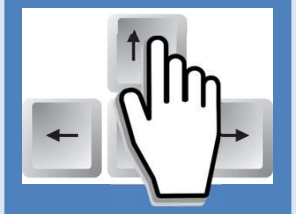
Game Loop

Benutzer Eingaben



Spiele Mechanik

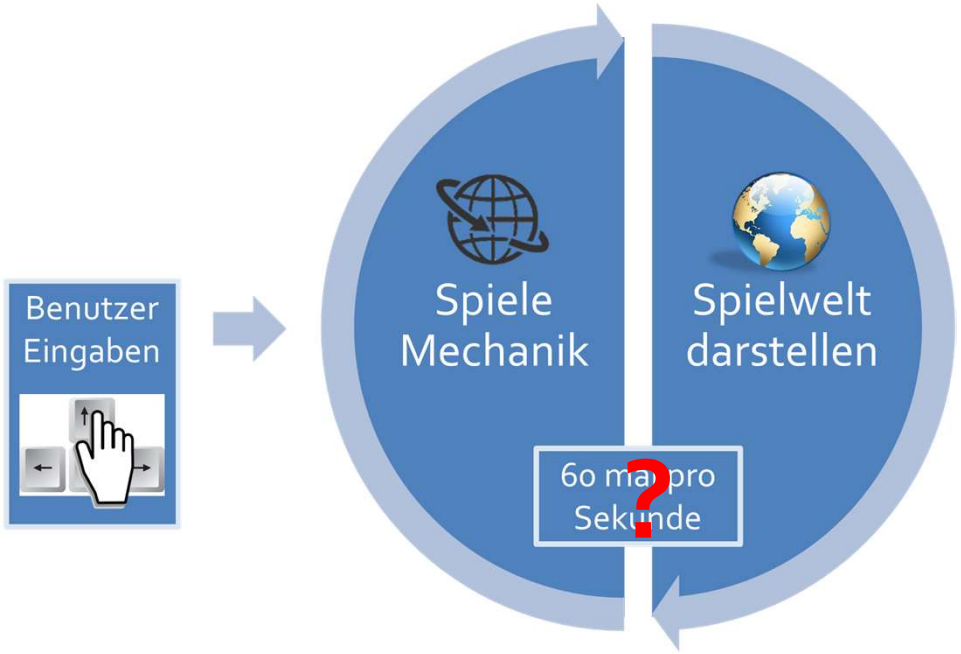
Spielwelt darstellen

60 mal pro Sekunde

Interactive Application

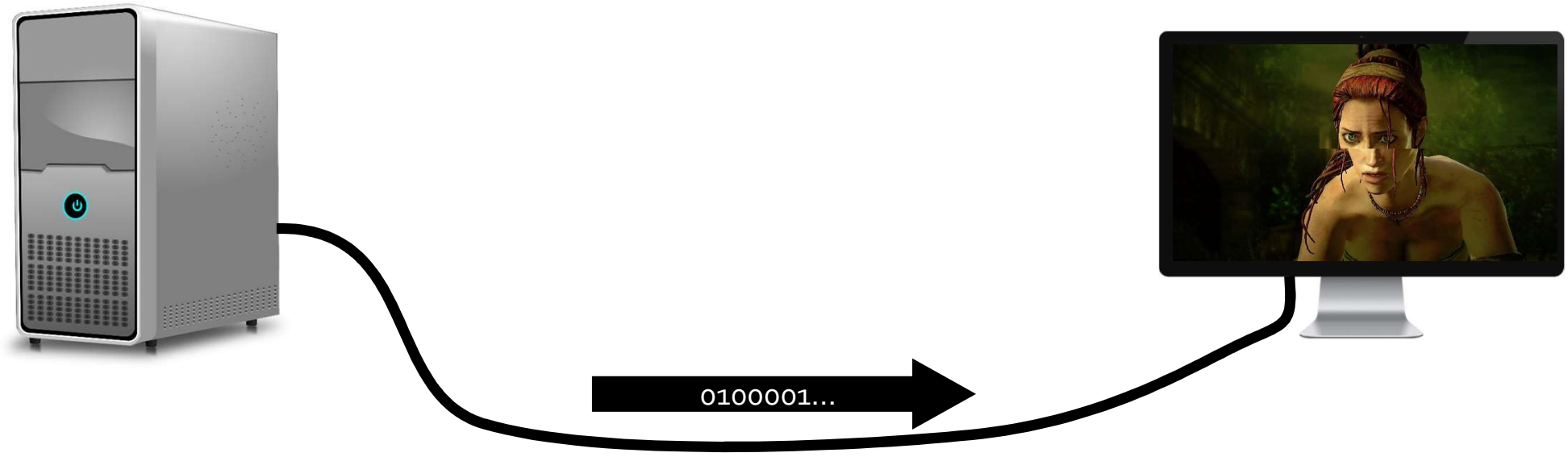
- Program must react on user input while updating state and visualization
- Loop may be executed very quickly therefore very often

```
while(not finished)
{
    input = getInputState()
    UpdateGameState(input)
    DrawGameWorld()
}
```

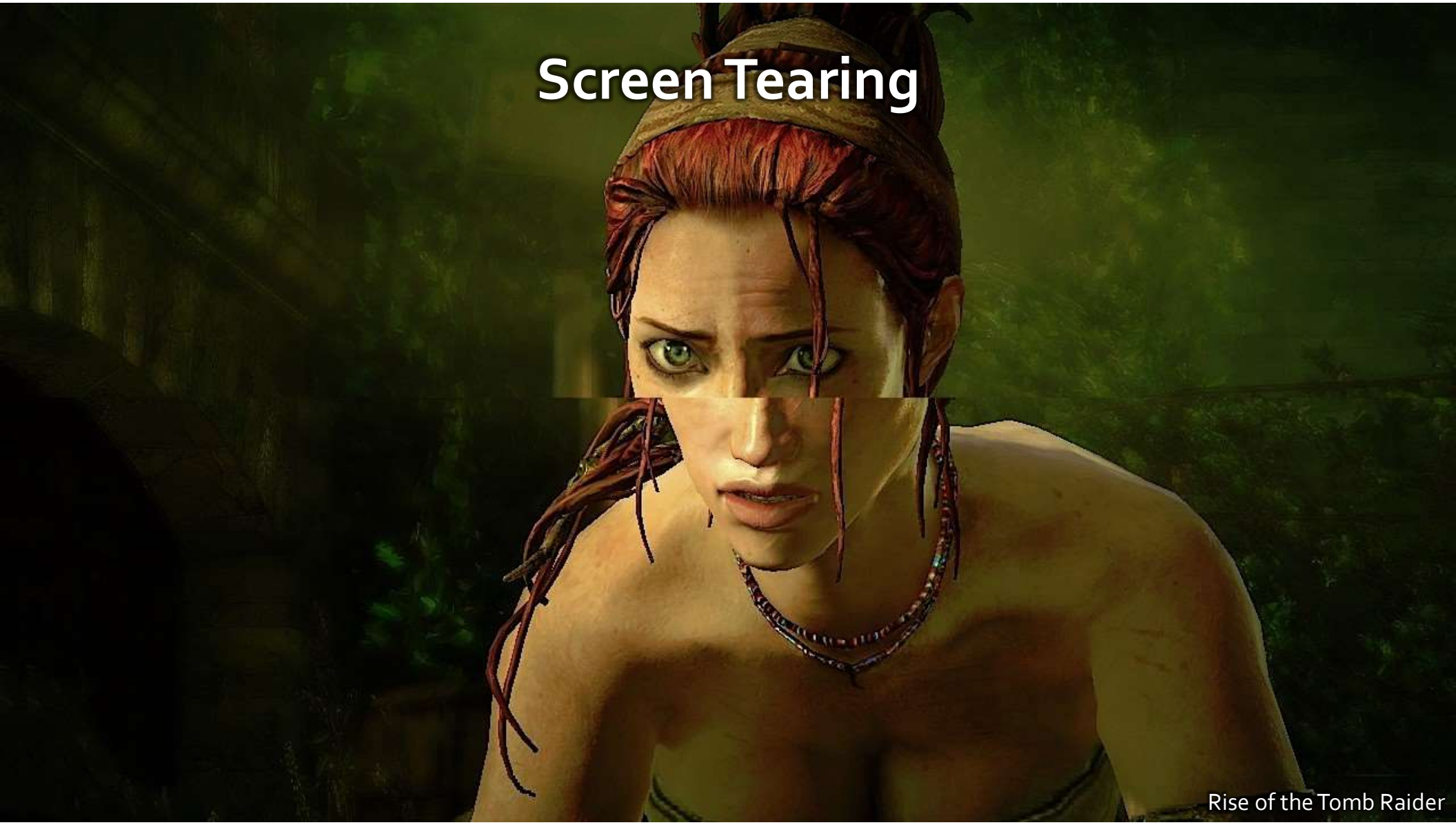


Problems with this Approach

- `DrawGameWorld()` creates an image by updating data in memory
- This data is sent from computer to monitor
- If data changes during transmission image may be inconsistent



Screen Tearing

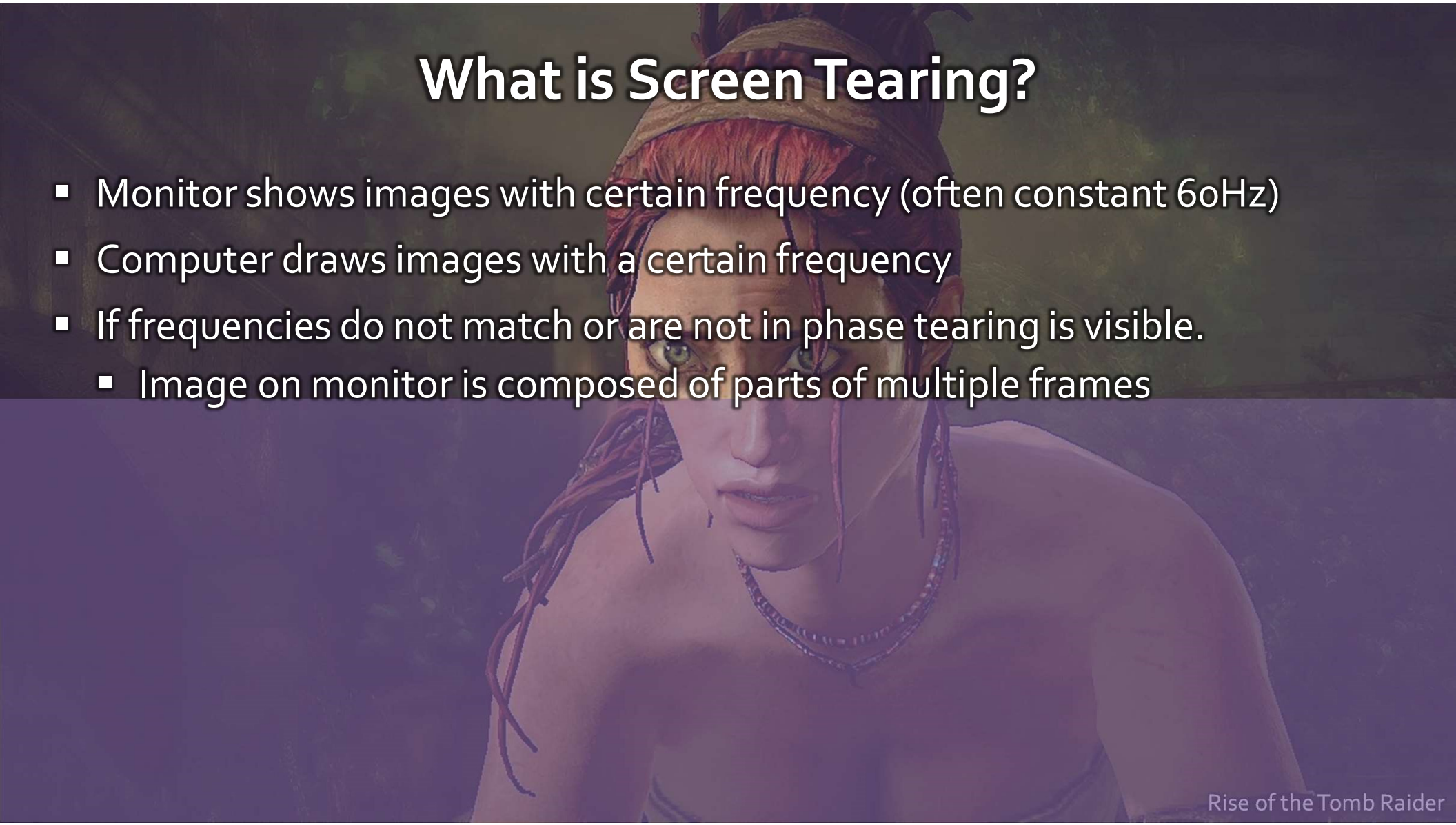


Rise of the Tomb Raider


What is Screen Tearing?
Old frame
New frame
Rise of the Tomb Raider

What is Screen Tearing?

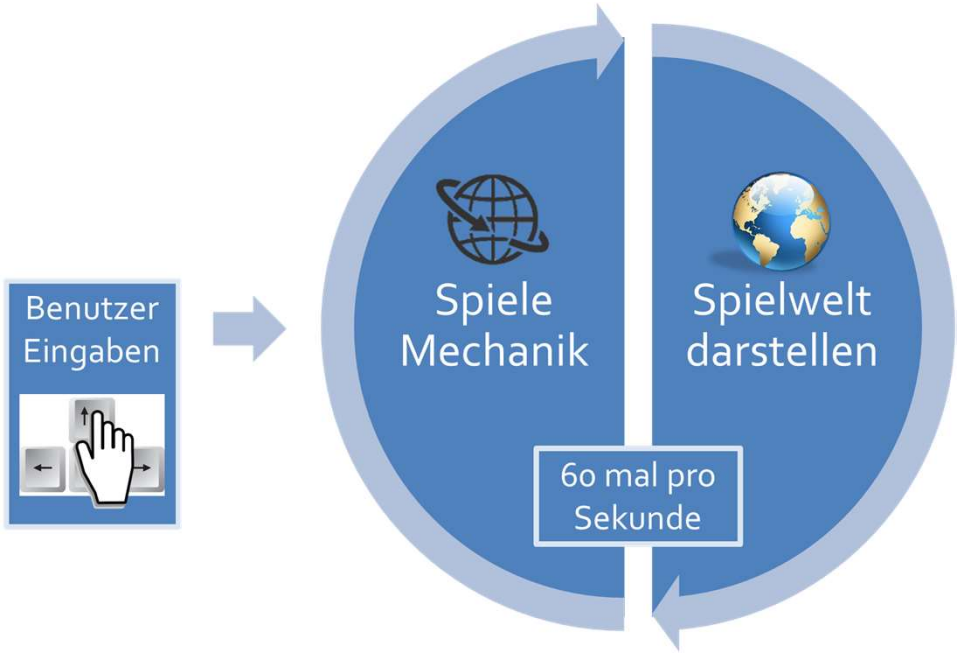
- Monitor shows images with certain frequency (often constant 60Hz)
- Computer draws images with a certain frequency
- If frequencies do not match or are not in phase tearing is visible.
  - Image on monitor is composed of parts of multiple frames



Interactive Application with Synchronization

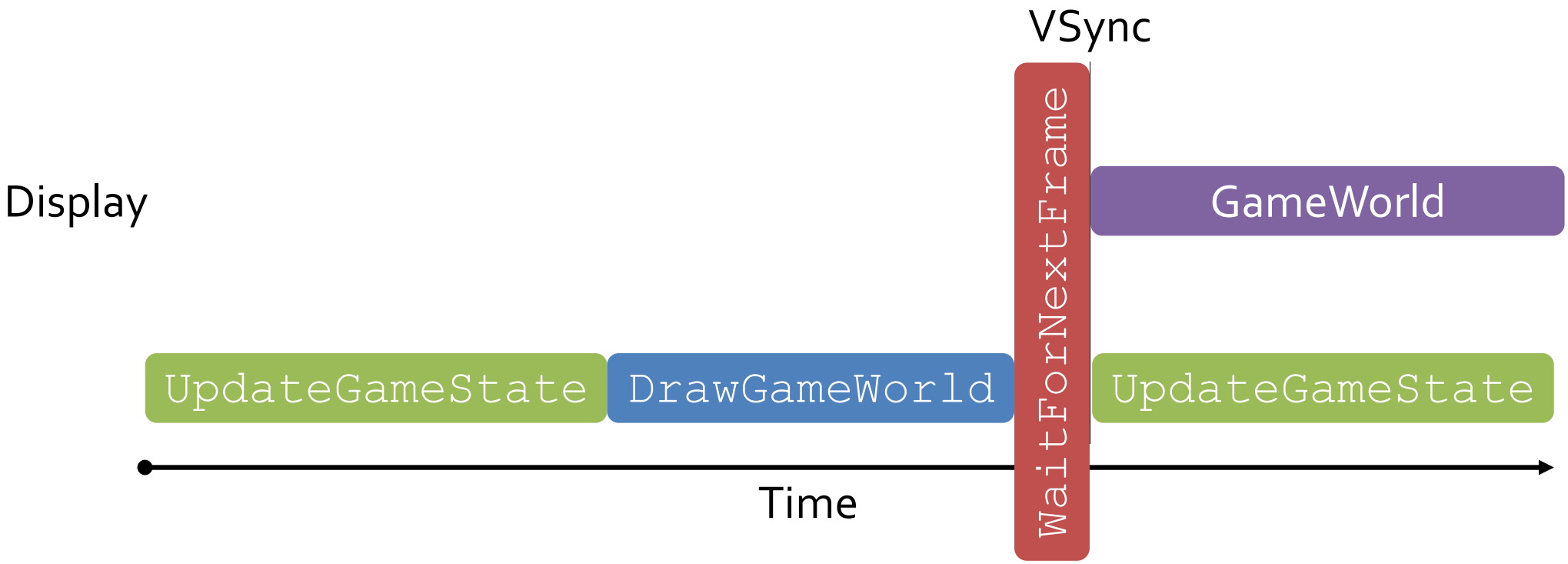
- Program must react on user input while updating state and visualization
- Loop waits for next frame (vertical synchronization)

```
while(not finished)
{
    input = getInputState()
    UpdateGameState(input)
    DrawGameWorld()
    WaitForNextFrame()
}
```

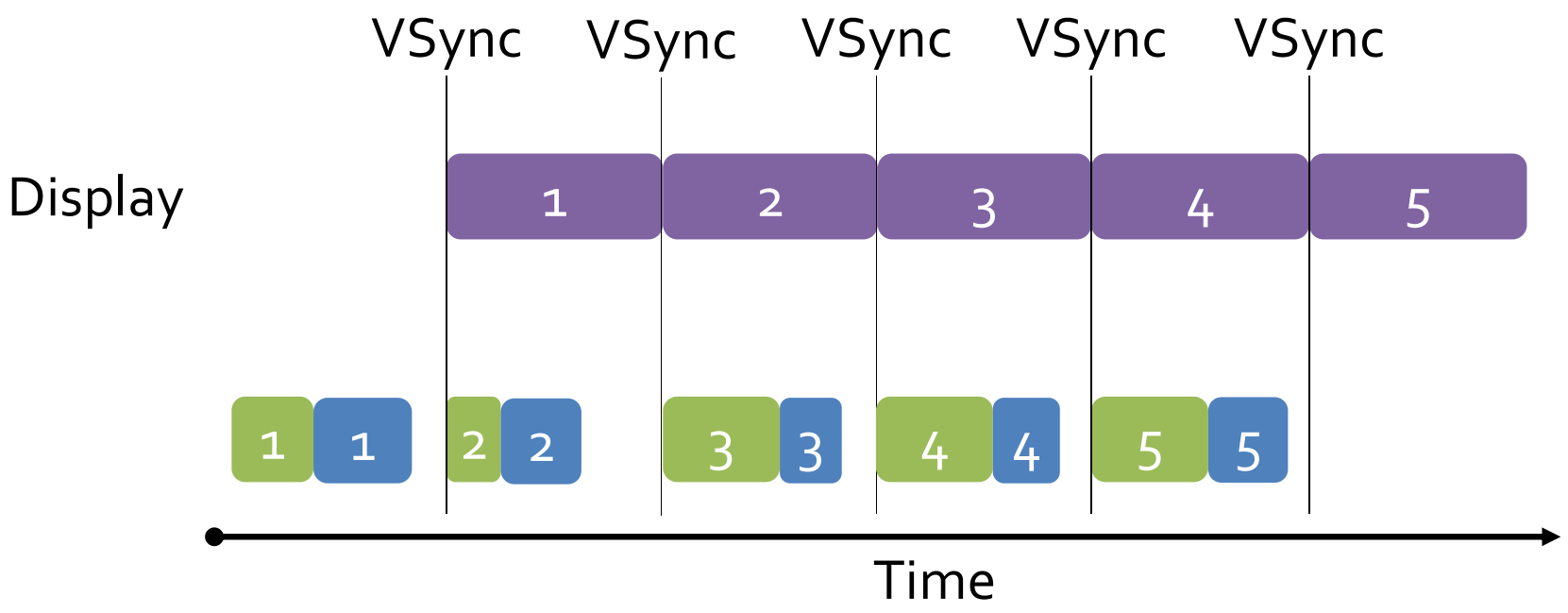


Vertical Synchronization (VSync)

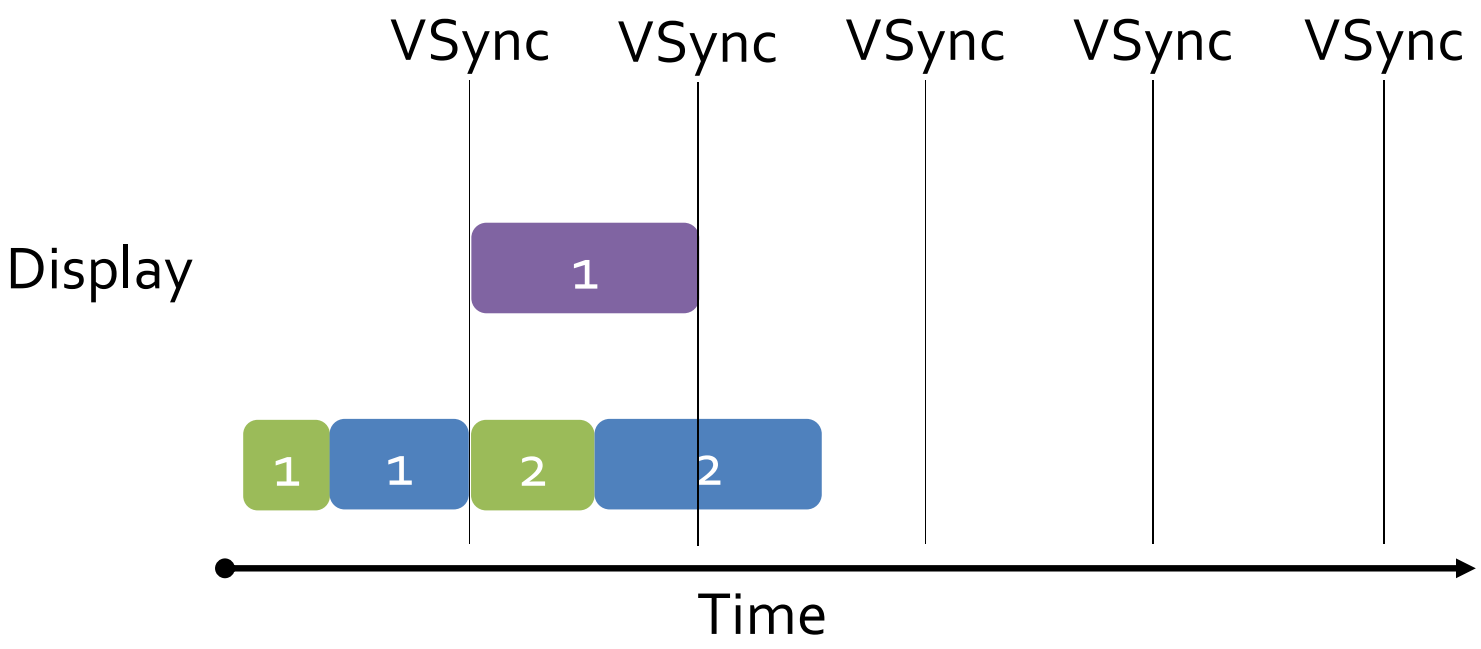
- GPU is prevented from changing display memory (frame buffer) until after monitor finishes its current refresh cycle



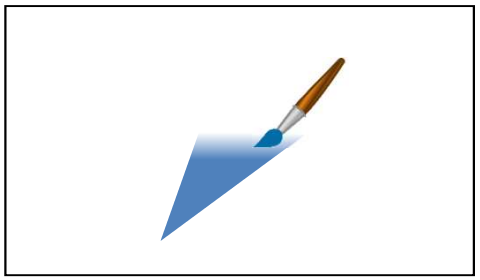
Vertical Synchronization (1 Refresh Cycle Input Lag)



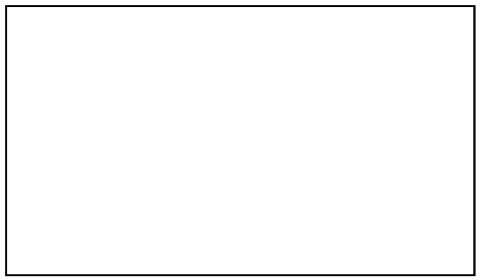
Vertical Synchronization (Slow Update)



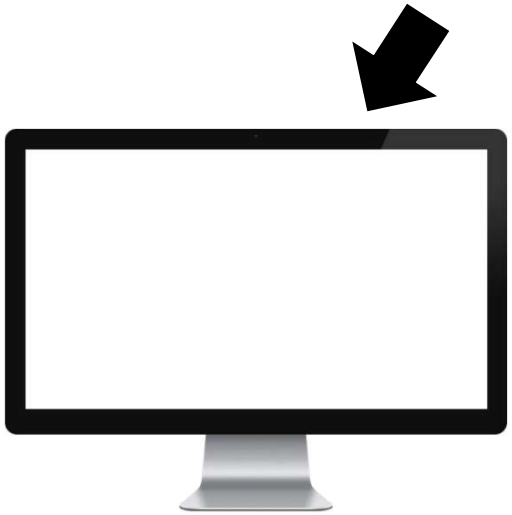
Double Buffering (2 Frame Buffer)



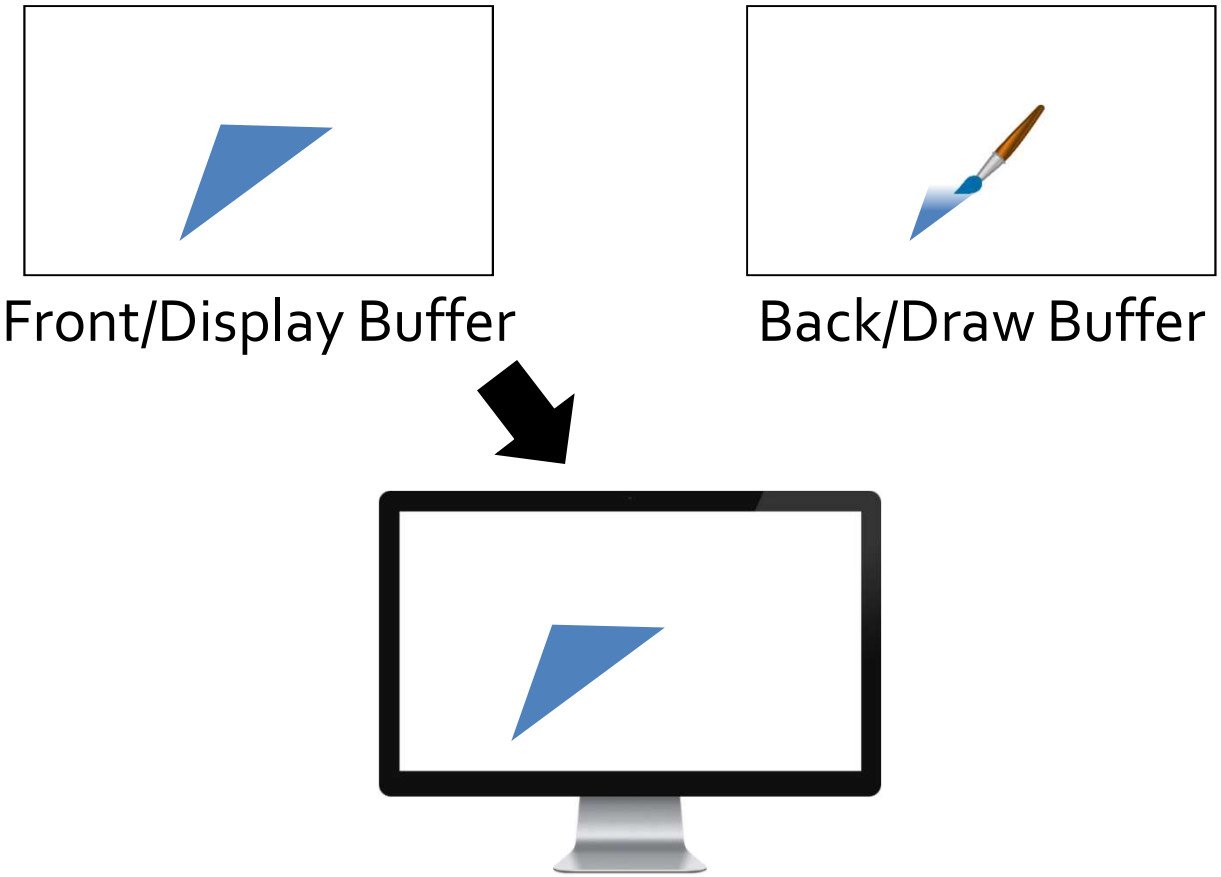
Back/Draw Buffer



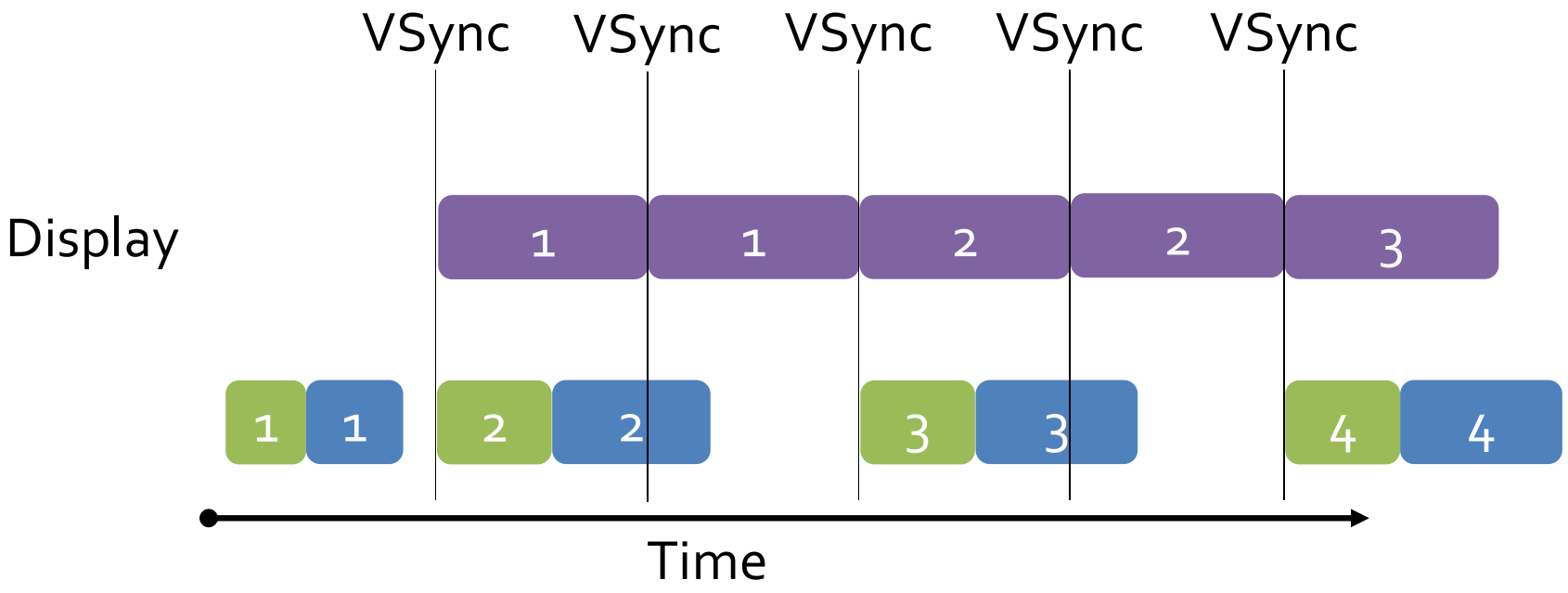
Front/Display Buffer



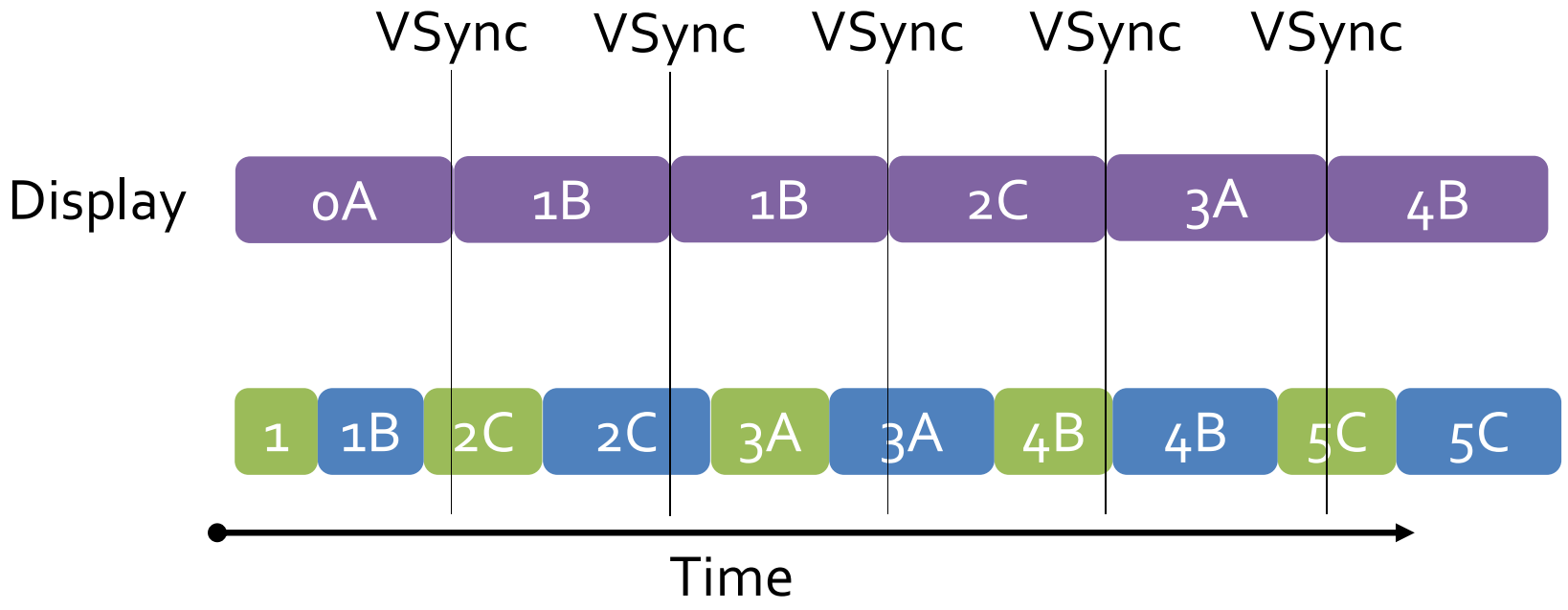
Double Buffering



Double Buffering – Stuttering



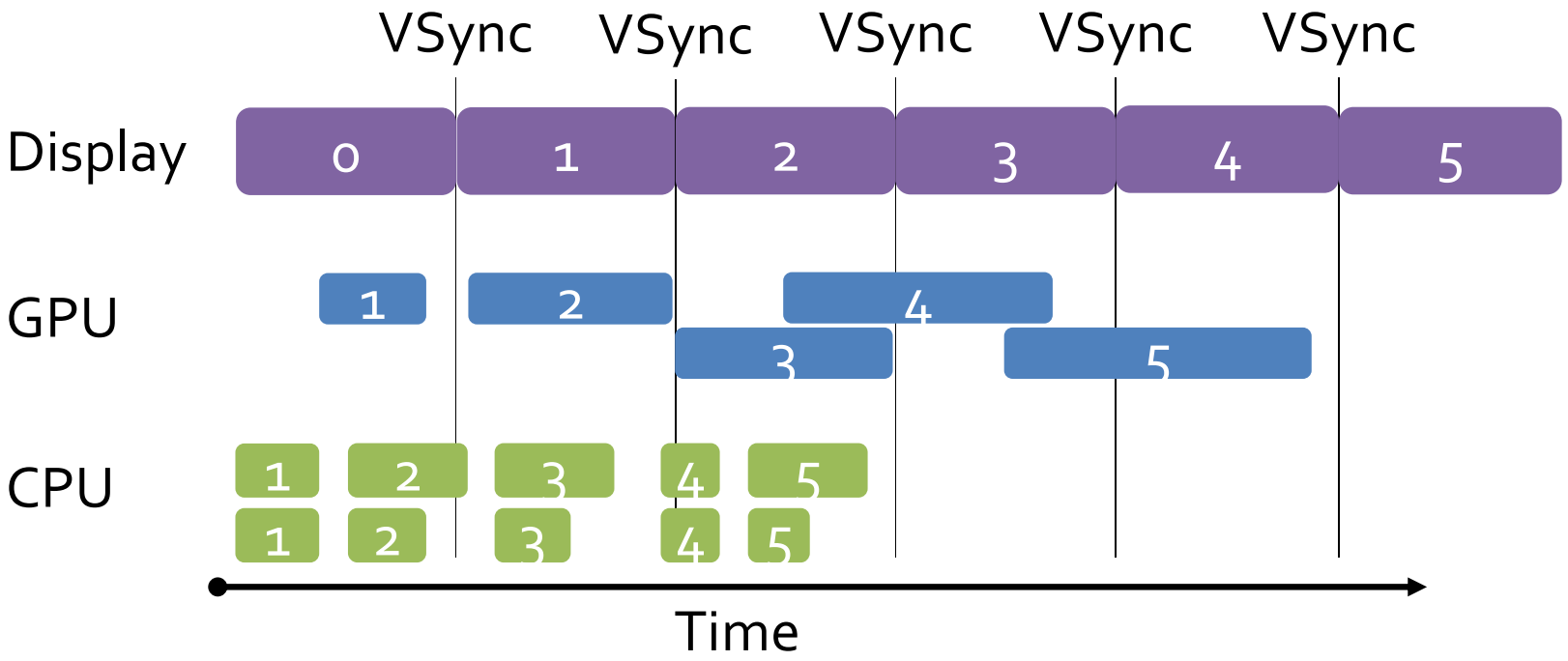
Tripple (Multi) Buffering



A,B,C,… buffers

Reality often more complex

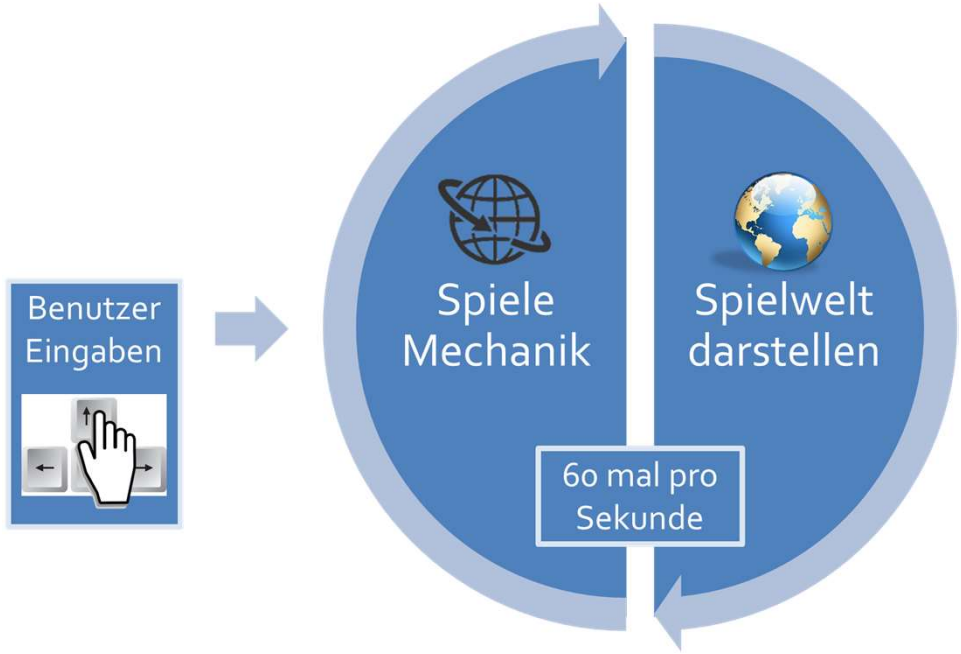
- Draw calls on hardware can be executed asynchronously after data copy
- E.x.: multi-threaded Vulcan (copy, marshal, …)



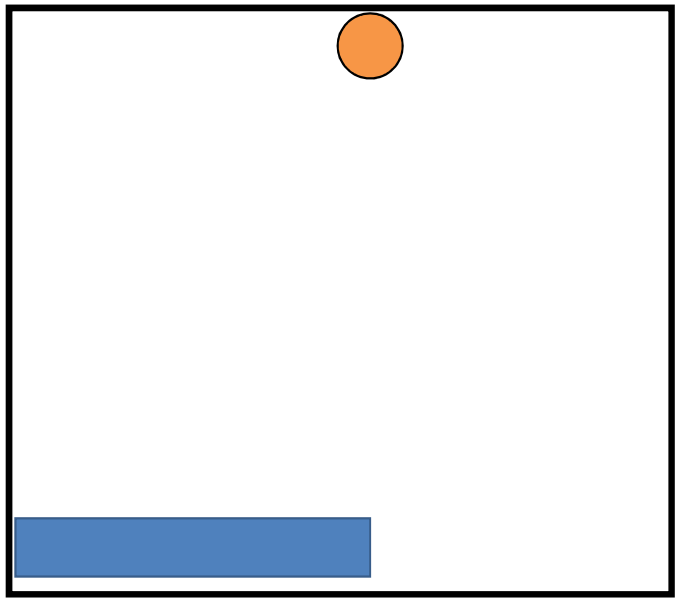
Render Loop

- Program must react on user input while updating state and visualization
- Loop waits for next frame (vertical synchronization)

```
while(not finished)
{
    input = getInputState()
    UpdateGameState(input)
    DrawGameWorld()
    WaitForNextFrame()
}
```



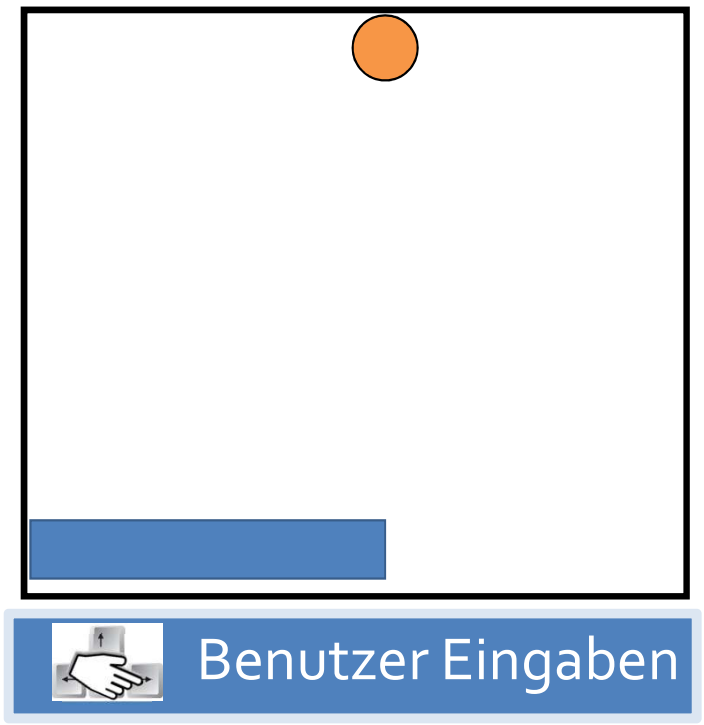
Animation – Idea?



- Move animated objects each frame a bit

```
y = 1
while(not finished)
{
    …
    y -= 0.1 //update
    DrawBall(y) //draw
    …
}
```

Animation with User Input

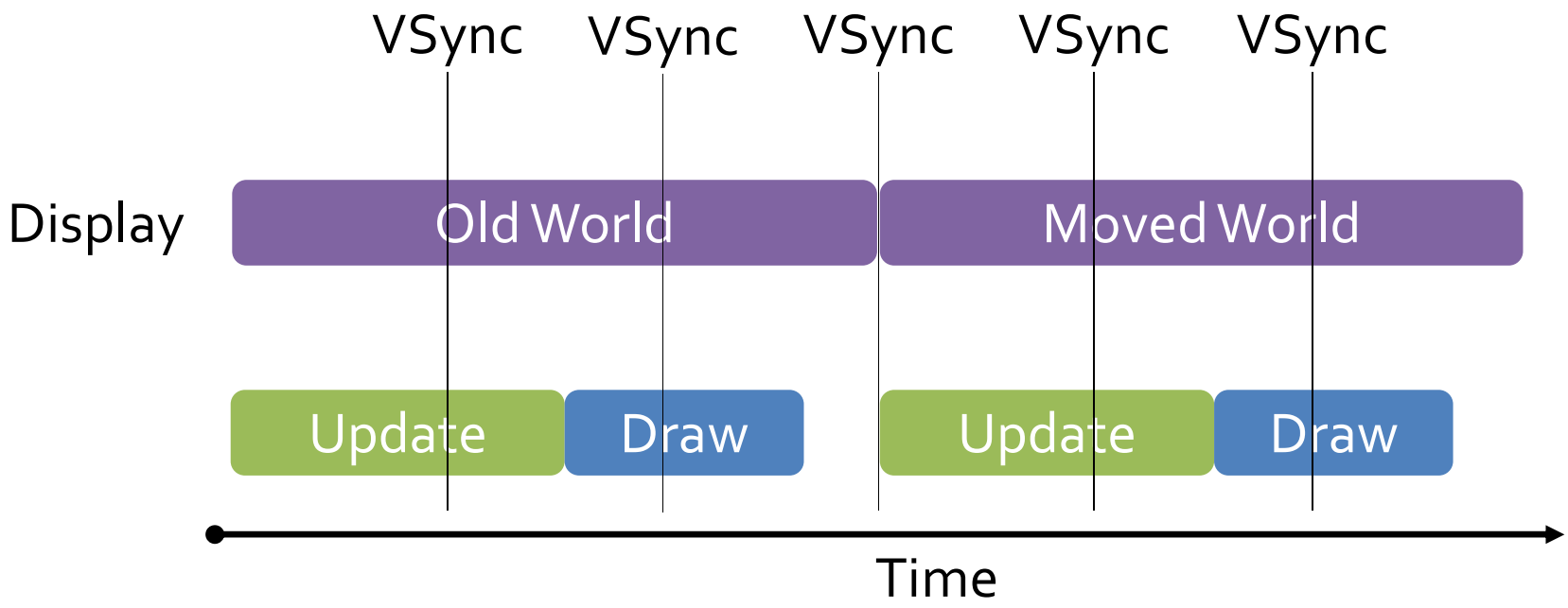


- Move objects according to input

```
while(not finished)
{
   //update
   inp = getInputState()
   if(inp.Left) x -= 0.1
   if(inp.Right)x += 0.1
   …
   DrawPaddle(x) //draw
   …
}
```

Speed Cheating

- If PC/GPU is very slow player gets more time to react



Frame Rate Independent Animation

- Animation speed **should not** change with PC speed
- Idea: use system time to scale all movements
- Each frame the time one frame takes is used to scale all movements

```
y = 1
while(not finished)
{
  …
  time = GetTime()
  tFrame = time – tLast
  tLast = time
  y -= tFrame * 0.1
  DrawBall(y)
  …
}
```