Summarized API for playing The Walking Dead

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

Positions and directions.

```
// Enum to encode directions. Alive units cannot move diagonally
enum Dir {
  Down, DR, Right, RU, Up, UL, Left, LD
};

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

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 + Down)) ...
```

State of the game.

```
// Returns whether pl is a valid player identifier.
bool player_ok (int pl) const;

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

// Defines kinds of cells.
enum CellType {
  Street ,
  Waste
};

// Describes a cell on the board, and its contents.
struct Cell {
  CellType    type; // The kind of cell (street or waste).
  int        owner; // The player that owns it, otherwise -1.
  int           id; // The id of a unit if present, or -1 otherwise.
  bool        food; // Whether it contains food
};

// Defines the type of the unit.
enum UnitType {
  Alive ,
  Dead,
  Zombie
};

// Describes an unit on the board and its properties.
struct Unit {
  UnitType type;            // The type of unit.
  int id;                   // The unique id of this unit during the game.
  int player ;              // The player that owns this unit
                            // (-1 if is a zombie)
  Pos pos;                  // The position on the board.
  int rounds_for_zombie; // Rounds before it becomes a zombie
                         // (-1 if is not being converted to zombie)
```

```
};

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

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

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

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

// Returns the ids of the alive units of a player
vector<int> alive_units(int pl) const;
// Example: vector<int> au = alive_units(3);

// Returns the ids of the dead units of a player
vector<int> dead_units(int pl) const;
// Example: vector<int> du = dead_units(0)

// Returns the ids of the zombies
vector<int> zombies() const;
// Example: vector<int> z = zombies();

// Returns the current strength of a player ( strength_points/alive_units )
int strength(int pl) const;

// Returns the current score of a player.
int score(int pl) const;
```

Command actions.

```
// Commands unit with identifier id to move following direction dir.
void move(int id, Dir dir);
// Example: move(23,Down);
```

Initial settings.

```
// Returns the number of players in the game.
int num_players() const;
```

```
// Returns the number of rounds a match lasts.
int num_rounds () const;

// Returns the number of rows of the board.
int board_rows () const;

// Returns the number of columns of the board.
int board_cols () const;

// Returns the initial number of units per clan
int num_ini_units_per_clan () const;

// Returns the initial number of zombies on the board
int num_ini_zombies () const;

// Returns the initial number of food items on the board
int num_ini_food () const;

// Returns the initial strength of each clan
int clan_ini_strength () const;

// Returns the points obtained after killing a person
int points_for_killing_person () const;

// Returns the points obtained after killing a zombie
int points_for_killing_zombie () const;

// Returns the points obtained for each owned cell at the end of a round
int points_per_owned_cell () const;

// Returns the units of strenght obtained by eating an item of food
int food_strength () const;

// Returns the number of rounds before a bitten/dead person
// becomes a zombie
int rounds_before_becoming_zombie () const;
```

Random.

```
// 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);
```