## Game concept and core rules[[edit](https://en.wikipedia.org/w/index.php?title=Gridwars&action=edit&section=1" \o "Edit section: Game concept and core rules)]

The game is played on a board aka "battlefield"—orthogonal grid of given size drawn on a torus (thus opposite edges of the field are in contact).

Each cell of the battlefield can be either empty or owned by one of several codes competing for the cells of the battlefield. The code which manages to take over the battlefield or owns most cells after a specified number of cycles is the winner.

The original terminology used by EI was peculiar in that it referred to the competing codes as "the warriors" and to the cells as "processors" of a virtual computing grid (hence "the battle for processors") capable, however, of "firing bullets" at each other.

The game proceeds in turns (cycles). At the beginning of the game, each code owns one cell. Every cycle, codes are executed for cells they own. As it happens, framework program supplies the codes with some data: who are the cell's eight immediate neighbors (by warrior number, 0 for free cell) and its own warrior number. Based on this data, warriors can "fire three bullets" at one/two/three of its 8 neighbors. Gridwars II introduced a principal extension of original rules: warriors could now return 32-bit word, called communication variable or comvar for short, which framework program would supply to each of its 8 neighbors during the next cycle of battle execution.

After all of the cells made their shots, control program evaluates how many bullets in total arrived at every cell from the neighboring warriors, i.e. cells executing the same code. Whichever warrior sends more bullets at the cell (and not less than three in total), takes it over. In case of a tie, cell ownership remains the same.

In principle, the game can be played without a computer: on a board whose size is small enough to make it fun. Players can use private boards or paper to specify shooting directions for all of their cells and then show the shots and update the board together.

Alternatively, firing directions can be specified on the main board in the cell-by-cell manner (using matches, for example), opponents taking turns.

**Краткие плавила игры.**

Есть прямоугольное поле с клетками.

Каждая клетка может быть заполнена бойцом или пуста.

Каждый ход обработки клетки клетка получает информацию о своих соседях. Это массив из 8 элементов по количеству соседей.

Каждому бойцу присваивается номер. Если 0, то это пустая клетка, если номер - то это номер бойца. Если номер – то это номер бойца. И последний вариант, если клетка занята своей клеткой, то возвращается 32 битное слово в виде строки: ”010110100100101010101010101001101010101010101100”. Из этого слова легко извлекать значение с определённых битов например так:

parseInt("010110100100101010101010101001101010101010".slice(3, 5), 2)

А таким образом, возможно, проводить обратные операции: Number(105).toString(2), Чтобы составить слово для ответа.

Функция shoot должна стрелять тем, что возвращать номера выстрелов и новое свое 32 битное слово. Это возвращается таким массивом: [5, 5, 5, ”32 битное слово”]. 1 вые 3 элемента номера ячеек выстрелов, последний – новое 32 битное слово, что будет хранить ячейка. И показывать другим. В остальном может иметь много методов, которые может использовать в процессе вычисления.

Учитываются интерпретатором только возвращаемые данные.

После произведения всех выстрелов. Подсчитывается количество выстрелов попавших в какую-либо клетку. Если попадания какого-либо бойца превышают 3 над попаданиями другого бойца, то клетка захватывается этим бойцом. Иначе – клетка остается в прежнем состоянии.

**История проекта.**