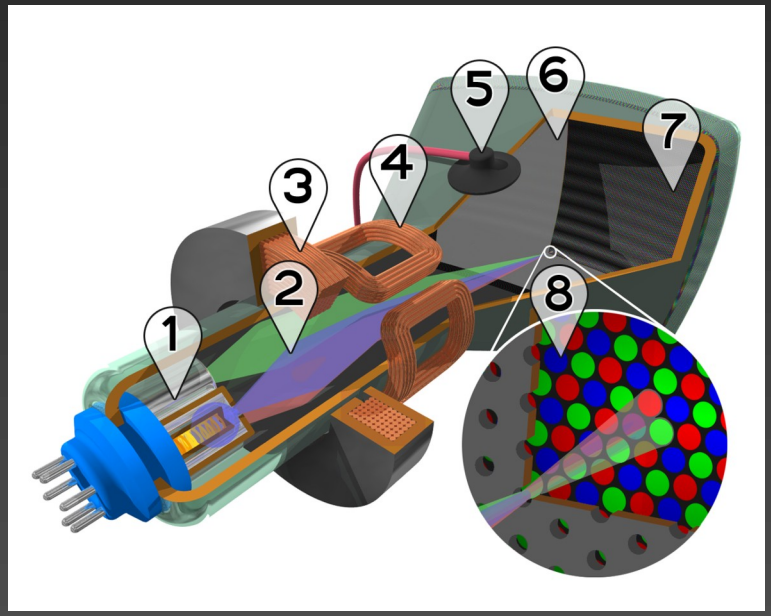
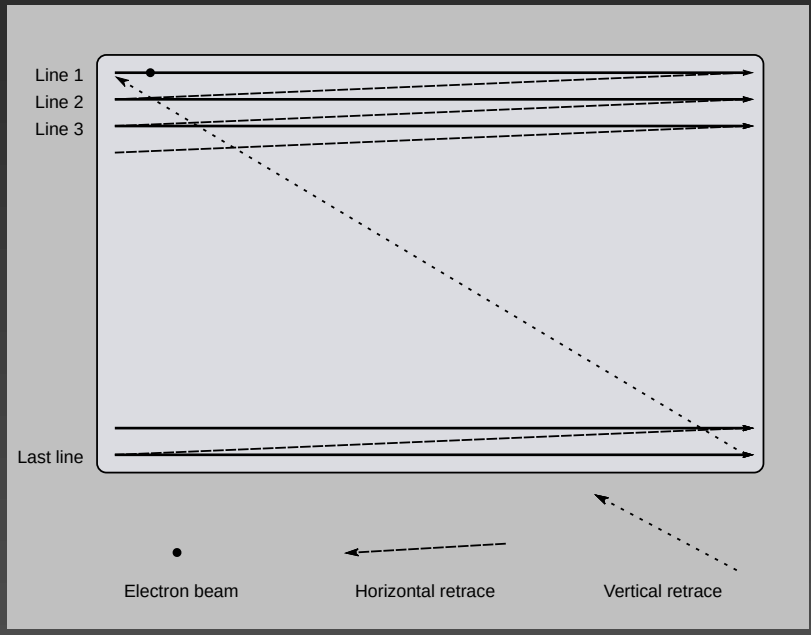
Display synchronization

Vsync,
double-buffering, triple-buffering

Changes in picture on V-sync

- V-sync is (was) signal for vertical retrace of electron beam to upper left corner
  - Retrace is the best time for changing displayed picture
  - without Vsync is old picture in top part and new picture in bottom part of screen – tearing
  - faster GPU → more tearing
- Constant RGB data rate = pixel clock





Tearing

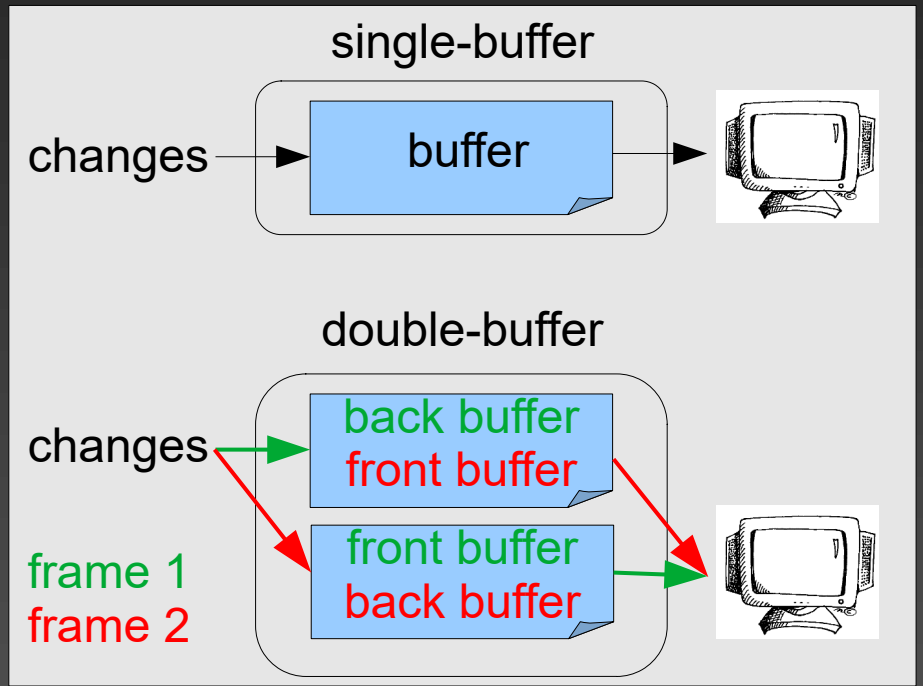
V-Sync in GLFW

- in init() or during runtime

```
// Wait for ‘n‘ V-Sync events before swapping color buffers
glfwSwapInterval(n);


//------------------------------------------------------------
// COMMON USE

// Do not wait for V-Sync = Set V-Sync OFF
glfwSwapInterval(0);

// Wait for ONE V-Sync = Set V-Sync ON
glfwSwapInterval(1);




//------------------------------------------------------------
// VERY UNCOMMON USE

// Wait for TWO V-Sync events = show result every other display refresh = 30 FPS limit
glfwSwapInterval(2);
```

Single vs. Double-buffering

- Drawing frame takes time
  - visible flicking caused by sequention of triangles etc.; we can see build of the scene
- Two color-buffers, front and back
  - drawing changes to back buffer, displaying from front buffer
  - swap buffers (HW accelerated)
  - without V-sync: tearing – horizontal picture tear



Init: GLFW creates double-buffer by default

Frame finished:

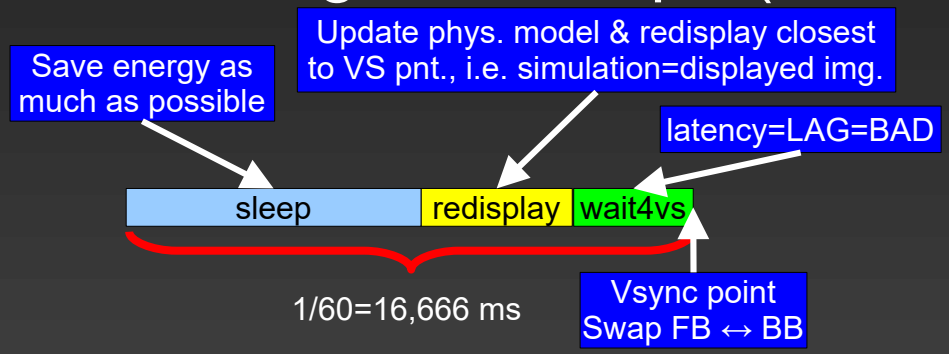
```
glfwSwapBuffers(window)
```

(contains glFlush();)

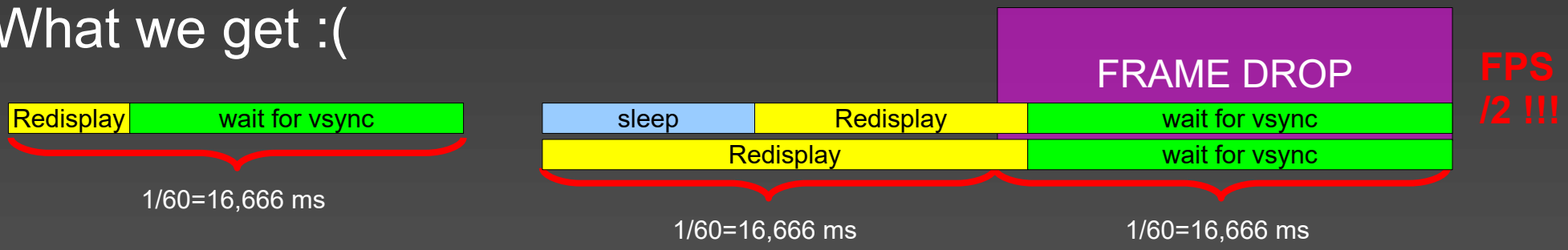
Frame drop

- min. 24 (16) fps necessary, double-buffering, vsync
- Infinite loop: update simulation → redraw
- redraw usually as fast as possible (no explicit wait)
  - 100% system load
  - can you use Sleep(milisec) etc.? How many ms?
    - using Sleep() blocks thread, CPU(GPU) time wasted, if we wait too long, FPS drop :-(

What we want:



What we get :(



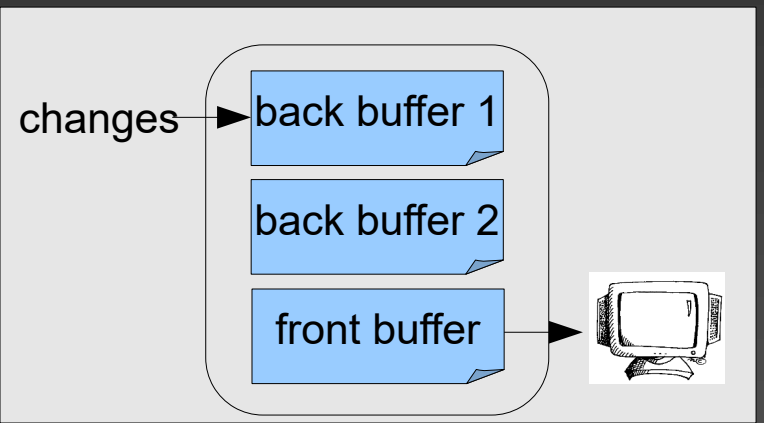
Double vs. Triple-buffering

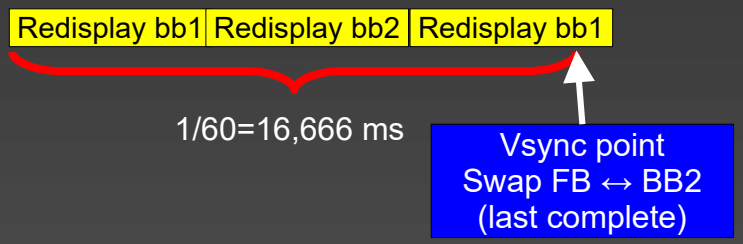
- Double-buffer disadvantage
  - significant FPS drop
    - 1/1, 1/2, 1/3, 1/4, … → 60, 30, 20, 15, …
  - higher latencies
- Triple buffering in GL – add third buffer
  - draw new frame to the second back buffer immediately after previous is finished

  \+ increase speed, lowers latency

  \- increase GPU+MEM load

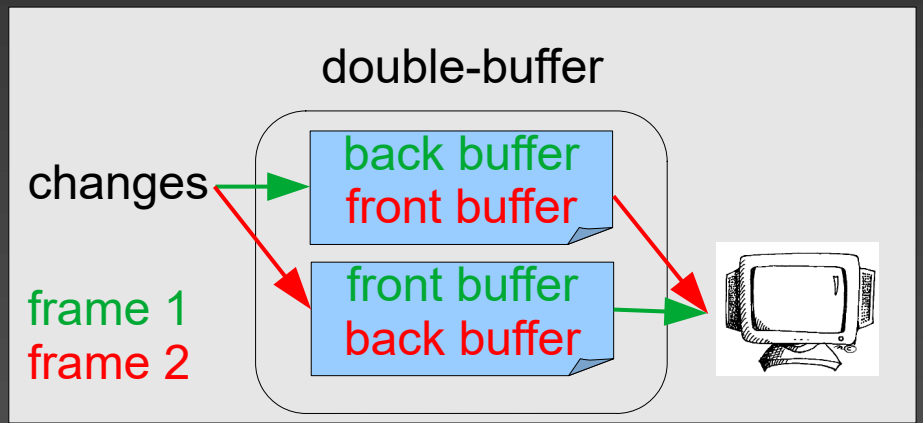
  (D3D: serial buffers – adds latency even more)
- Vulkan = mailbox; nvidia = FastSync

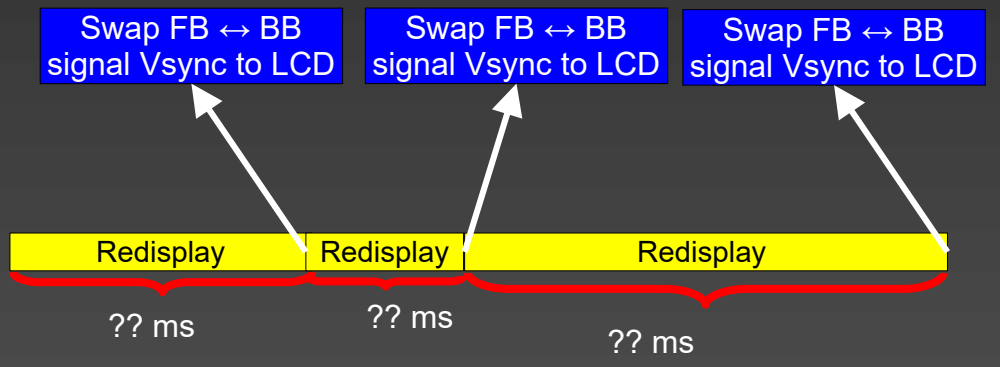




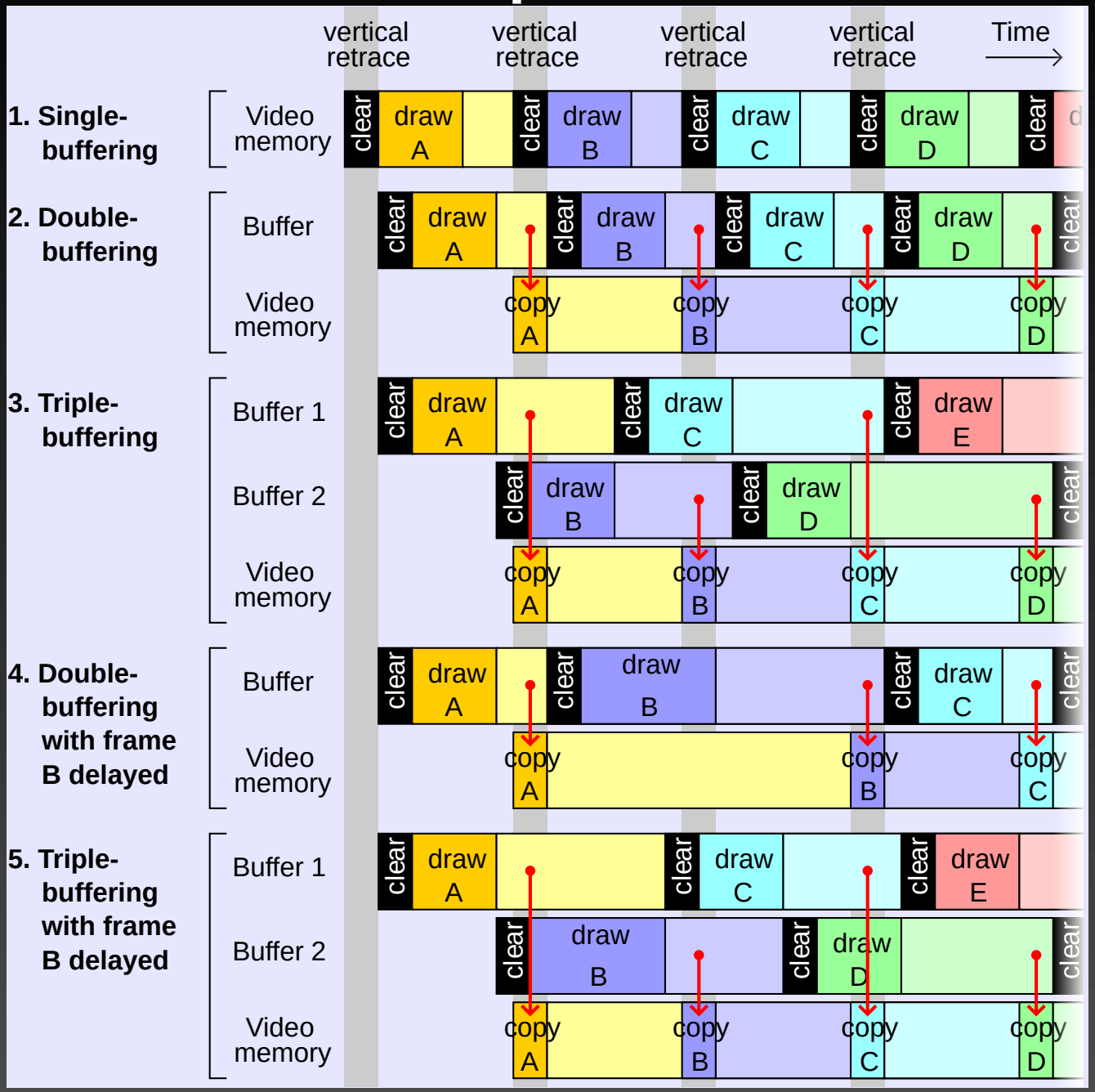
The best solution

- VRR Variable refresh rate (+ high FPS monitor, 120, 144, 160, …)
  - Redisplay image + swap buffers when ready
    - LCD waits for vsync… it does NOT start new refresh itself
  - Send vsync signal to LCD
  - Send data as fast as possible (max supported LCD speed)
    → No fixed FPS, variable within technological limits, e.g. 40…160Hz
  - Use only one back buffer (like double-buffering)
  - No waiting = draw as fast as you can
  - If actual_fps < max_lcd_fps
    - vsync=on → no tearing
    - Else: vsync = off → tearing, but max possible fps
- AMD FreeSync = VESA Adaptive Sync (Displayport) = HDMI 2.1 VRR, Nvidia G-Sync, …





Comparison


vertical retrace
vertical retrace
vertical retrace
vertical retrace
Time
1. Single-buffering
Video memory
clear
draw A
draw B
draw C
draw D
2. Double-buffering
Buffer
Video memory
copy A
copy B
copy C
copy D
draw E
3. Triple-buffering
Buffer 1
Buffer 2
4. Double-buffering with frame B delayed
5. Triple-buffering with frame B delayed