



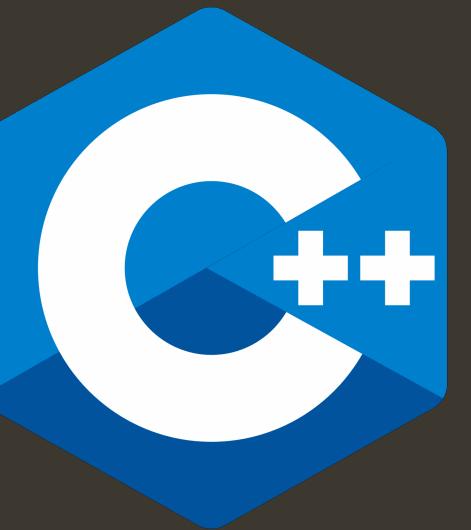
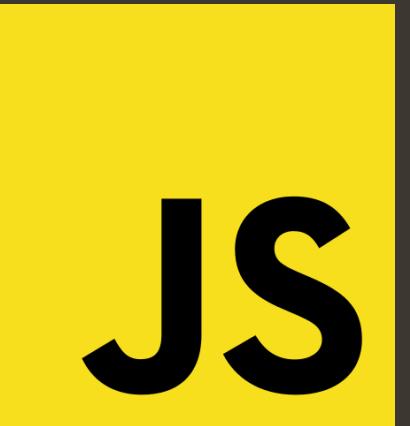
POLYGLOT PONG

PONG
IN
DIVER
SI
LINGUAGGI

POLIGLOTTA?

INTRODUZIONE ALLA PROGRAMMAZIONE, TRA DIVERSI LINGUAGGI

- **Demo Pong single-player**
- **Python vs JS**
- **Demo Pong multi-player Rust**
- **Rust vs C++**
- 

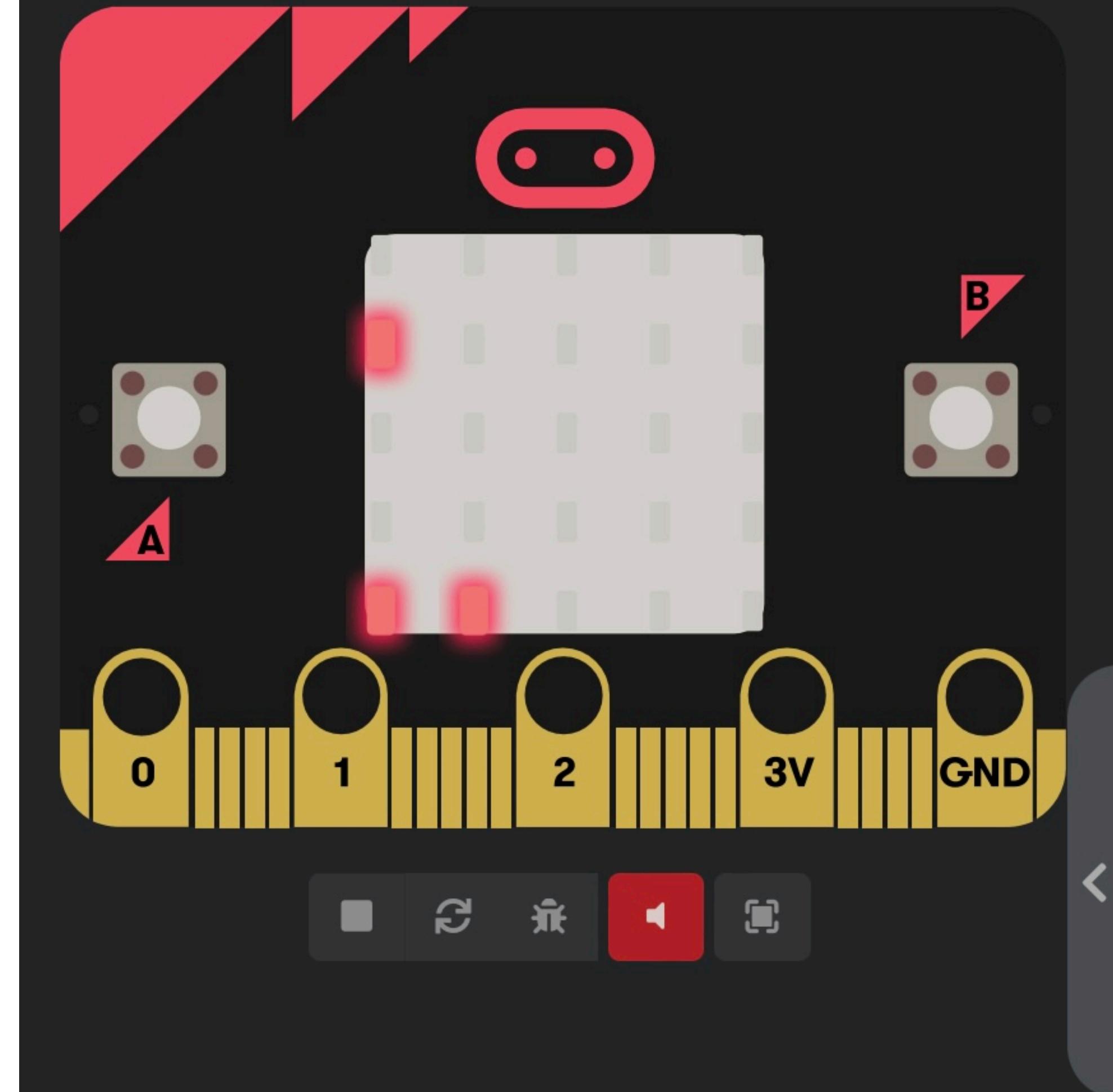


micro:bit

DEMO SINGLE PLAYER



micro:bit



Score: 3

Score: 3

python.py — terminal multiplexer

```
python.py U X  pong.js U

Users > enrico > Library > CloudStorage > OneDrive-unige.it > icdd > polyglot-pong > singleplayer > python.py
1  bar_x = 0
2  score = 0
3  interval = 0
4  interval_step = 0
5  ball_x = 0
6  ball_y = 0
7  ball_dx = 0
8  ball_dy = 0
9  in_game = False
10
11 def on_button_pressed_a():
12     global bar_x
13     if bar_x > 0:
14         led.unplot(bar_x + 1, 4)
15         bar_x = bar_x - 1
16         led.plot(bar_x, 4)
17     input.on_button_pressed(Button.A, on_button_pressed_a)
18
19 def on_button_pressed_b():
20     global bar_x
21     if bar_x < 3:
22         led.unplot(bar_x, 4)
23         bar_x = bar_x + 1
24         led.plot(bar_x + 1, 4)
25     input.on_button_pressed(Button.B, on_button_pressed_b)
26
27 def on_forever():
28     global score, interval, interval_step, ball_x, ball_y, ball_dx, ball_dy, bar_x, in_game
29     score = 0
30     interval = 500
31     interval_step = 10
32     ball_x = 3
33     ball_y = 4
34     ball_dx = -1
35     ball_dy = -1
36     bar_x = 0
37     basic.show_string("GO")
38     led.plot(ball_x, ball_y)
39     led.plot(bar_x, 4)
40     led.plot(bar_x + 1, 4)
41     in_game = True
42     while in_game:
43         if ball_x + ball_dx > 4:
```



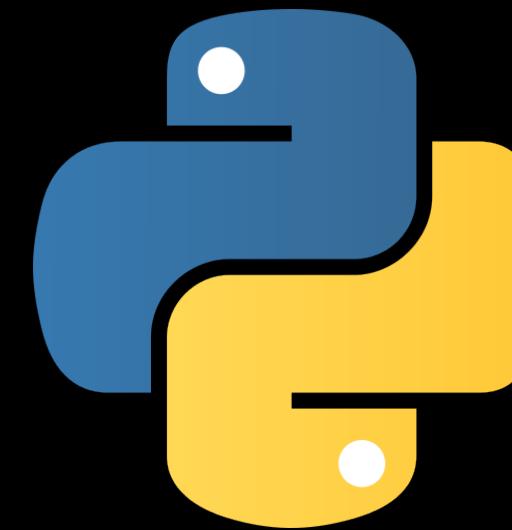
pong.js — terminal multiplexer

```
pong.js U X  python.py U

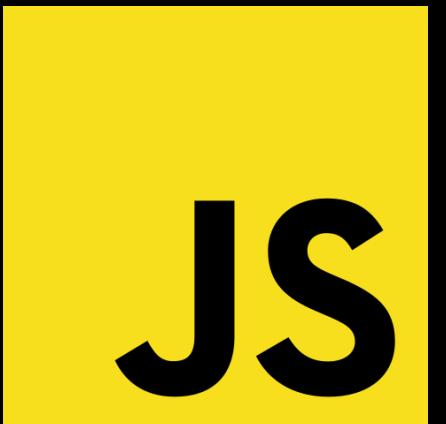
Users > enrico > Library > CloudStorage > OneDrive-unige.it > icdd > polyglot-pong > singleplayer > JS pong.js > ...
1  let bar_x = 0
2  let score = 0
3  let interval = 0
4  let interval_step = 0
5  let ball_x = 0
6  let ball_y = 0
7  let ball_dx = 0
8  let ball_dy = 0
9  let in_game = false
10
11 input.onButtonPressed(Button.A, function () {
12     if (bar_x > 0) {
13         led.unplot(bar_x + 1, 4)
14         bar_x = bar_x - 1
15         led.plot(bar_x, 4)
16     }
17 })
18
19 input.onButtonPressed(Button.B, function () {
20     if (bar_x < 3) {
21         led.unplot(bar_x, 4)
22         bar_x = bar_x + 1
23         led.plot(bar_x + 1, 4)
24     }
25 })
26 basic.forever(function () {
27     score = 0
28     interval = 500
29     interval_step = 10
30     ball_x = 3
31     ball_y = 4
32     ball_dx = -1
33     ball_dy = -1
34     bar_x = 0
35     basic.showString("GO")
36     led.plot(ball_x, ball_y)
37     led.plot(bar_x, 4)
38     led.plot(bar_x + 1, 4)
39     in_game = true
40     while (in_game) {
41         if (ball_x + ball_dx > 4) {
42             ball_dx = ball_dx * -1
43         } else if (ball_x + ball_dx < 0) {
```

JS

- Interpretato (Python VM)
- Sviluppo back-end
- No uso dei “;”, ma...
- ...indentazione per definire i code-block
- nomeVar = 7
- none
- Memoria delle variabili virtualmente infinita

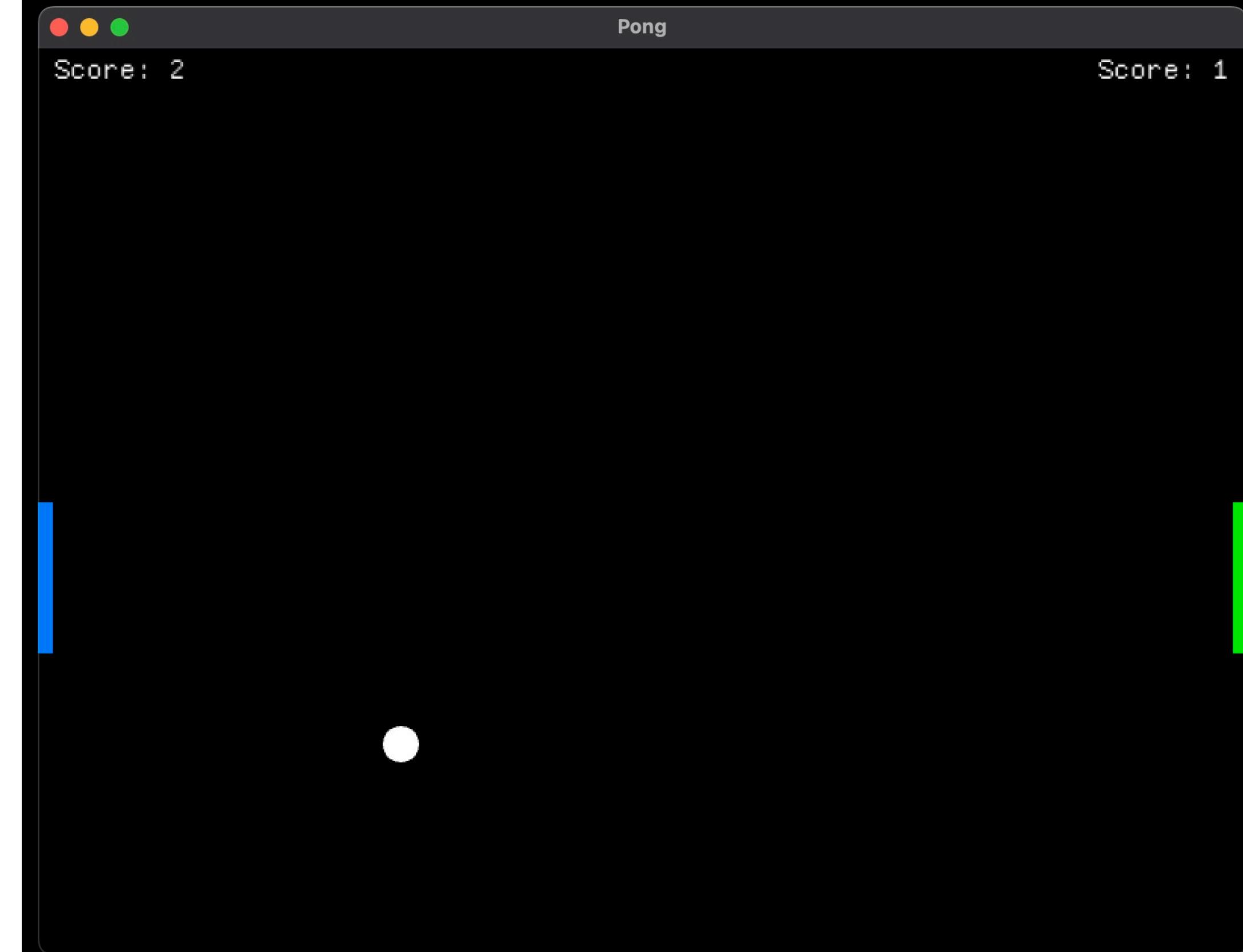
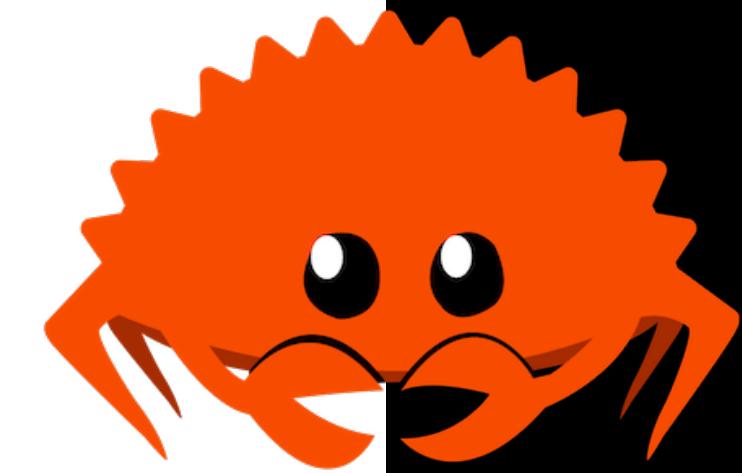


- Interpretato (nel browser)
- Sviluppo back-end E front-end
- Sì uso dei “;” e ...
- ...parentesi {} per definire i code-block
- var nomeVar = 7;
- null
- Variabili allocate automaticamente



(De)Allocazione memoria automatica

DEMO MULTI PLAYER



Score: 3

Score: 3



score.rs — terminal

```
score.cpp U score.rs X
pong > src > score.rs
1 use macroquad::prelude::*;
2
3 pub struct Score {
4     pub player_1: u32,
5     pub player_2: u32,
6 }
7
8 impl Score {
9
10     pub fn new() -> Score {
11         Score {
12             player_1: 0,
13             player_2: 0,
14         }
15     }
16
17     pub fn increment_score(&mut self, player: u32) {
18         if player == 1 {
19             self.player_1 += 1;
20         } else {
21             self.player_2 += 1;
22         }
23     }
24
25     pub fn draw(&self) {
26         draw_text(&format!("Score: {}", self.player_1), 10.0, 20.0, 25.0, WHITE);
27         draw_text(&format!("Score: {}", self.player_2), 700.0, 20.0, 25.0, WHITE);
28     }
29
30 }
```

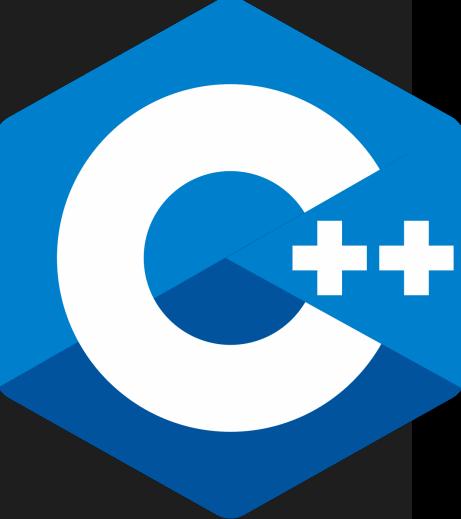
main 0 △ 0 Live Share Git Graph



score.cpp — terminal multip

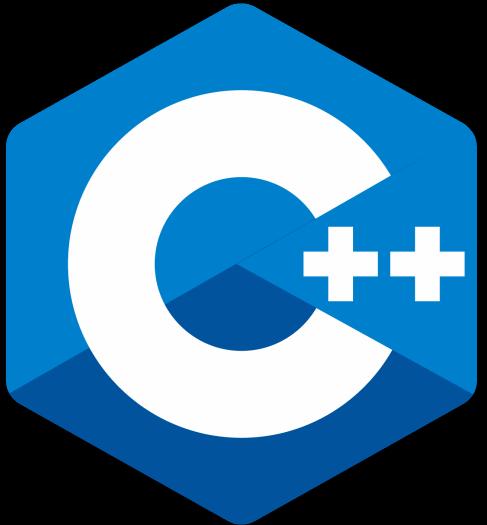
```
score.cpp U score.rs X
Users > enrico > Library > CloudStorage > OneDrive-unige.it > icdd > polyglot-pong > singleplayer > score.cpp
1 #include <iostream>
2
3 struct Score {
4     unsigned int player_1;
5     unsigned int player_2;
6
7     Score() : player_1(0), player_2(0) {}
8
9     void increment_score(unsigned int player) {
10         if (player == 1) {
11             player_1++;
12         } else {
13             player_2++;
14         }
15     }
16
17     void draw() {
18         std::cout << "Score: " << player_1 << std::endl;
19         std::cout << "Score: " << player_2 << std::endl;
20     }
21 }
22
```

main* 0 △ 0 Live Share Git Graph





- È più sicuro a livello di memoria (safe mode)
- Ownership (Proprietà)
- Strumenti sta per creare pacchetti (Cargo)
- Compila leggermente più lento
- Type checking statico o dinamico
- Espressioni di tipo esplicite o implicite
- `let i: i8 = 1;`
- Meno sicuro, gestione manuale
- Dangling Pointers
- Nessun strumento standard
- Compila leggermente più veloce
- Type checking solo statico
- Espressioni di tipo solo esplicite
- `int i = 1;`



Entrambi compilano in linguaggio macchina



**CODING IS TODAY'S LANGUAGE OF
CREATIVITY. ALL OUR CHILDREN
DESERVE A CHANCE TO BECOME
CREATORS INSTEAD OF CONSUMERS OF
COMPUTER SCIENCE.**

MARIA KLAWE, PRESIDENT OF HARVEY MUDD COLLEGE

TAKE-HOME MESSAGE



TROPPI LINGUAGGI?

- Ognuno ha le sue differenze (motivazioni storiche, memoria e garbage-collector)
- A seconda del contesto si sceglie il più adatto (tempi di compilazione, framework, etc)
- Difficoltà intrinseca
- Portabilità, cross-platform
- Didattica

