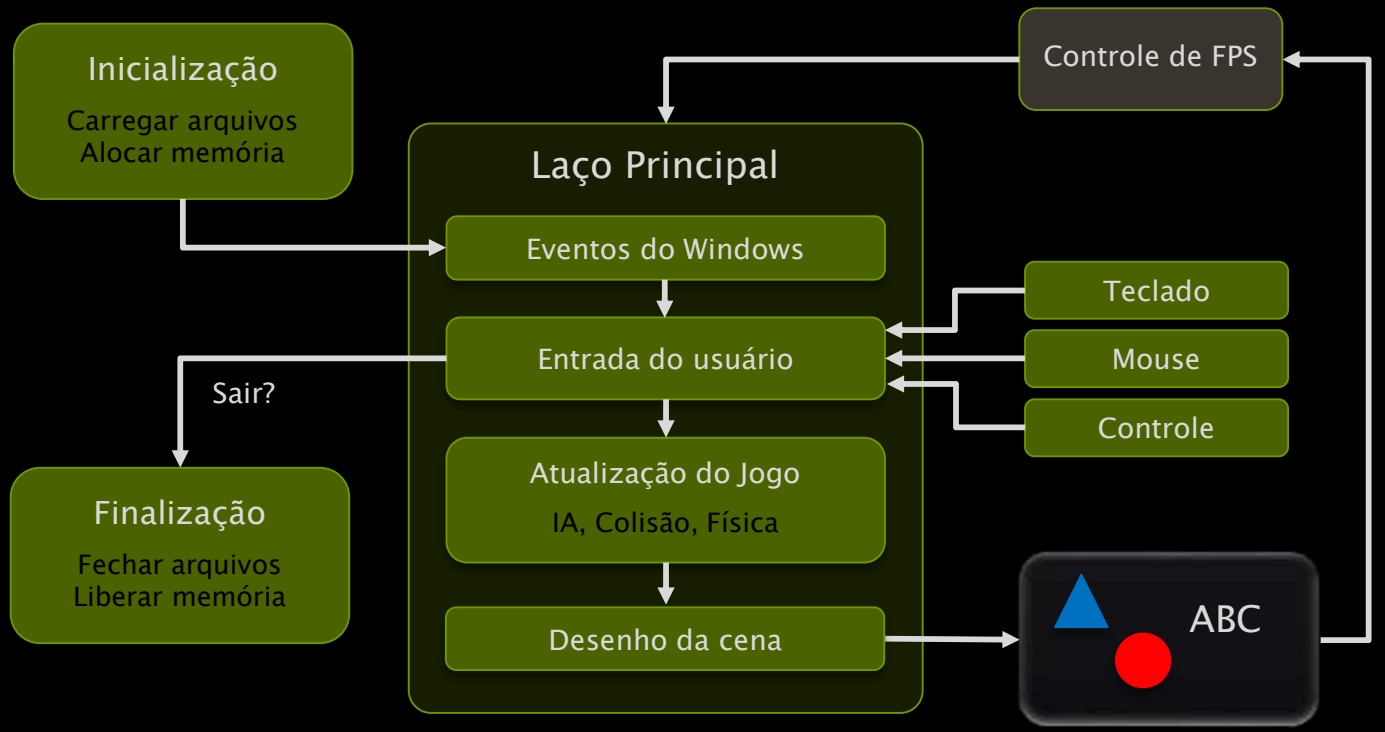
Taxa de Atualização

Programação de Jogos

Judson Santos Santiago

Introdução

- Um jogo é um laço rodando continuamente em uma frequência fixa ou variável

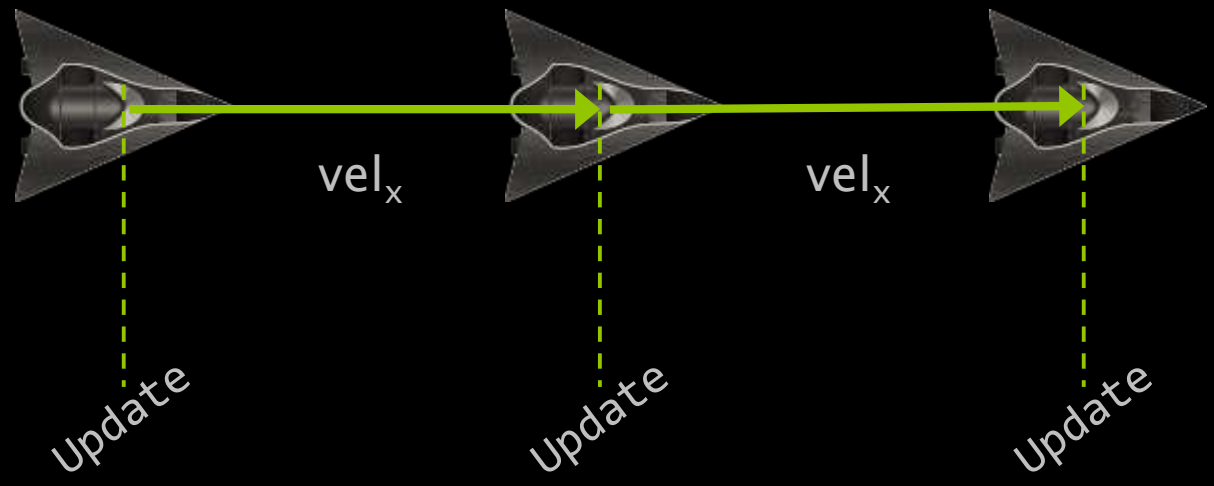


Por que o controle de FPS é necessário?

Introdução

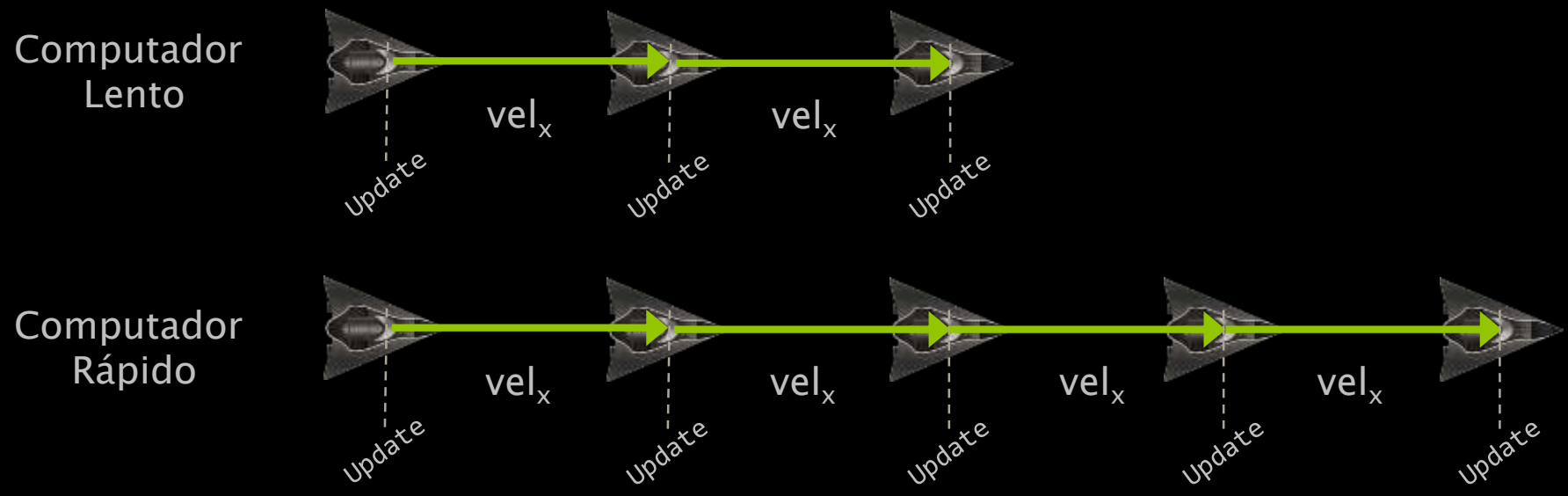
- O movimento de um objeto na tela é feito através da translação do objeto por um valor constante a cada atualização do jogo

```
Game::Update()
{
    // deslocamento
    // da nave com
    // velocidade
    // velx
    x = x + velx;
}
```



Introdução

- Se não houver um controle da taxa de atualização, quanto mais rápido o computador maior será a velocidade do objeto

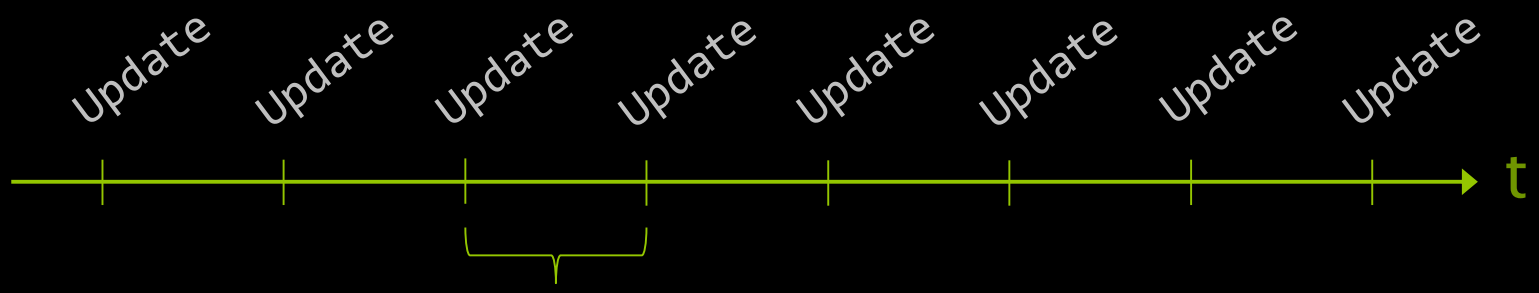


Introdução

- Para movimentar os objetos em uma velocidade independente de máquina, existem duas soluções:
  - Taxa Constante: fixar a taxa de atualização
    - 30 FPS e 60 FPS são os valores comumente utilizados
    - Pressupõe que a máquina seja capaz de atingir esses valores
  - Taxa Variável: medir o tempo para processar cada quadro
    - Mover os objetos por um valor proporcional ao tempo do último quadro
    - Máquinas mais rápidas vão apresentar cenas mais fluídas

Taxa Constante

- O objetivo é manter a taxa de atualização do jogo constante em todas as máquinas

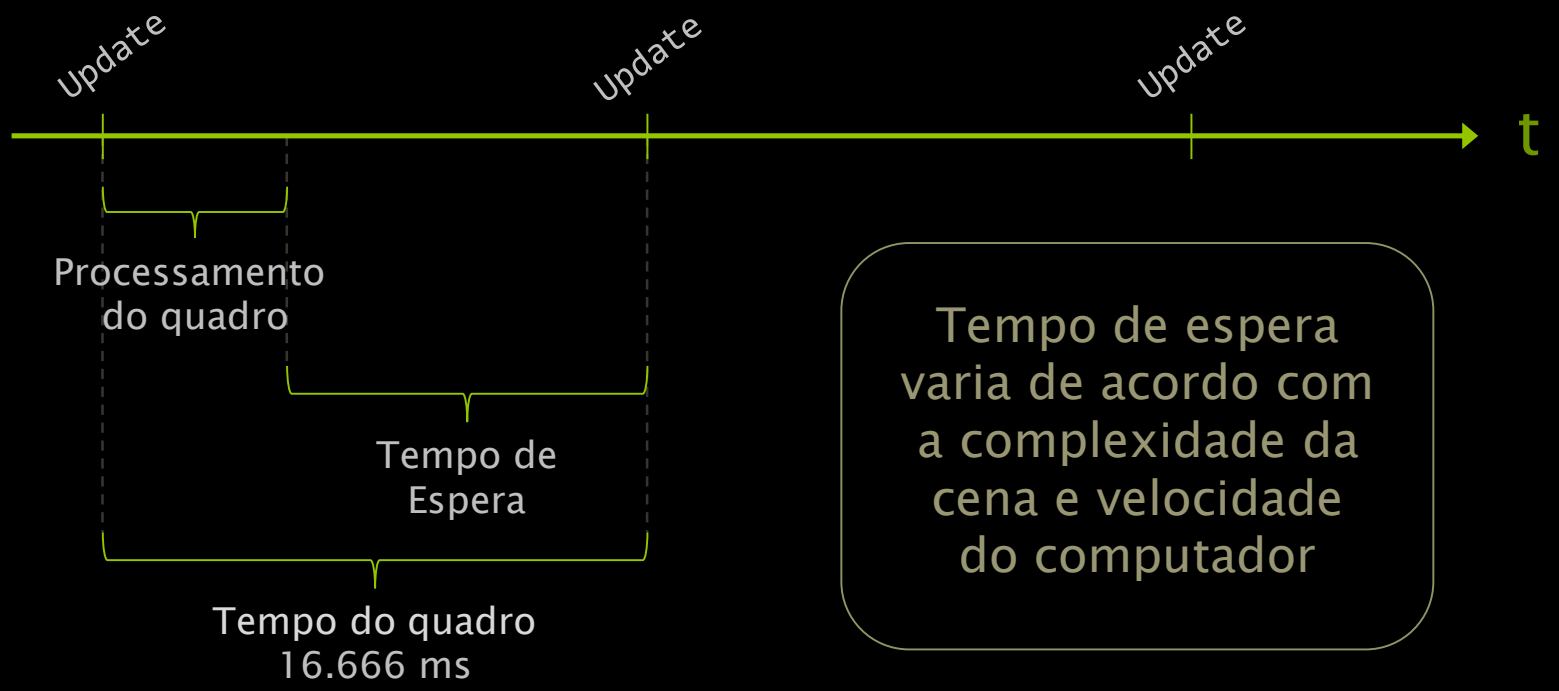


Tempo do quadro

16.666 ms
(60 FPS)

Taxa Constante

- Se o processamento do quadro levar menos tempo que o desejado, deve-se esperar



Tempo de espera varia de acordo com a complexidade da cena e velocidade do computador

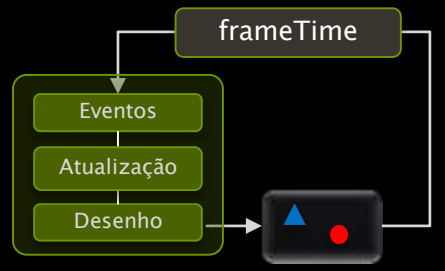
Taxa Constante

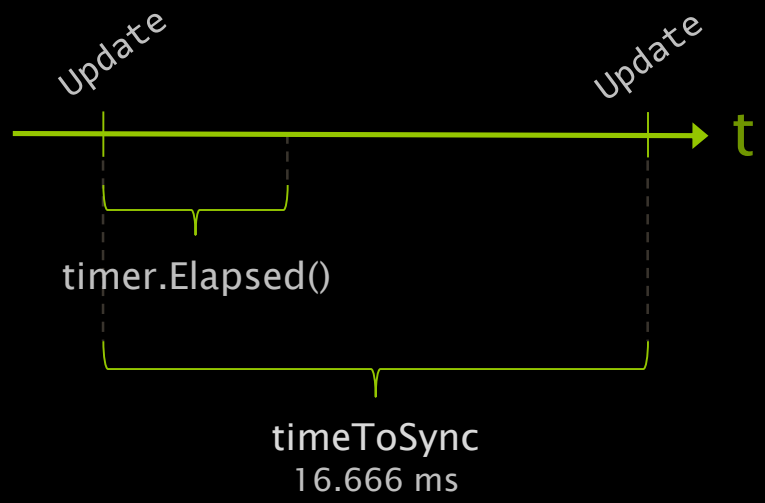
- A espera pode ser implementada por um laço:
  - Que não faz nada: uso máximo da CPU

```
// espera até atingir o tempo alvo
while (timer.Elapsed() < timeToSync)
    continue;
```

  - Que dorme por um pequeno intervalo

```
// dorme até atingir o tempo alvo de sincronização
while ((frameTime = timer.Elapsed()) < timeToSync)
{
    if ((timeToSync - frameTime) > 0.002f)
        Sleep(1);
    else
        Sleep(0);
}
```





Taxa Constante

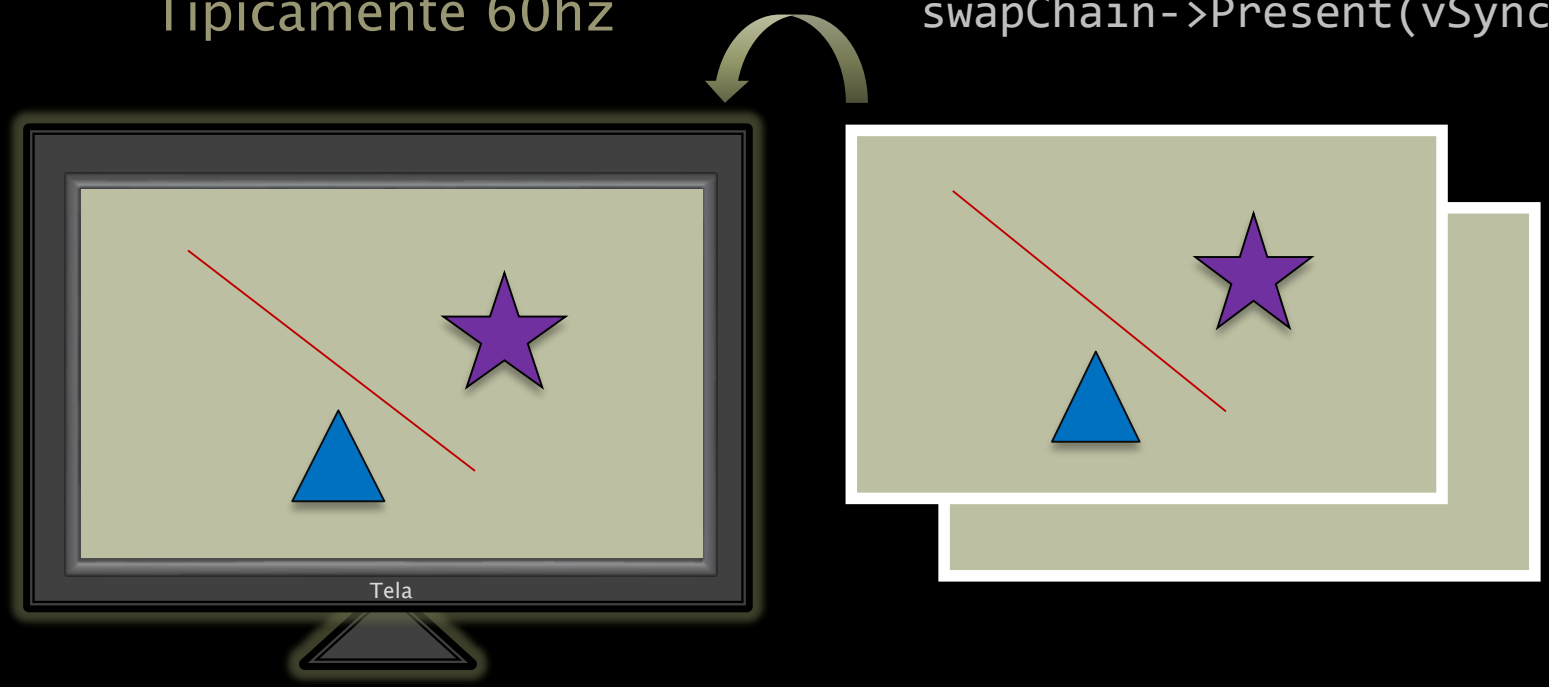
- Na prática, fazer o controle manual da taxa de atualização para mantê-la constante é problemático:
  - É muito sensível a oscilações do sistema operacional
    - Sistema Operacional é multitarefa
    - Laço pode ser suspenso pelo agendador de tarefas
    - Função `Sleep` não é precisa
- Soluções:
  - Usar a taxa de atualização do monitor (ligar o VSync)
  - Usar uma taxa de atualização variável

Taxa Constante

- Vertical Sync é o tempo entre atualizações da tela

Tipicamente 60hz

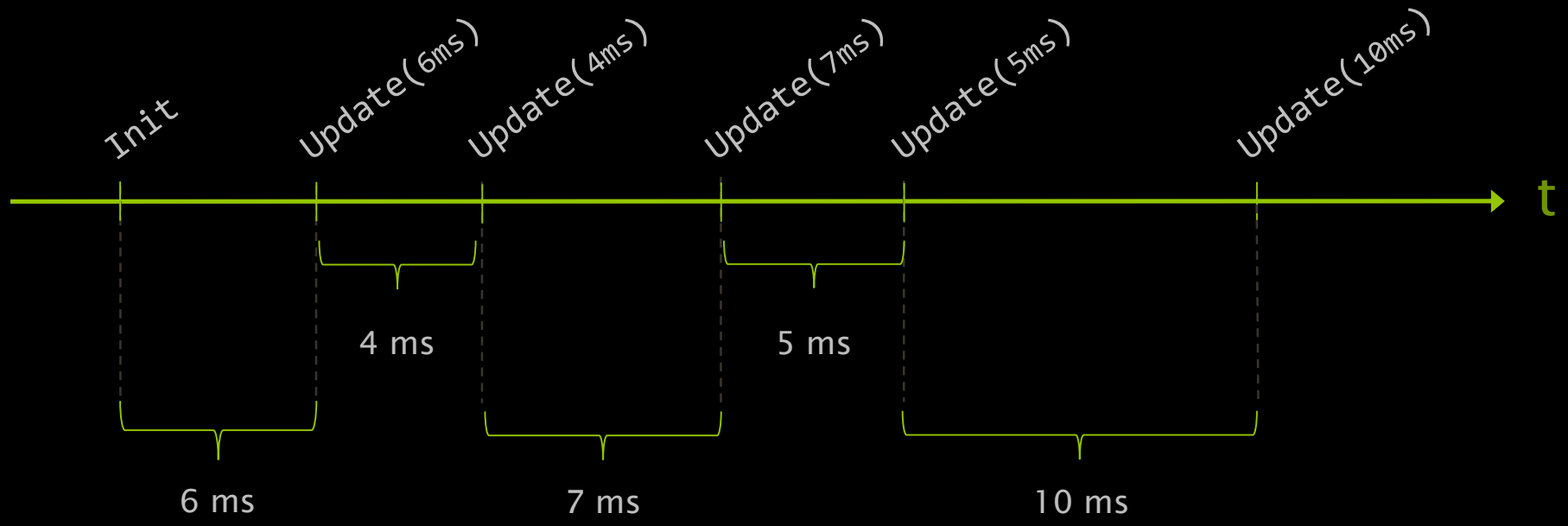
```
swapChain->Present(vSync, NULL);
```



A espera pode ser feita na hora de apresentar um novo quadro na tela

Taxa Variável

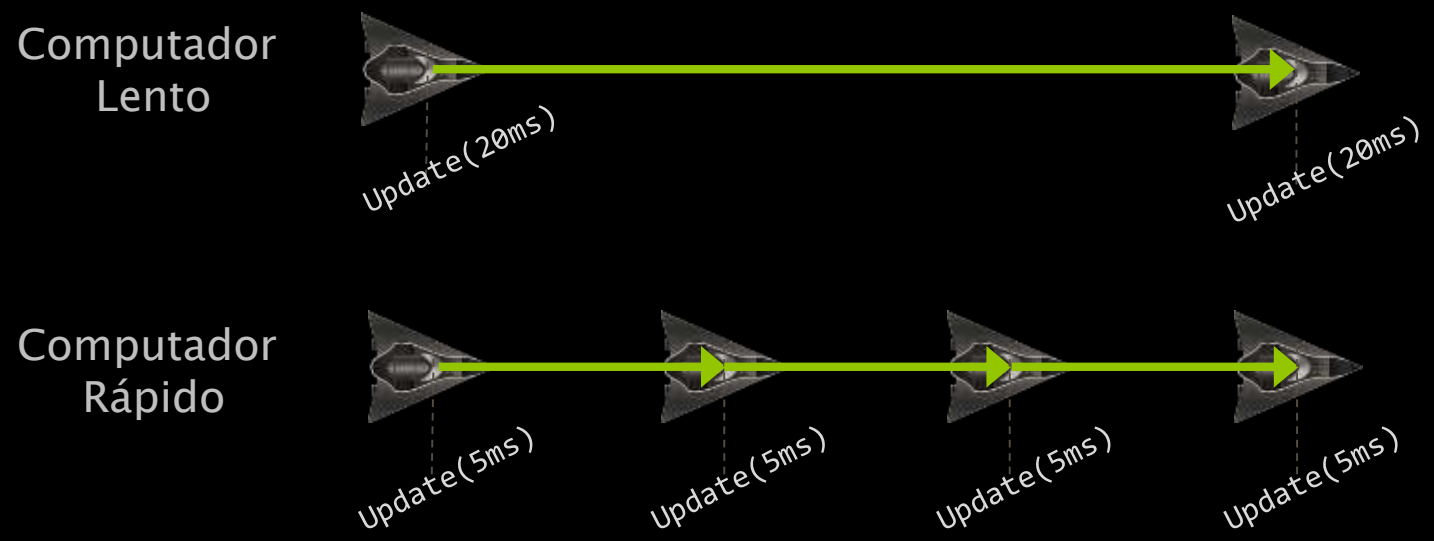
- O deslocamento dos objetos deve ser proporcional ao tempo de processamento do quadro



Taxa Variável

- Deslocamento deve ser proporcional ao frameTime

```
// deslocamento da nave proporcional ao tempo do quadro
Game::Update(float frameTime) { x = x + velx * frameTime; }
```



Timer

- Para controlar o FPS é preciso utilizar um timer
  - O C++ possui duas opções:
    - Clock() – Baixa resolução

      ```
      #include <ctime>
      ```

    - C++11 Chrono – Várias opções de resolução (baixa, média e alta)

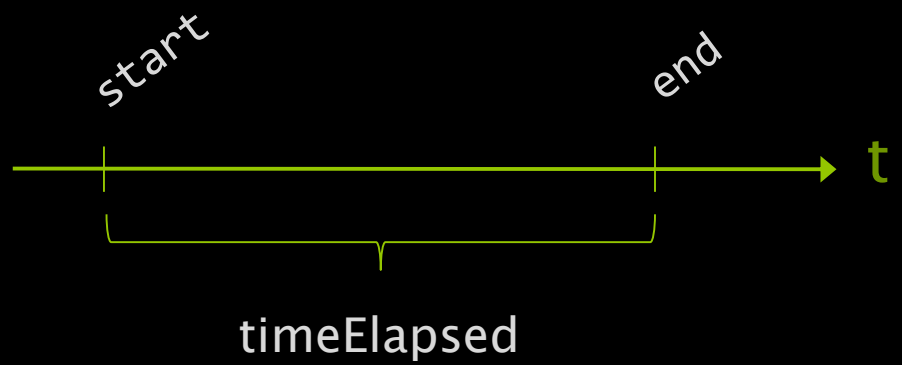
      ```
      #include <chrono>
      ```

  - O Windows possui três funções:
    - GetTickCount() – Baixa Resolução
    - timeGetTime() – Resolução Média (winmm.lib)
    - QueryPerformanceCounter() – Alta Resolução

      ```
      #include <windows.h>
      ```

Timer

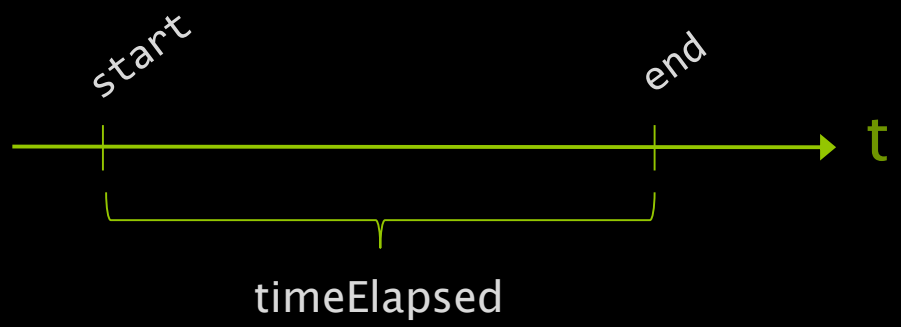
- Um quadro pode ser processado em microssegundos
  - Uma forma de obter esta resolução é utilizando o timer de alta-resolução do Windows através da função QueryPerformanceCounter



O tempo transcorrido é dado em ciclos

Timer

- Um quadro pode ser processado em microssegundos
  - A frequência do timer pode ser obtida (em ciclos/segundo) através da função `QueryPerformanceFrequency`



O tempo transcorrido precisa ser dividido pela frequência do timer

$$\frac{\text{timeElapsed}}{\text{timerFreq}} = \frac{(\text{ciclos})}{(\text{ciclos/seg.})} = \frac{\cancel{(\text{ciclos})}}{} \times \frac{(\text{seg.})}{\cancel{(\text{ciclos})}} = \text{segundos}$$

Timer

- O exemplo abaixo ilustra a utilização das funções:

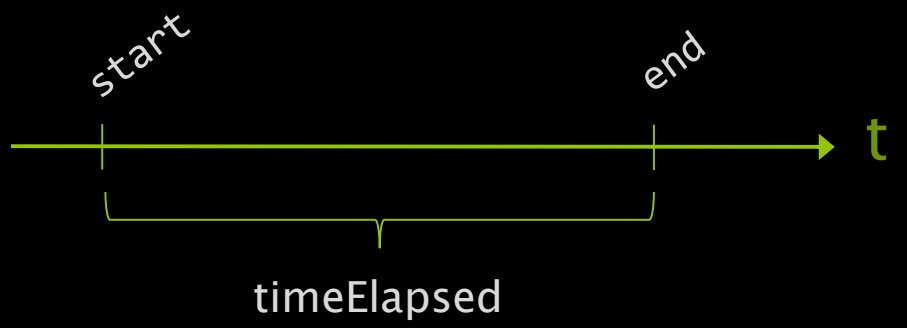
```
LARGE_INTEGER start, end, freq;
QueryPerformanceFrequency(&freq);
QueryPerformanceCounter(&start);

// faz algo que leva tempo...

QueryPerformanceCounter(&end);

// calcula tempo transcorrido (em ciclos)
long long timeElapsed = end.QuadPart - start.QuadPart;

// converte tempo para segundos
double seg = timeElapsed / double(freq.QuadPart);
```



Conclusão

- O controle de FPS deve ser feito para evitar que a velocidade do jogo dependa da velocidade da máquina
- O controle pode ser feito usando:
  - Taxa constante: atualizações em intervalos constantes
    - 33.333ms (30FPS)
    - 16.666ms (60FPS)
  - Taxa variável: os objetos devem se movimentar em quantidades proporcionais ao tempo do quadro