by a player.
int *num_cities* (**int** *player*);

// Returns the total score of a player.
int *total_score* (**int** *player*);

// Returns the percentage of cpu time used in 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);