# "Lunar Lander" Game

## Algorithmics and Programming Techniques

C. Ferry[1]

[1]Department of Information Technologies
ENS Rennes

September 11, 2015

# Outline

# Purpose and goal

- Begin programming in teams
- Have the prettiest possible code
- Use optimized algorithmic techniques
- Subject: a lunar lander game
- I was given the game engine

## Creation of on-screen Turtle objects

- Terrain, ship generation via `compound` objects
- Shape registration
- Instantiation of classes

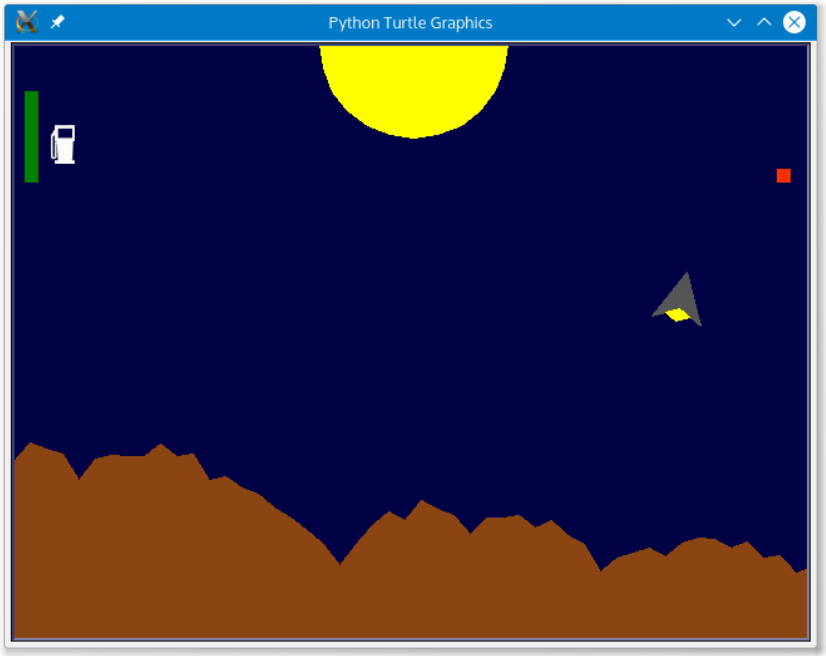
```
super().__init__(-WIDTH/2+20,130,0,0,'fuel','green')
```

# Terrain Generation

- Initial idea: random slopes
- Fractal generation of mountains
- Ground is divided into $n$ equal parts

## Fractal algorithm

- Existing segment: $[(x_0, y_0); (x_1, y_1)]$; $r \in \mathbf{R}^+$
- $x_m = \frac{(x_0+x_1)}{2}$ is assigned: $y_m = y(x_m) + \text{rand}(-r; r)$
- Recursive call with $r/2$ for $[(x_0, y_0); (x_m, y_m)]$ and $[(x_m, y_m); (x_1, y_1)]$



# Collision detection

- Lander radius: $2B$
- Lander - ground: point-to-line $(ax + by + c = 0)$ distance

$$d = \frac{|ax_L + by_L + c|}{\sqrt{a^2 + b^2}} \leqslant 2B$$

## Addressing the issue of high slopes

$d$ is only calculated if

$$x_0 \leqslant x_L \leqslant x_1 \text{ and } y_L \leqslant \max[y_0, y_1]$$

- Bullets, enemies, sun: bounding circles

$$d(P, L) \leqslant 2B + r(P)$$

# Scrolling

- Scrolling when $|x_L| \geqslant \frac{W}{6}$
- Bullets and enemies scroll along
- On the two borders: switching between primary/secondary grounds

**Consequence of scrolling**

Enemies and bullets disappear when changing grounds!

# More things to do

- Rewrite the collision detection mechanism to pixel level
- Implement levels (increasing difficulty?)
- Add fuel loading stations (with wider levels)
- Smart enemy shooting mechanism

# Screencast

- Show it...