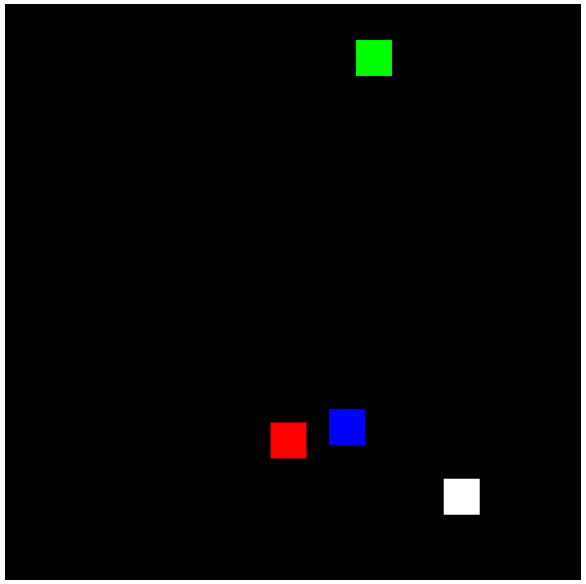
Project 2

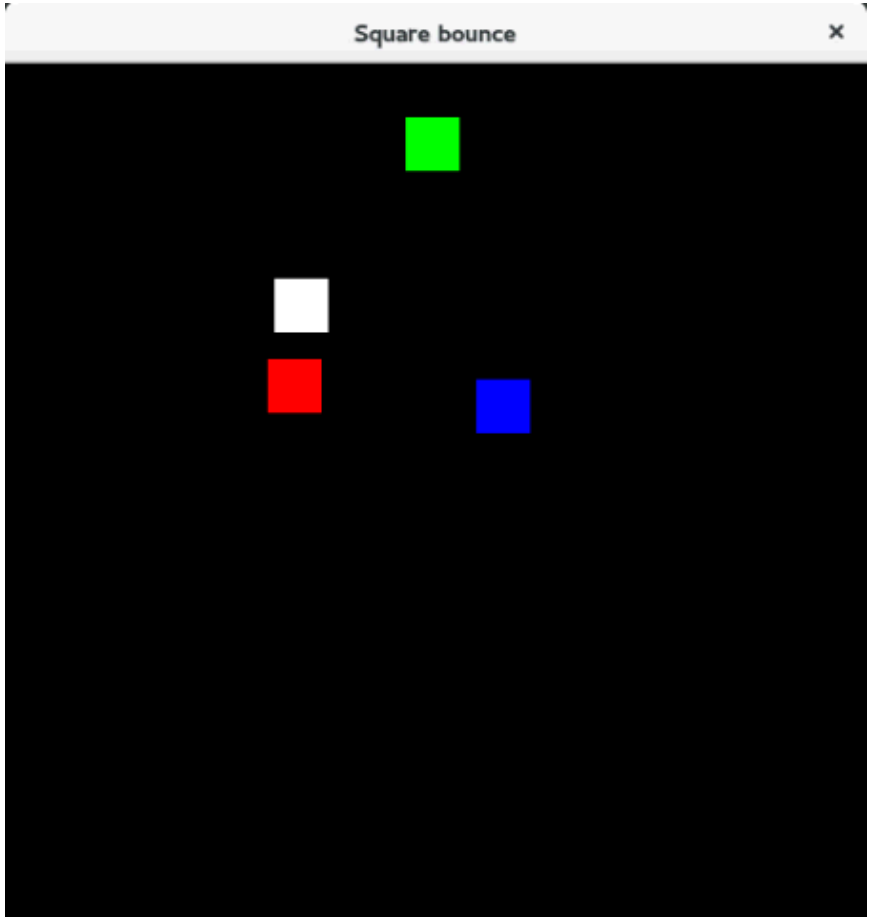
Parallel programming



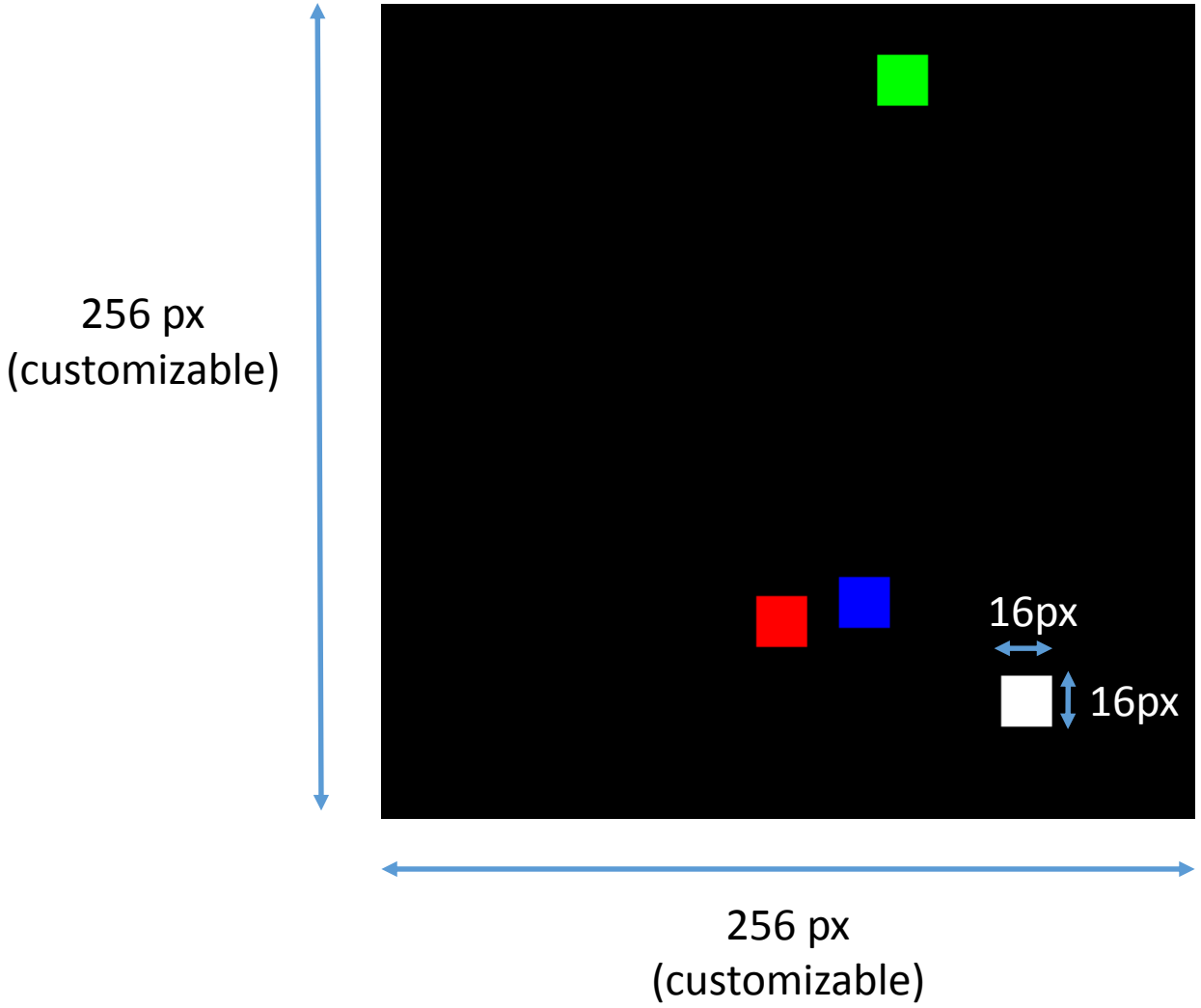
To be done by teams of **two people**

Deadline : December 18, 2017, 23:59

Bouncy squares



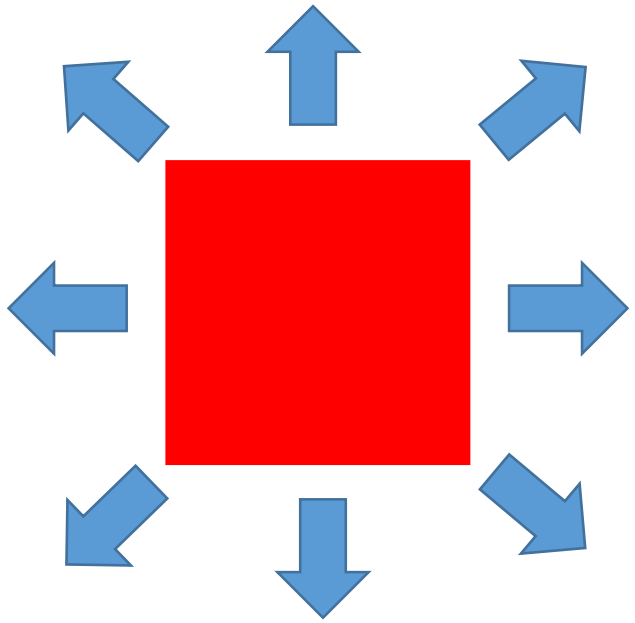
Dimensions



Speed

Velocity on x-axis $\in \{-1,0,1\}$

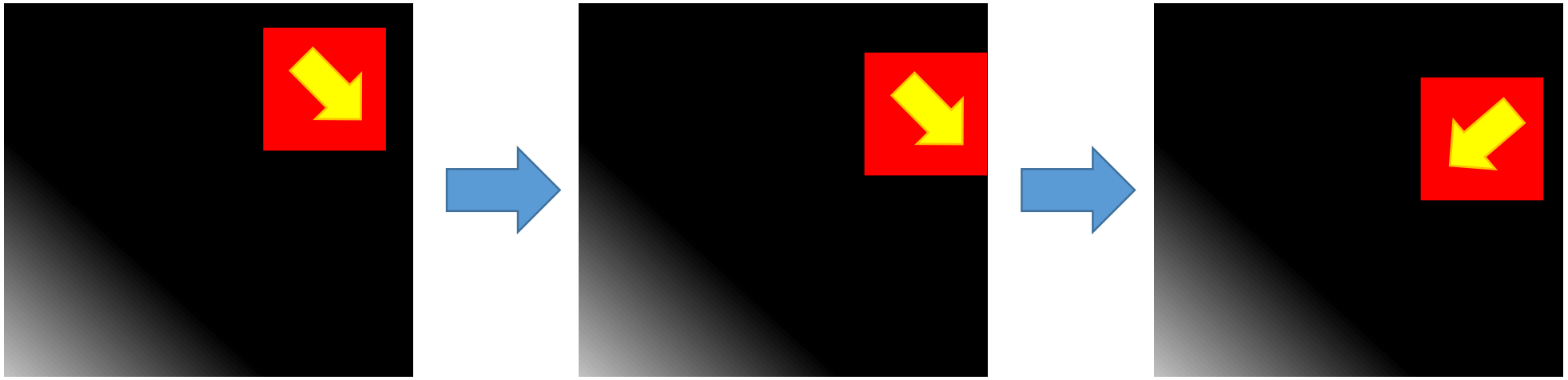
Velocity on y-axis $\in \{-1,0,1\}$



A square can move towards 8 directions (or stay in the same spot)

Collisions/Out of bounds

- Out of bounds
  - Change the direction that lead to crossing the boundary



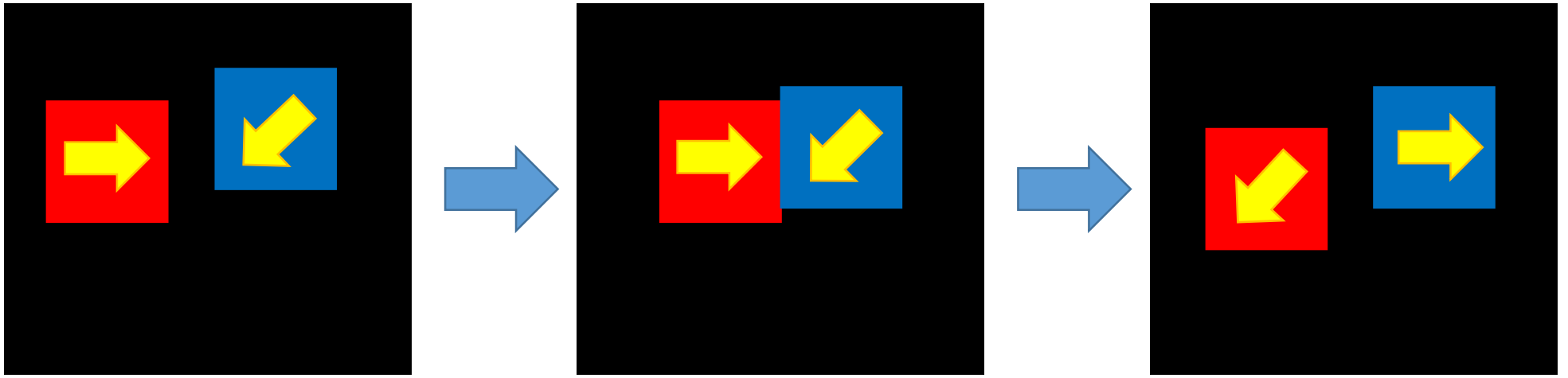
X-velocity = 1
Y-velocity = 1

X-velocity = 1
Y-velocity = 1

X-velocity = -1
Y-velocity = 1

Collisions/Out of bounds (2)

- Collisions
  - Swap the velocities (special rule for more than 2 squares)



X-velocity = 1
Y-velocity = 0

X-velocity = -1
Y-velocity = 1

X-velocity = 1
Y-velocity = 0

X-velocity = -1
Y-velocity = 1

X-velocity = -1
Y-velocity = 1

X-velocity = 1
Y-velocity = 0

Parallel Processing

- Single-process program provided
- Each square = 1 process (workers)
- Master process for input/output
- Position of squares in shared memory
- Velicities NOT in shared memory
- Shared memory protected by semaphores (if needed)
- Communication between workers and master process by semaphores and shared memory
- Communication between workers by message queues

Parallel processing (2)

Worker N :

- Move square N
- See if out of bounds or collision
- If so, find new direction (possibly by interacting with another process)
- Warn Master process that movement is done
- Wait for Master process to display

Master process:

- Wait for all workers
- Display new positions
- Warn Workers that display is done
- If user pressed <ENTER>, quit program **properly**

Coding Guidelines

- Focus on code clarity and understandability before efficiency
- Still, your code shouldn’t be unreasonably inefficient (tip: use as few IPCs as possible, avoid repeating useless operations)
- Document your code !!
- Functions should be documented:
  - Parameters
  - Operations performed

Files and submission

You are provided with:

- **Bounce.zip** : a zip archive containing C implementation of the bouncy squares program using SDL displays. You can use it as basis for your implementation.

You must submit **in a ZIP file named « sXXXXXX_NAME1_sYYYYYY_NAME2.zip »**:

- The source code of your program (.c, .h, makefile)
- **report.pdf** : Describe using simplified C syntax how you implemented the synchronization between processes.

Submitting other files will be sanctioned