The *gorillas* game

1 Context

Gorillas is a classical video game where two giant gorillas fight on top of a city. Their goal is to hit the other gorilla by throwing explosive bananas.
In this exercice, the goal is to program the behavior of your gorillas and adjust how his banana is thrown in order to hit your opponent.

Figure 1: Game interface with two Gorillas where the right one has just thrown his banana.

2 Installing the game

On the virtual machine or another Ubuntu OS, you have to install the game with:

```
wget -O duels.deb https://box.ec-nantes.fr/index.php/s/7JFHGJ5YGpKW936/download
sudo dpkg -i duels.deb
sudo apt install --fix-missing
```

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

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

The files are then ready to use in the `gorillas` 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 1 to 3) depending on your confidence in your own AI. The code should be compiled according to the included `CMakeLists.txt` file.

3.1 Input rules

The game is played on a 640×335 grid. At each new run, a random city is generated and the two gorillass are randomly placed on top of two different buildings.
The input to send to the game is composed by two member variables:

- `input.vel (float)` : the initial velocity v, in m/s
- `input.angle (float)` : the initial angle θ, in degrees

3.2 Game physics

Once thrown, the banana will fly according to the following differential equation:

$$\begin{cases} \ddot{x}(t) &= -d.(\dot{x}(t)-w).|\dot{x}(t)-w| \\ \ddot{y}(t) &= -d.\dot{y}(t).|\dot{y}(t)| - g \end{cases} \quad \text{with initial conditions} \quad \begin{cases} x(0) &= x_0 \\ y(0) &= y_0 + 18 \\ \dot{x}(0) &= v.\cos\theta \\ \dot{y}(0) &= v.\sin\theta \end{cases}$$

where two values are random but constant for a given game:

- $d \in [0, 0.005]$ is the viscous drag coefficient
- $g \in [8, 11]$ is the gravity force

Other values are either your own input, or found in the `feedback` structure at each turn:

- `feedback.x` : the x-position x_0 of your gorillas
- `feedback.y` : the y-position y_0 of your gorillas
- `feedback.xo` : the x-position of your opponent
- `feedback.yo` : the y-position of your opponent
- `feedback.xb` : the final x-position of your last banana throw
- `feedback.yb` : the final y-position of your last banana throw
- `feedback.wind` : the upcoming wind $\in [-5, 5]$ (random but constant during a given turn)

As soon as the banana passes at less than 5 from a building, or a gorillas (which are of radius 12) it explodes with a radius of 15. If a gorillas is touched by the explosion then it loses the game.

3.3 City information

Another member variable `feedback.building` is an array of 640 `int` corresponding to the highest building at each x position.
As such, it will be updated if the top of a building was damaged by a banana (grey building in the image).
On the opposite, it will not be updated if only the side of a building is damaged (blue building).

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. `GorillAI`) 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/gorillas/msg.h>` file
- the `duels::gorillas` namespace

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

4.2 Programming hints

There are two ways to approach your AI:

- simple: forget about physics, update your input according to the previous x-error
- difficult: try to identify the physics after several turns, then compute the perfect input

Besides, here are some useful tips:

- in difficulty 1, the game AI will not even try to win - you can test your code here
- during the first turn, the wind is always 0
- it is possible to use a constant angle and only ajust the velocity
- do not consider the height of the building at first, it may lead to a very complex approach