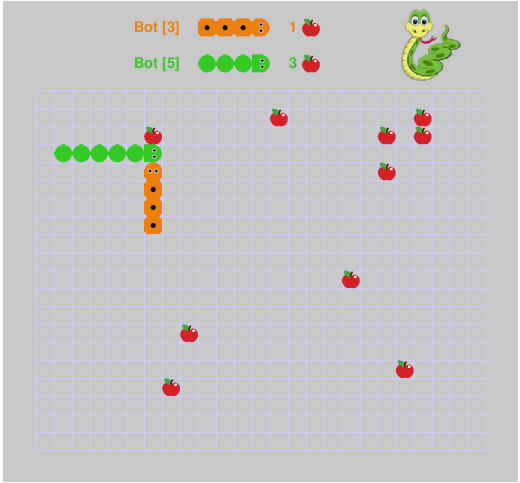
The *snake* game

1 Context

Snake is a classical video game where two snakes have to move around a grid and eat apples. The game is depicted in the figure below and four conditions may lead to the end.

- A snake goes out of the grid and hits the border: it loses.
- A snake hits the body of the other snake: it loses.
- The two snakes move to the same position at the same time and their heads collide: the snake with the higher score wins.
- No apple is eaten for 42 turns: the snake with the higher score wins.



2 Installing the game

To get the code template, go to your favorite coding folder and type:

```
cp -r /opt/duels/games/snake .
```

The files are then ready to use in the `snake` folder.

3 Game description

The initial code is an infinite loop where the *feedback* variable is sent by the game. You have a few milliseconds to compute your *input* and send it to the game. You can fight various AI levels (from 0 to 6) depending on your confidence in your own AI. The code should be compiled according to the included `CMakeLists.txt` file.

3.1 Feedback rules

The feedback variable is a structure containing the following information:

- `pose`: a `Snake` class corresponding to your snake
- `pose_other`: a `Snake` class corresponding to the opponent snake
- `apples`: a `std::vector<Position2D>` containing the positions of the apples

The `Snake` class combines the head and the body parts as such:

- `head`: a `Pose2D` (`x,y,orientation`) for the snake head
  - `orientation` can be `Orientation::LEFT, Orientation::UP, Orientation::RIGHT, Orientation::DOWN`
- `body`: a `std::vector<Position2D>` containing the positions of the body

3.2 Input rules

The game is played on a 20×25 grid. At each new run, the snakes begin with a length of 4 (head + 3 body parts) and a few apples are randomly placed on the grid.
The input to send to the game is defined by the action you want to take:

- `Input::Action::MOVE`: move in the current direction of the head
- `Input::Action::TURN_LEFT` and `Input::Action::TURN_RIGHT`: make a turn with the head and move

4 Expected work

4.1 A class for your AI

In order to design your own AI, you have to:

- Create a class (named e.g. `SnakeAI`) that manages your AI
- The class can have any member variable / function to design your AI
- The class should have at least the following method:

```
// compute next game input from current feedback from the game
Input computeFrom(const Feedback &feedback);
```

where `Input` and `Feedback` are defined in:

- the `<duels/snake/msg.h>` file
- the `duels::snake` namespace

You should of course `include` the file and use either the explicit namespace, or the `using` keyword.

4.2 Programming hints

The `Position2D`, `Pose2D` and `Orientation` classes have useful methods to hand future positions, after a forward, left or right move.