DCC192

2025/1

UFMG

Desenvolvimento de Jogos Digitais

A3: Game Loop

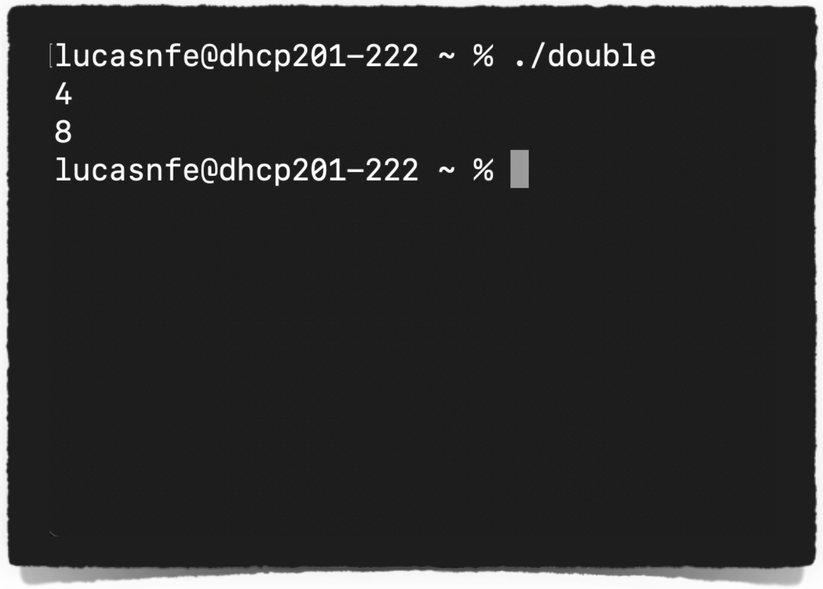
Prof. Lucas N. Ferreira

Plano de aula

- Game Loop
  - Eventos de entrada
  - Atualização de objetos do jogo
  - Geração de saída
- Gerenciamento do tempo do jogo
  - Intervalos de tempo (FPS) fixos
  - Intervalos de tempo (FPS) variáveis
  - Intervalos de tempo Fixos (FPS) apenas para o Update

Execução em lote (batch)

Os programas que escrevemos na graduação são geralmente executados em lote (*batch*):

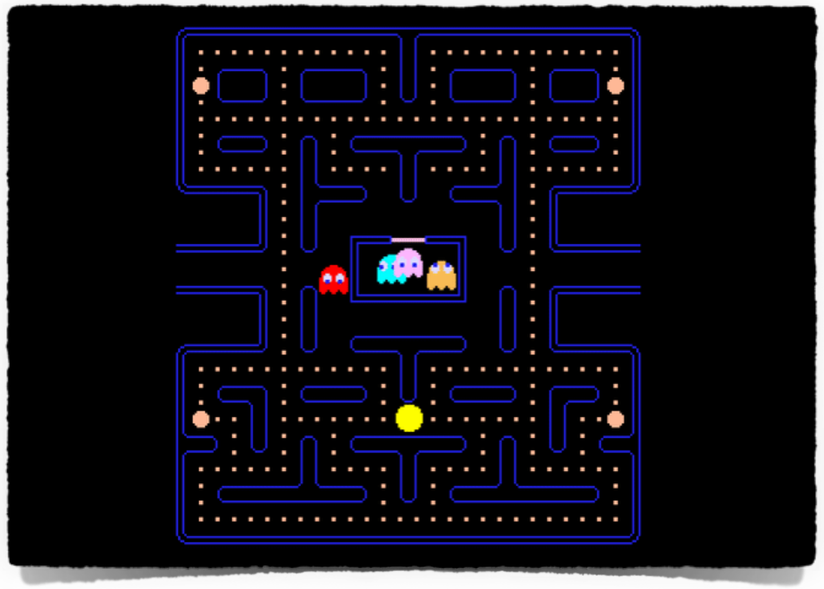



```
1. int x;
2. cin >> x;
3. cout << 2 * x << endl;
```

- Processa dados de uma só vez, sem interação do usuário durante a execução
- Foco em eficiência no processamento de dados

Game Loop

Jogos digitais são programas interativos em tempo real que executam em loop:



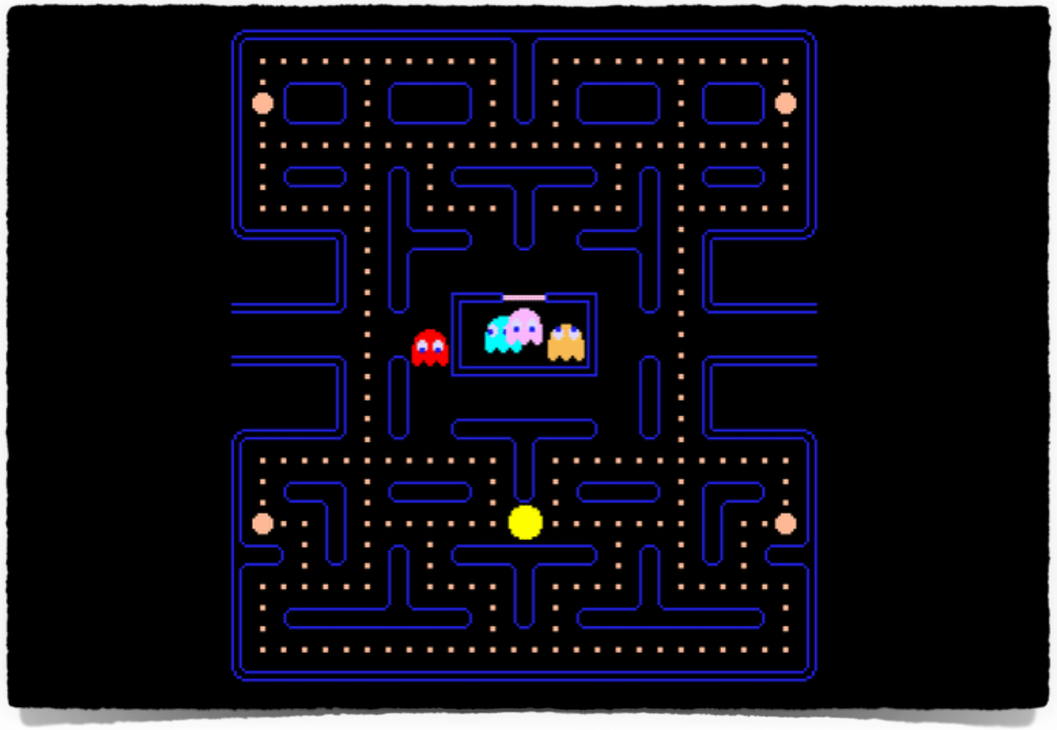
```
1. while Game is running
2.     Process input
3.     Update game world
4.     Generate output
```

Atualiza estado do jogo mesmo que nenhum evento tenha sido produzido!
Ex.: posição dos fantasmas

- Processa eventos do usuário, atualiza o estado jogo e gera saída constantemente
- Foco em baixa latência e alta taxa de quadros

Game Loop: Processar Entrada

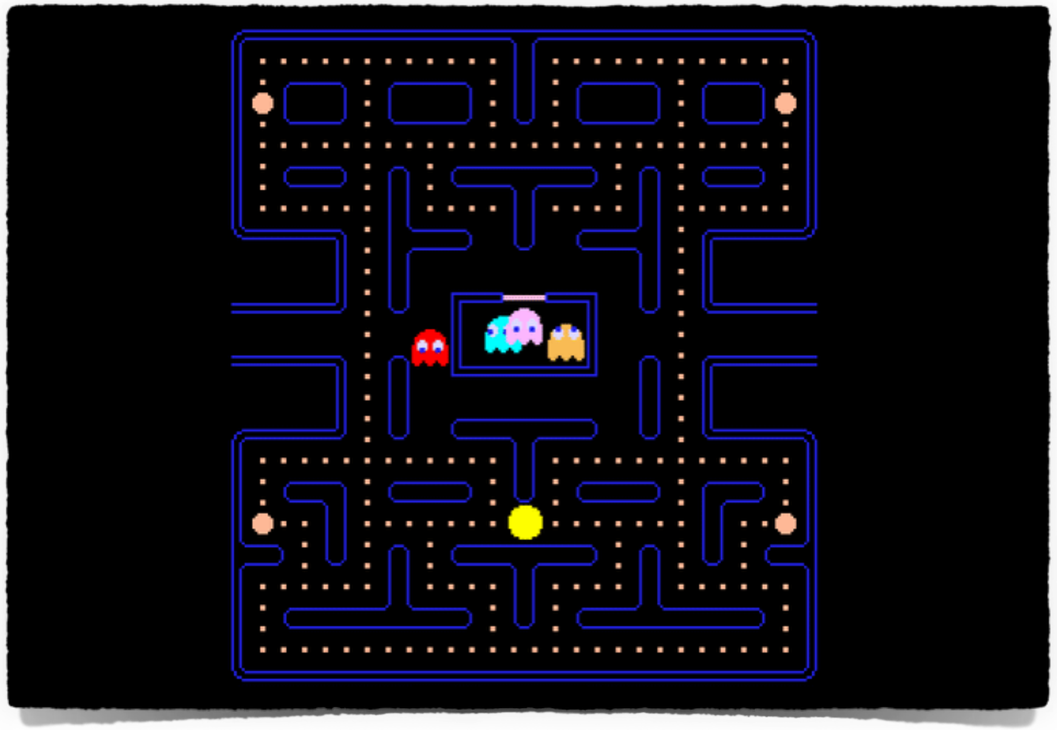
1. A primeira etapa de todo game loop é processar os eventos de entrada do jogador:



```
while Game is running
    ▸ Process input
        ▸ ⌨️ Keyboard
        ▸ 🖱️ Mouse
        ▸ 🕹️ Joystick
        ▸ ...?
    Update game world
    Generate output
```

Game Loop: Atualizar Objetos do Jogo

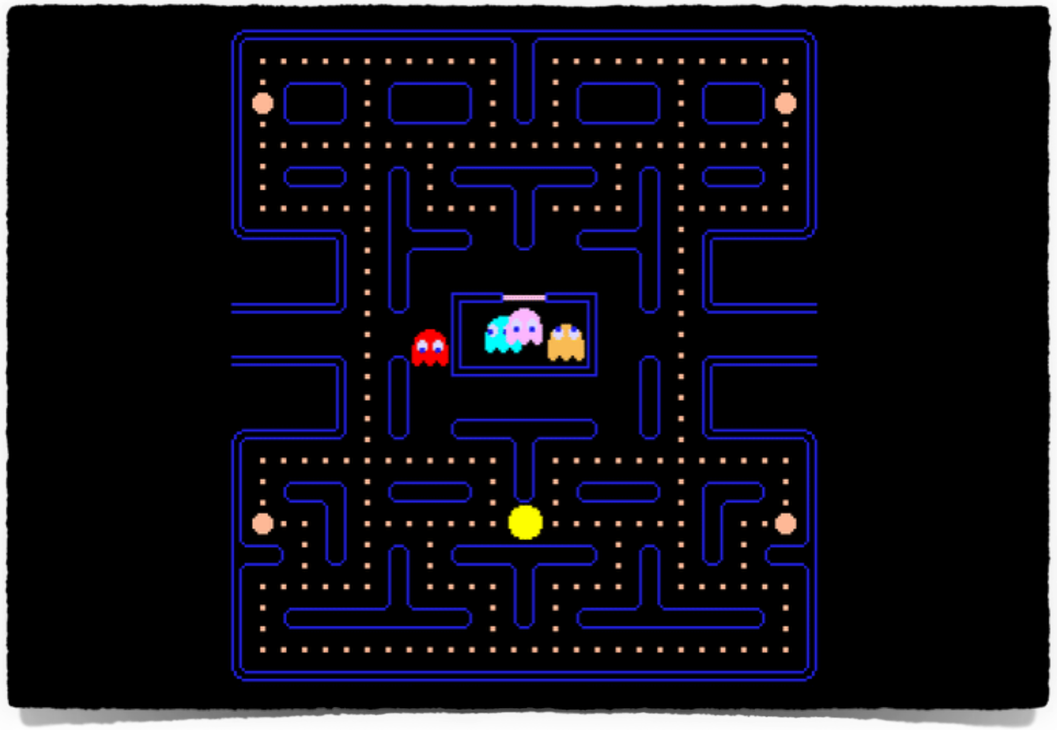
2. A segunda etapa é a **atualização do estado de todos os objetos do jogo**:



```
while Game is running
        Process input
        ▸ Update game world
            ▸ Pacman
            ▸ Ghosts
            ▸ ...?
        Generate output
```

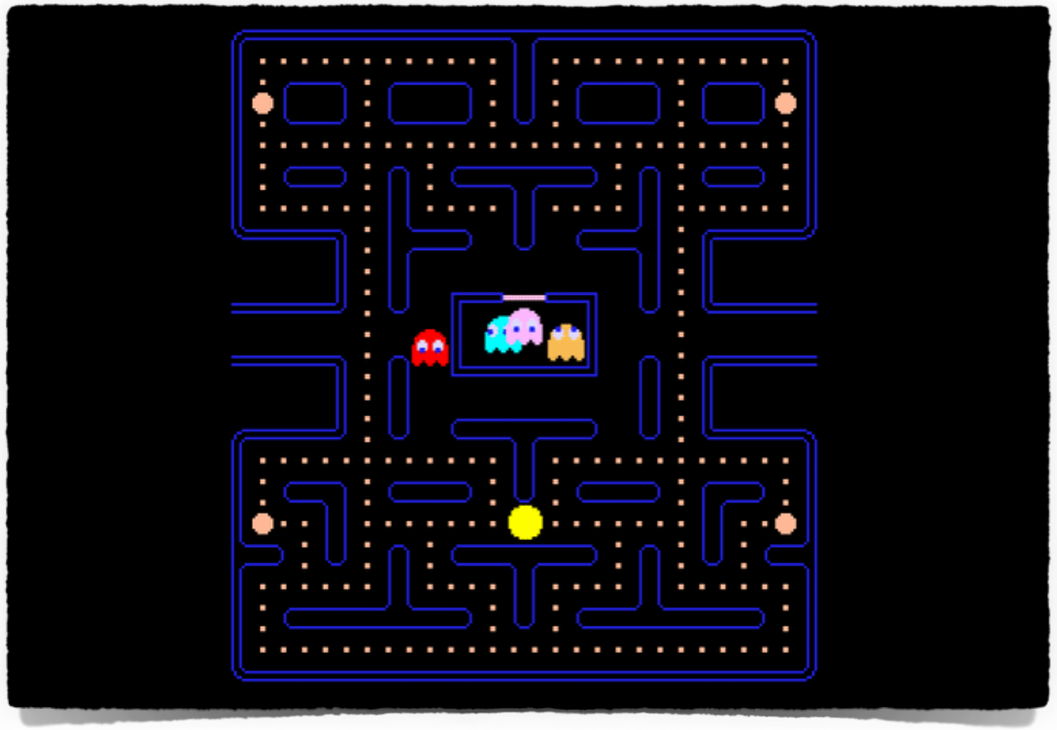
Game Loop: Gerar Saídas

3. A terceira etapa é **gerar saídas** com base no estado atualizado dos objetos do mundo:



```
while Game is running
        Process input
        Update game world
        ▸ Generate output
            ▸ Image
            ▸ Sound
            ▸ ...?
```

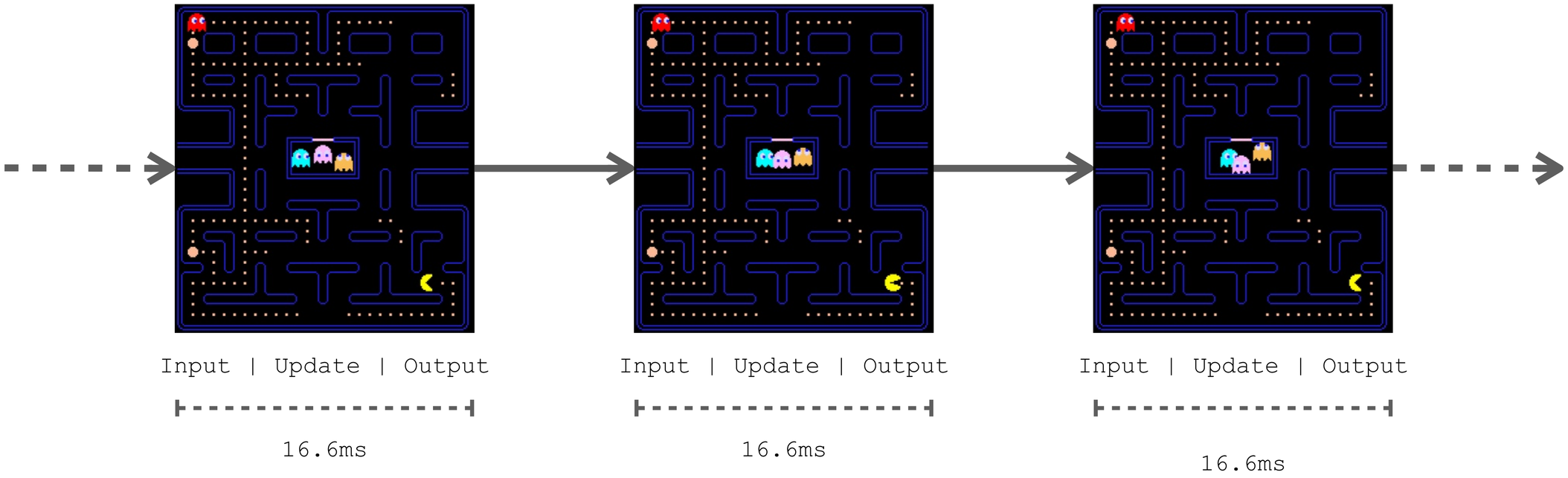
Pseudocódigo do PacMan



```
while player.lives > 0
   // Processar entrada
   input = read raw input data

   // Atualizar o mundo do jogo
   update player.position based on input
   foreach Ghost g in world
      if player collides with g
         kill either player or g
      else
         update AI for g

   // Comer as pastilhas
   ...

   // Gerar saídas
   draw graphics
   play audio
```

Execução em lote vs. jogos

Pense em um jogo como uma sequência de quadros, cada um com um limite de tempo definido.



Por exemplo: 60 FPS, 1/60 = 16.6ms por quadro

Tempo real vs. tempo do jogo

O **tempo do jogo** pode ser diferente do **tempo de relógio**:

Tempo decorrido no jogo

Tempo decorrido no mundo real

Tempo real vs. tempo do jogo

O tempo do jogo pode ser diferente do tempo de relógio:

Tempo decorrido no jogo

Tempo decorrido no mundo real

- **Parar o tempo (pause)**
- Mais rápido
- Mais devagar
- Voltar no tempo

Em quase todos os jogos modernos, o jogador pode pausar e resumir o jogo quando quiser.

Tempo real vs. tempo do jogo

O tempo do jogo pode ser diferente do tempo de relógio:

Tempo decorrido no jogo

Tempo decorrido no mundo real

- Parar o tempo (pause)
- **Mais rápido**
- Mais devagar
- Voltar no tempo

Em alguns jogos, como os de esporte, o tempo costuma passar mais rápido, para que a partida seja mais rápida do que na vida real.

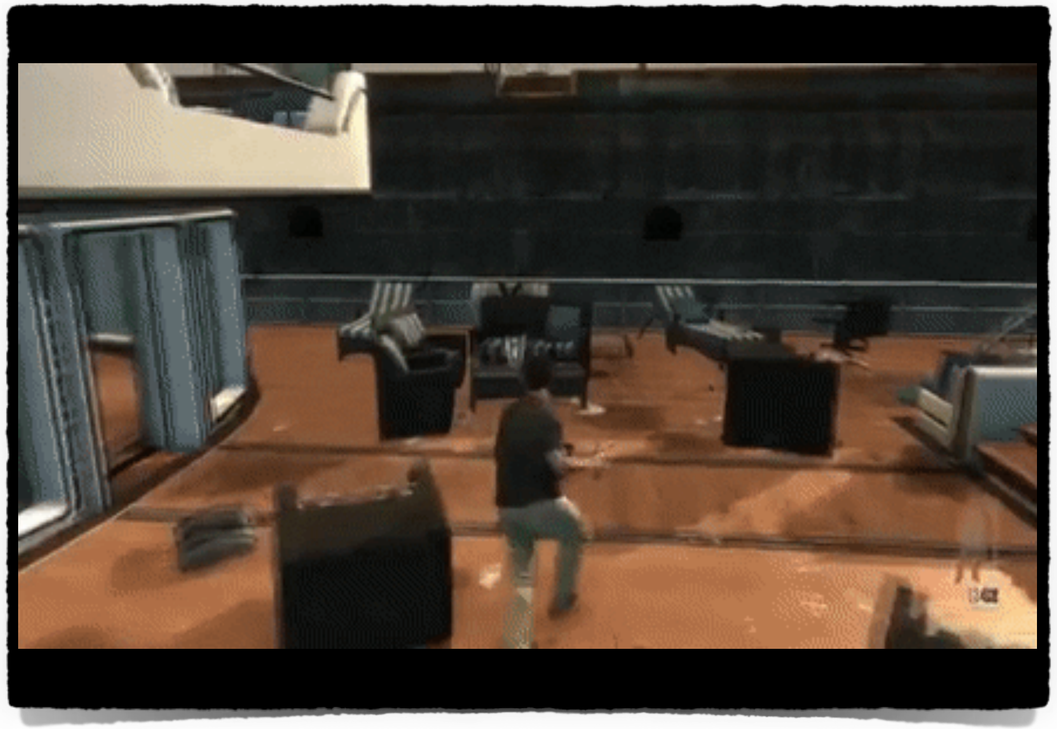
Tempo real vs. tempo do jogo

O tempo do jogo pode ser diferente do tempo de relógio:

Tempo decorrido no jogo

Tempo decorrido no mundo real

- Parar o tempo (pause)
- Mais rápido
- **Mais devagar**
- Voltar no tempo



Em outros, o tempo passa mais devagar para que o jogador tenha mais tempo para planejar e executar uma ação.

Tempo real vs. tempo do jogo

O tempo do jogo pode ser diferente do tempo de relógio:

Tempo decorrido no jogo

Tempo decorrido no mundo real

- Parar o tempo (pause)
- Mais rápido
- Mais devagar
- **Voltar no tempo**

Além disso, em alguns jogos, o jogador pode voltar no tempo como parte da própria mecânica do jogo.

Game Loop e Tempo de Jogo

```
while (gameIsRunning) {
        ProcessInput();
        Update();
        GenerateOutput();
}
```

Qual o problema com esse loop?
Nenhum controle sobre o tempo!

```
enemy.position += 5;
```

- Processador 8 MHz → jogador vai mover 1x
- Processador 16 MHz → jogador vai mover 2x

Existem três técnicas principais para gerênciamento do tempo do jogo:

- Intervalos de tempo (FPS) fixos
- Intervalos de tempo (FPS) variáveis
- Intervalos de tempo Fixos (FPS) apenas para o Update

Intervalos de tempo (FPS) fixos

```
MS_PER_FRAME = 16.6
while (gameIsRunning) {
        start = getCurrentTime();
        ProcessInput();
        Update();
        GenerateOutput();
        Sleep(start + MS_PER_FRAME - getCurrentTime())
}       Dormir no tempo que sobrou!
```

Qual o problema com esse loop?
Se o jogo rodar muito devagar, o tempo de dormir fica negativo.

Intervalos de tempo (FPS) variáveis

```
lastTime = getCurrentTime();
while (gameIsRunning) {
      deltaTime = (getCurrentTime() - lastTime)/1000;
      ProcessInput();
      Update(deltaTime);
      GenerateOutput();
      lastTime = getCurrentTime();
}
```

Todos os objetos são atualizados em função de deltaTime!

```
// Atualiza a posição x por 150 pixels/segundo
position.x += 150 * deltaTime;
```

Intervalos de tempo (FPS) variáveis

```
// Atualiza a posição x por 150 pixels/segundo
position.x += 150 * deltaTime;
```

Quantos pixels esse objeto se move em 1 segundo se o jogo é atualizado a:

(a) 30 quadros por segundo?

A 30 FPS, o delta time é ~0.033, então o objeto irá se mover a ~5 pixels/quadro;
30 * 5 = 150 pixels/segundo.

(b) 60 quadros por segundo?

A 60 FPS, o delta time é ~0.016, então o objeto irá se mover a ~2.5 pixels por quadro;
60 * 2.5 = 150 pixels/segundo.

Mesma quantidade de movimento, mas com 60 FPS será mais suave!

Intervalos de tempo (FPS) variáveis

```
lastTime = getCurrentTime();
while (gameIsRunning) {
        deltaTime = (getCurrentTime() - lastTime)/1000;
        ProcessInput();
        Update(deltaTime);
        GenerateOutput();
        lastTime = getCurrentTime();
}
```

Qual o problema com delta time loop?

1. Física e IA instáveis (ex. pulos diferentes em jogos de plataforma)
2. Delta time muito grande (ex. durante debug)

Intervalo de tempo (FPS) fixo apenas para o Update

```
MS_PER_FRAME = 16.6
MAX_DELTA_TIME = 0.05
lastTime = getCurrentTime();
while (gameIsRunning) {
        ProcessInput();
        Sleep(lastTime + MS_PER_FRAME - getCurrentTime())
        deltaTime = (getCurrentTime() - lastTime)/1000;
        if deltaTime > MAX_DELTA_TIME:
          deltaTime = MAX_DELTA_TIME
        lastTime = getCurrentTime();
        Update(deltaTime);
        GenerateOutput();
}
```

1. Esperar o tempo que falta para completar 16.6*ms* desde o último Update!

2. Limitar deltaTime!

Próxima aula

A4: Game Objects

- Gerenciamento de objetos do jogo
- Modelo de hierarquia de classes
- Modelo de componentes
- Modelo híbrido