DCC192

2025/2

UFMG

Desenvolvimento de Jogos Digitais

A12: Câmeras 2D

Prof. Lucas N. Ferreira

Logística

Avisos

- O **TP2: Asteroids** foi publicado com entrega para o dia 01/10, às 23:39h
- **Não temos aula nessa quarta-feira (atividades complementares)!**

Última Aula

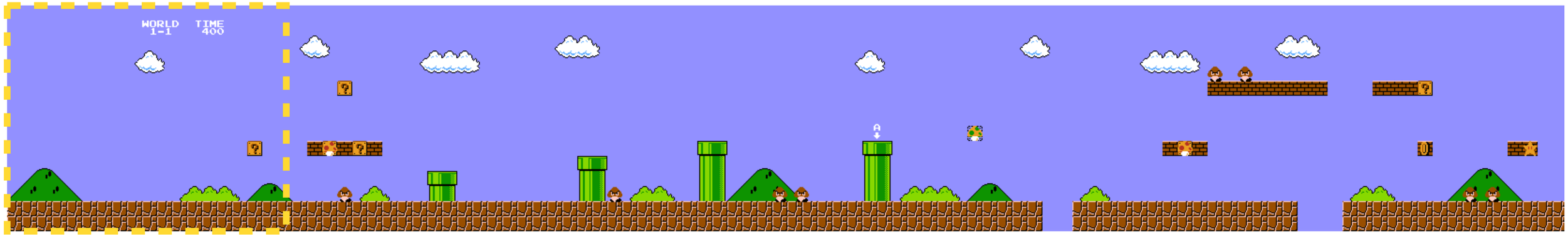
- Mapeamento de Textura em OpenGL
- Sprites e Spritesheets
- Animações 2D
- Tilesets

Plano de Aula

- Representação de Câmeras 2D
- Movimentação de Câmeras 2D
  - Suavização de Movimento
- Efeitos de câmeras 2D
  - Paralaxe
  - Chacoalhão (*Shake*)

Câmera 2D

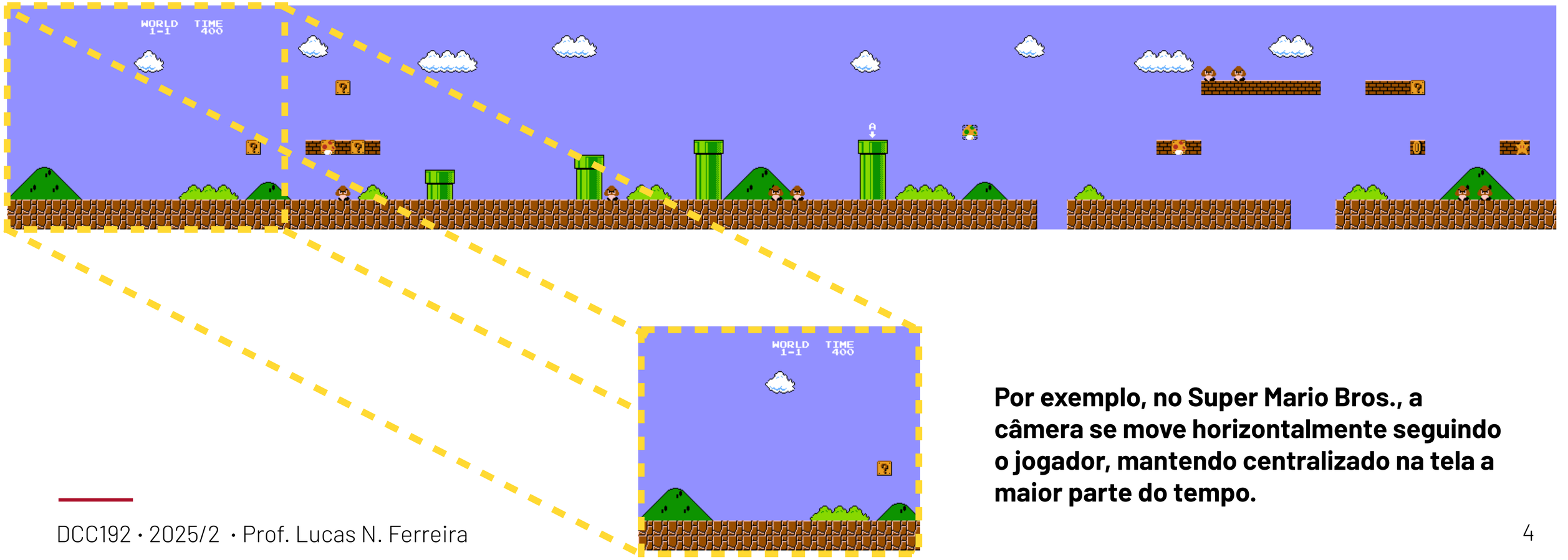
É muito comum que jogos 2D tenham mundos grandes que não cabem na tela. Nesse caso, precisamos implementar uma câmera, que se move para mostrar a região de interesse atual.



Por exemplo, no Super Mario Bros., a câmera se move horizontalmente seguindo o jogador, mantendo centralizado na tela a maior parte do tempo.

Câmera 2D

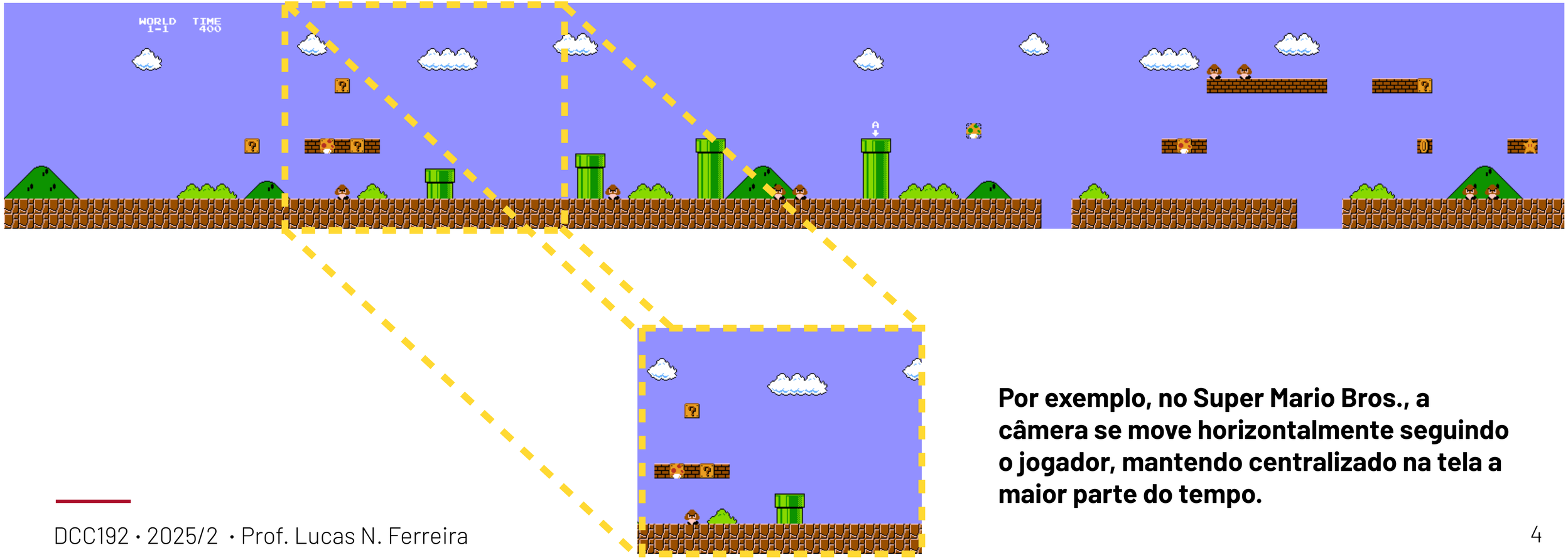
É muito comum que jogos 2D tenham mundos grandes que não cabem na tela. Nesse caso, precisamos implementar uma câmera, que se move para mostrar a região de interesse atual.



Por exemplo, no Super Mario Bros., a câmera se move horizontalmente seguindo o jogador, mantendo centralizado na tela a maior parte do tempo.

Câmera 2D

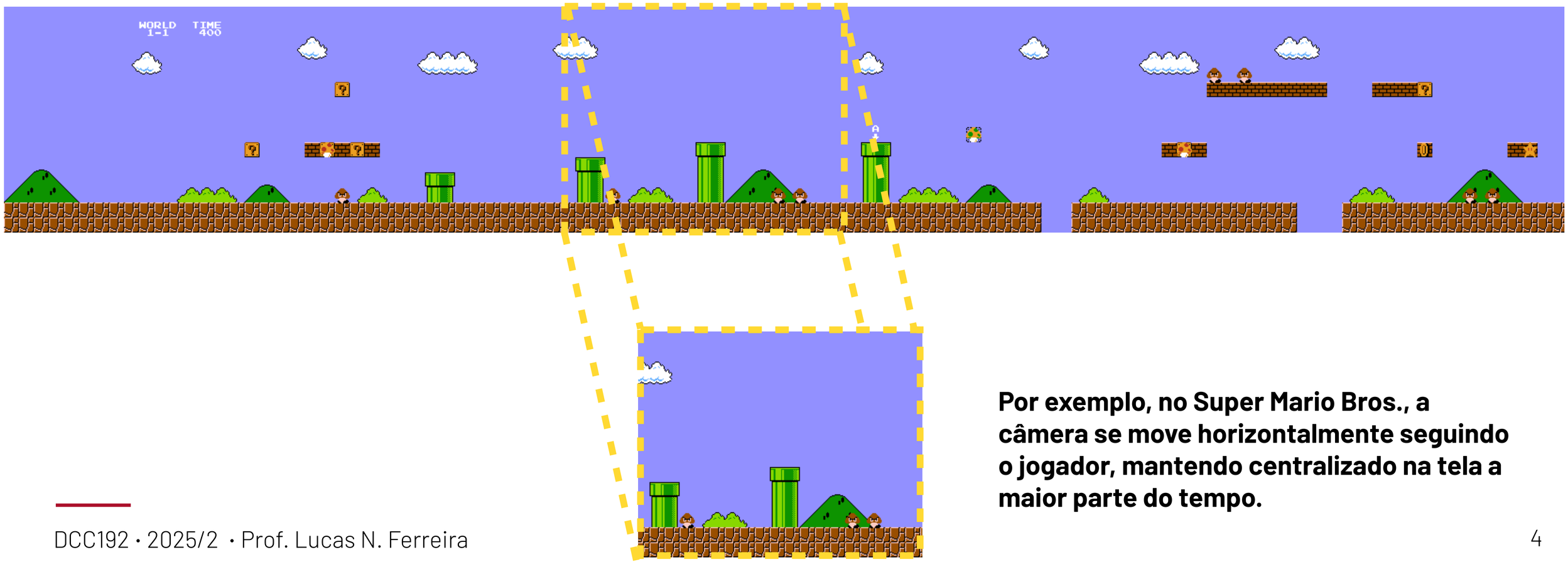
É muito comum que jogos 2D tenham mundos grandes que não cabem na tela. Nesse caso, precisamos implementar uma câmera, que se move para mostrar a região de interesse atual.



Por exemplo, no Super Mario Bros., a câmera se move horizontalmente seguindo o jogador, mantendo centralizado na tela a maior parte do tempo.

Câmera 2D

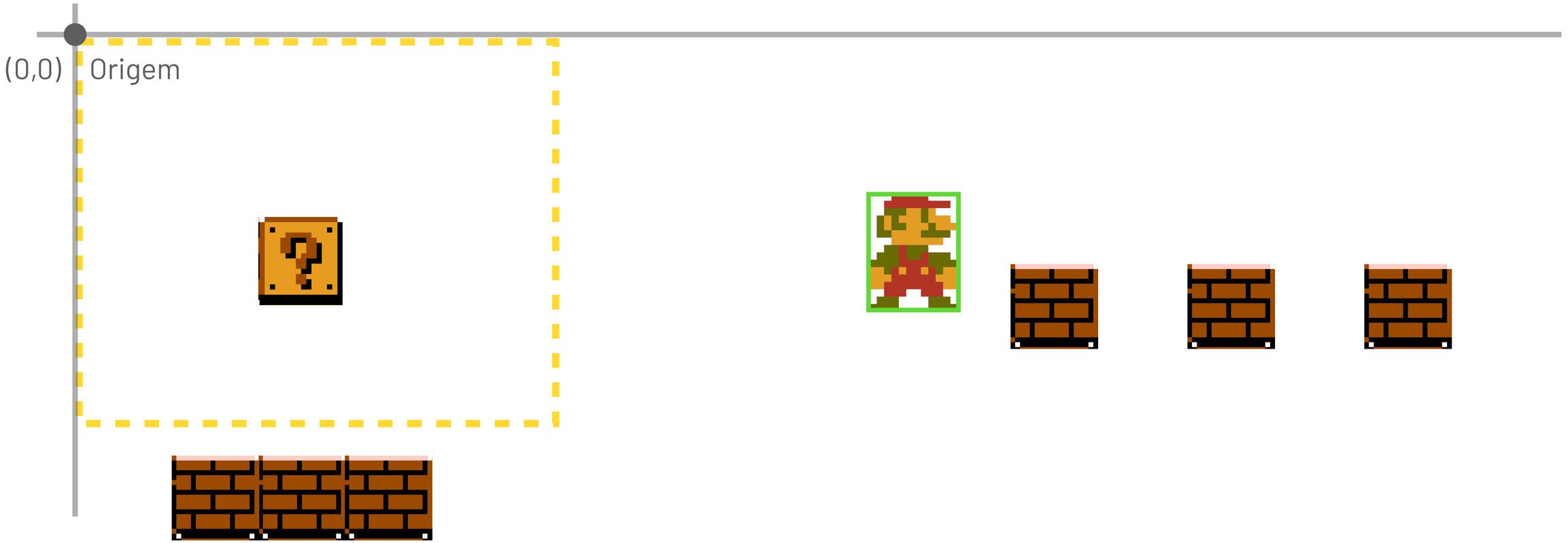
É muito comum que jogos 2D tenham mundos grandes que não cabem na tela. Nesse caso, precisamos implementar uma câmera, que se move para mostrar a região de interesse atual.



Por exemplo, no Super Mario Bros., a câmera se move horizontalmente seguindo o jogador, mantendo centralizado na tela a maior parte do tempo.

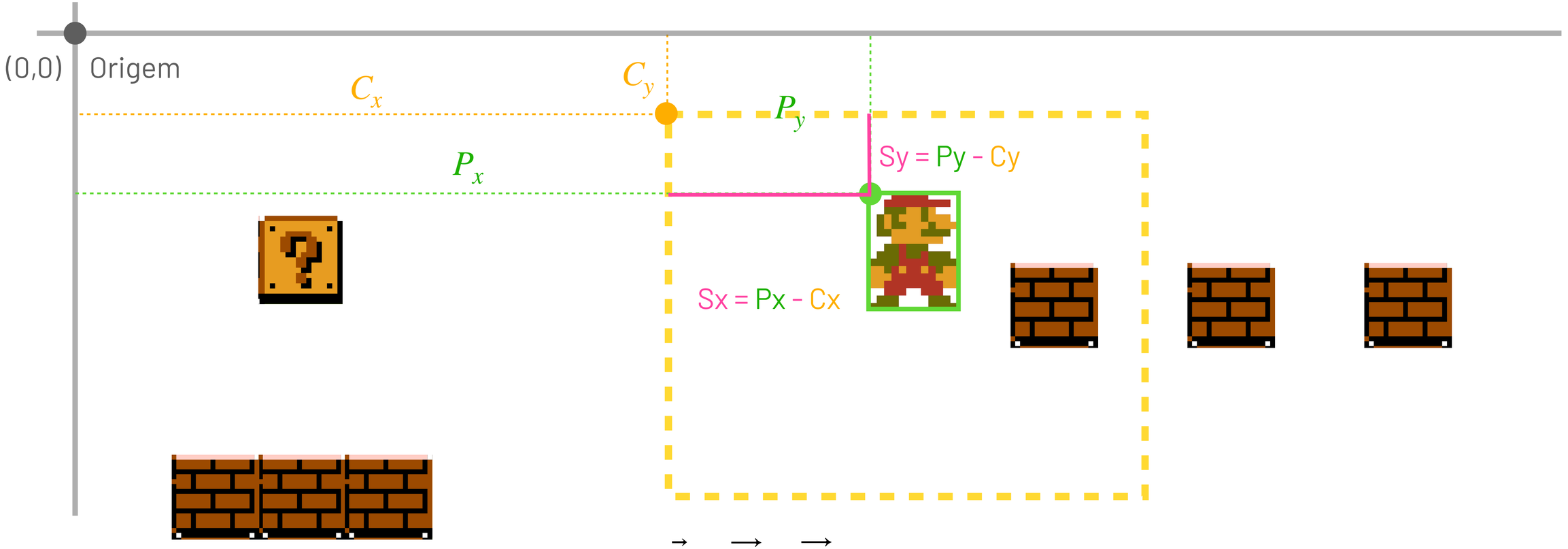
Representação de Câmeras 2D

Originalmente, os objetos são desenhados com relação à origem do mundo (o canto esquerdo superior da tela). Porém, objetos que estão fora da tela não aparecem!



Representação de Câmeras 2D

Basta desenhar os objetos com relação à posição da **câmera** C, representada por uma posição relativa à origem do mundo.



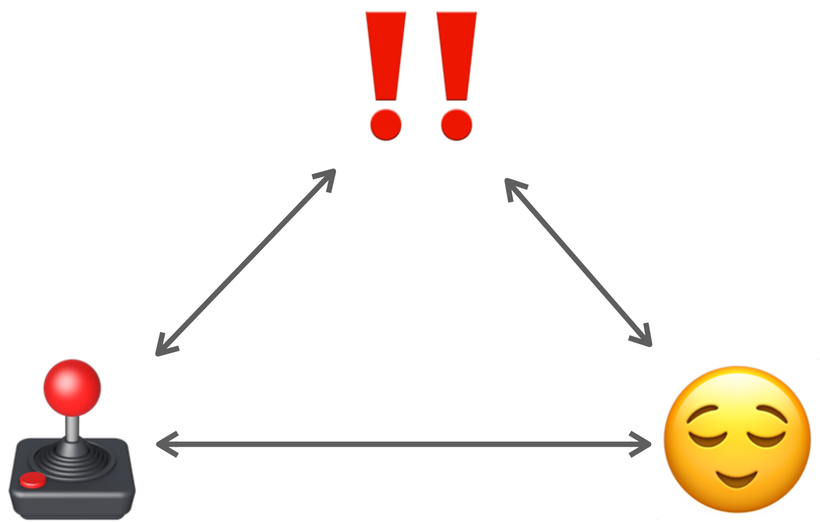
$$\vec{S} = \vec{P} - \vec{C}$$

```
screenPosition = worldPosition - cameraPosition
```

Movimentação de Câmera

O movimento da câmera deve ser controlado visando balencear três variáveis conflitantes entre si: atenção, interação e conforto.

Atenção
O que o jogador deve ver.



Interação
O que o jogador quer ver.

Conforto
Facilidade em acompanhar os movimentos.

Movimentação de Câmera

Existem duas técnicas básicas de movimentação de câmera para balancear essas variáveis, dependendo dos gráficos e das mecânicas do jogo:

1. Seguir um ponto
   - Jogador
   - Caminho
   - Alvo
2. Janela Delimitadora
   - Bidimensional
   - Unidimensional

Seguir o Jogador

A posição da câmera segue a posição do jogador em um eixo, por exemplo, no horizontal:

Centralizada

Kung-Fu Master, 1984 Irem

Deslocaca (esquerda)

Pac-land, 1984 Namco

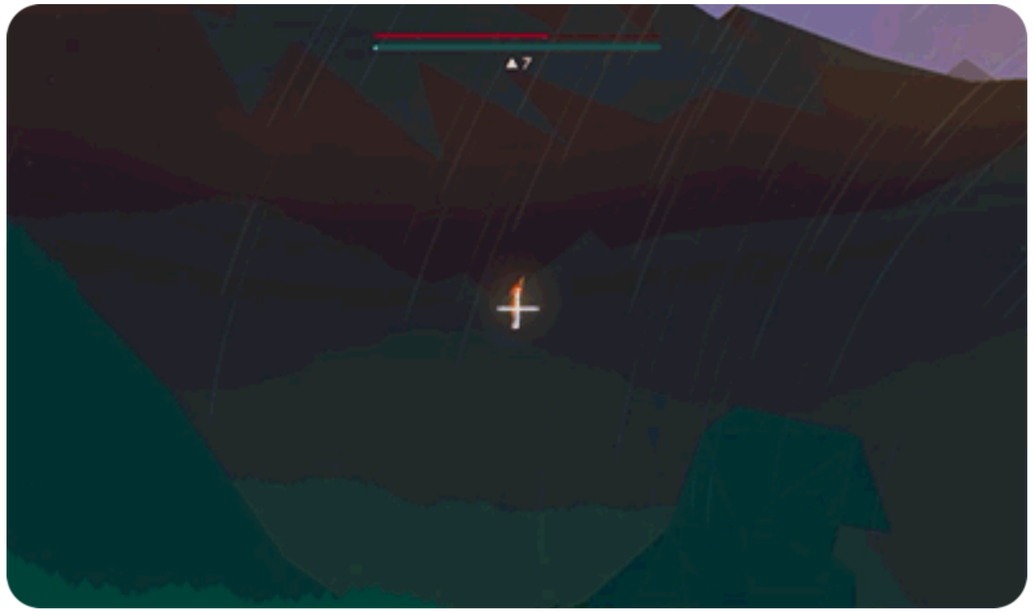
Seguir o Jogador

A posição da câmera segue a posição do jogador nos dois eixos, horizontal e vertical:

Centralizada

Terraria, 2011 Re-Logic

Deslocada (frente)

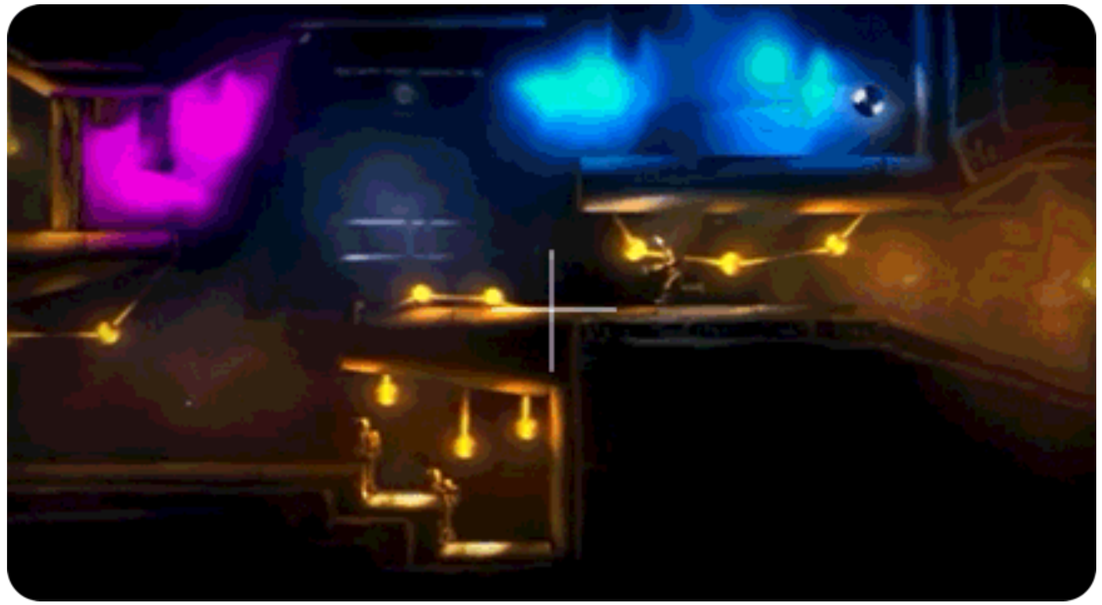


Secrets of Rætikon, 2014 Broken Rules

Seguir um alvo

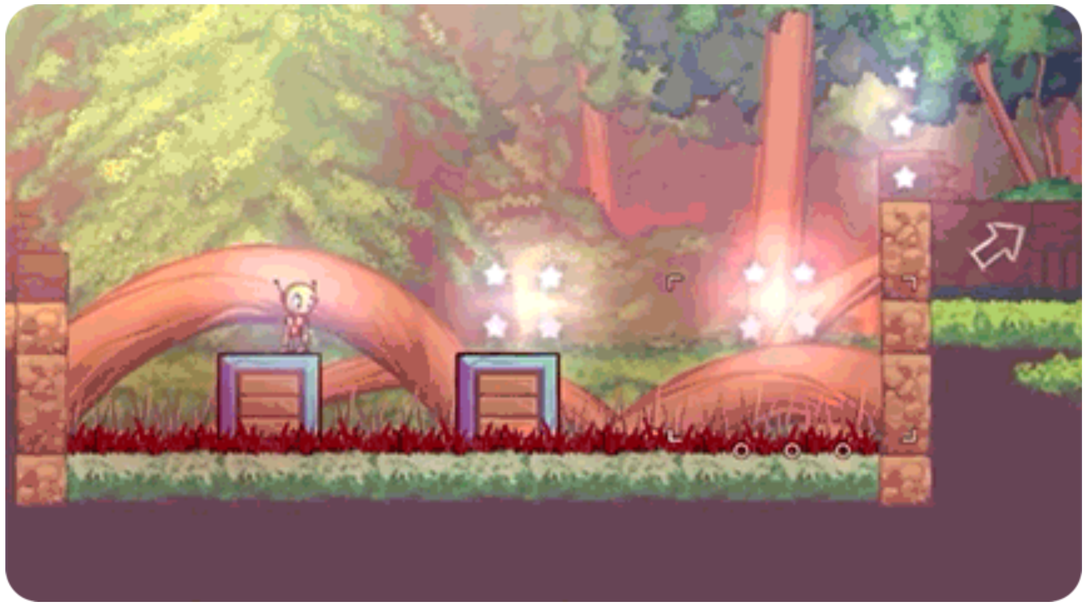
A posição da câmera segue um ponto alvo controlado pelo jogador:

Centralizada



The Swapper, 2013 Facepalm Games

Deslocada (média alvo/jogador)



Snapshot, 2012 Retro Affect

Seguir um caminho

A posição da câmera é definida por pontos em um caminho pré-definido:

Wonder Boy, 1986 Sega

Janela Delimitadora

A câmera se move quando o jogador encosta nos limites de uma janela delimitadora com posição fixa na tela:

Centralizada

Jump Bug, 1981 Hoei

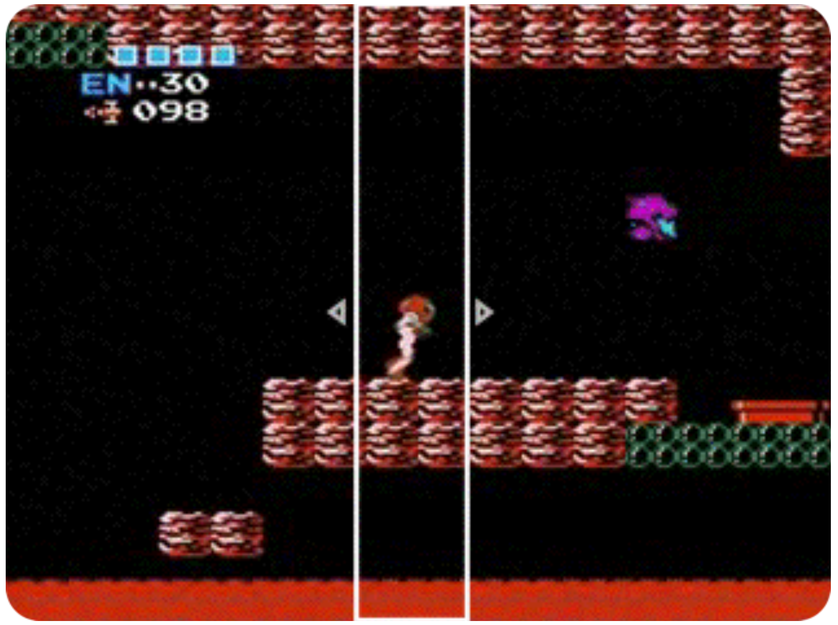
Deslocada (esquerda/cima)

Rastan Saga, 1987 Taito

Janela Delimitadora por Eixo

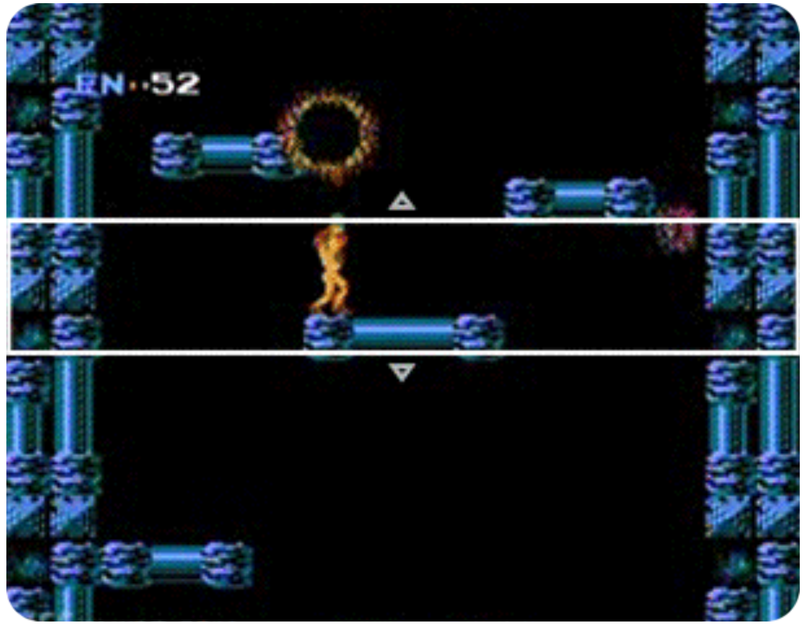
A janela pode ser definida para um único eixo dependendo da estrutura do nível:

Horizontal



Metroid, 1986 Nintendo

Vertical



Metroid, 1986 Nintendo

Janela Delimitadora por Eixo

A janela pode ser definida para um único eixo dependendo da estrutura do nível:

Bonanza Bros., 1990 Sega

Super Mario World, 1990 Nintendo

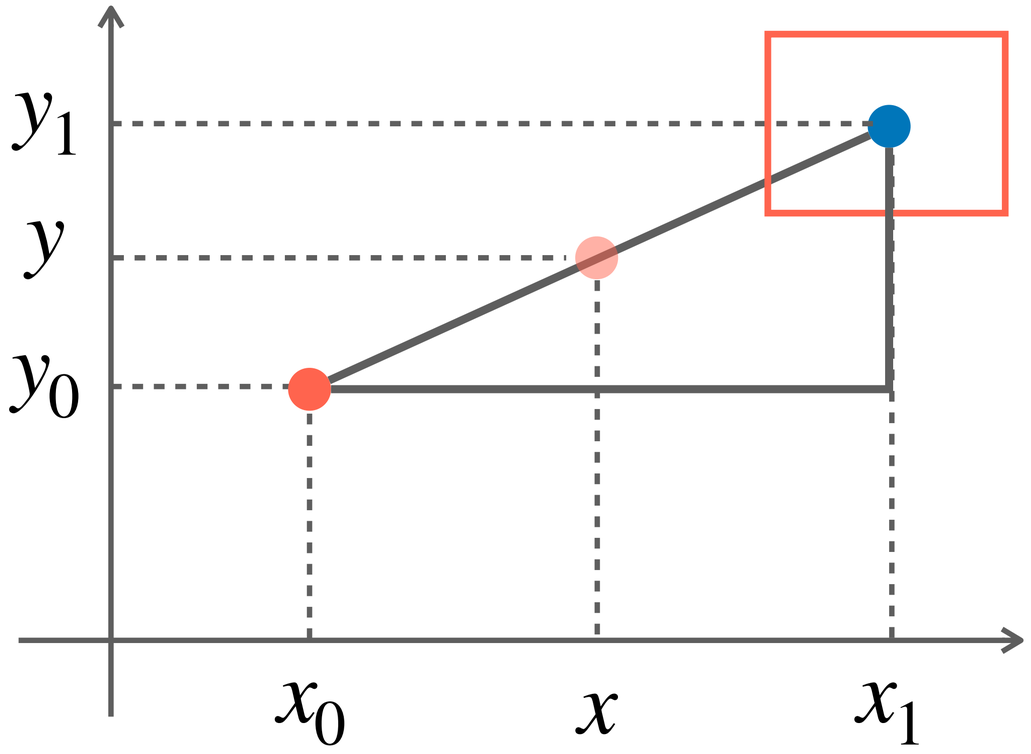
Técnicas para Suavizar Movimento

Uma câmera que se move demais pode ser desconfortável para o jogador. Uma forma simples de aliviar esse problema é **suavizar o movimento da câmera** com:

- Interpolação
- Simulação Física
- Regras para evitar movimentos

Interpolação linear

A **interpolação linear** é um método para estimar valores desconhecidos entre dois pontos dados, utilizando uma linha reta para conectar esses pontos e prever valores intermediários:



Semelhança de triângulos

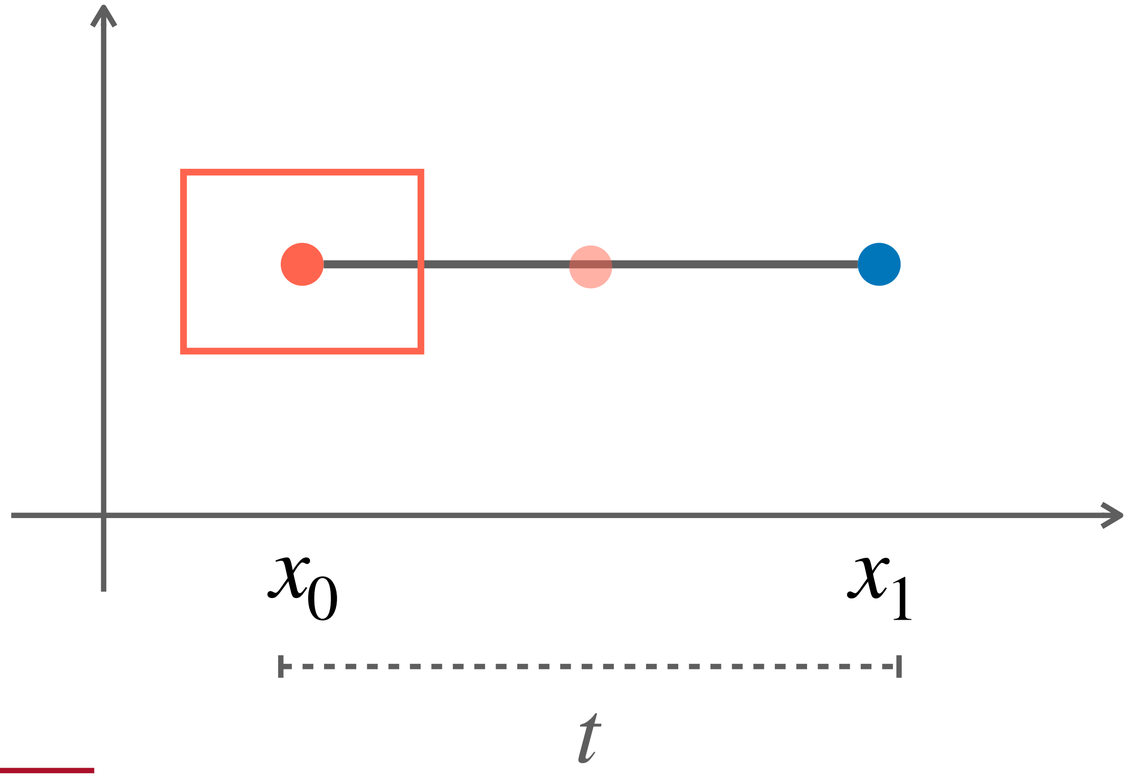
$$\frac{y - y_0}{y_1 - y_0} = \frac{x - x_0}{x_1 - x_0}$$

$$y = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0}$$

Mas nós não temos x!

Interpolação linear paramétrica

Em jogos, geralmente implementamos interpolação linear com um parâmetro $0 \leq t \leq 1$, que controla a posição do ponto (x, y) ao longo da reta:



$$y = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0}$$

$$y = y_0 + t(y_1 - y_0)$$

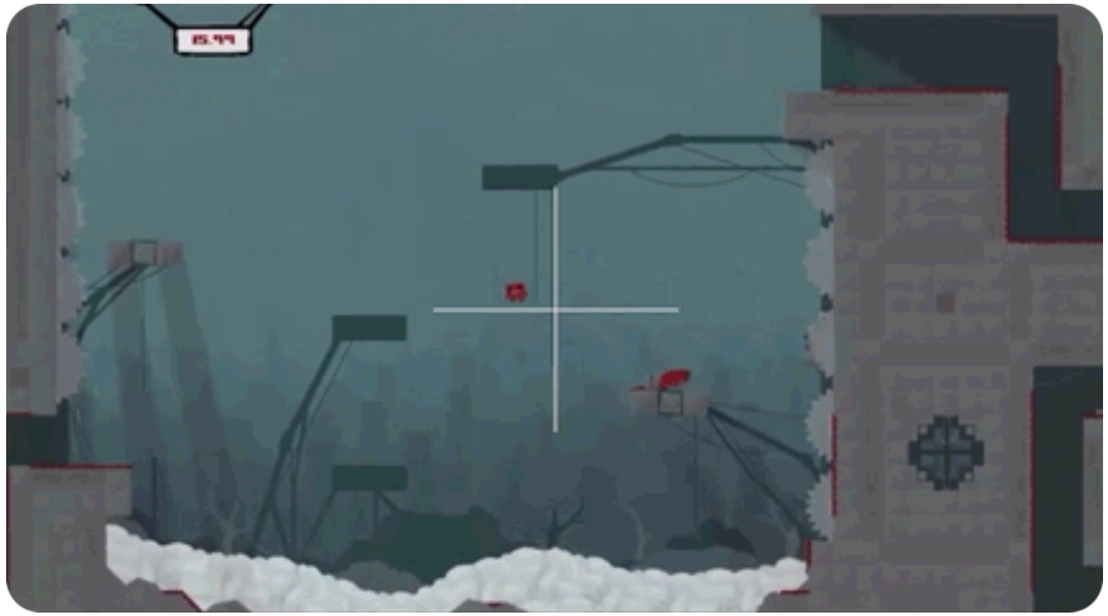
$$x = x_0 + t(x_1 - x_0)$$

- Se $t = 0 \rightarrow (x, y) = (x_0, y_0)$
- Se $t = 1 \rightarrow (x, y) = (x_1, y_1)$

Seguir com Interpolação

Reduzir continuamente a distância entre a câmera e o ponto focal usando interpolação linear:

Centralizada



Super Meat Boy, 2010 Team Meat

Deslocada (pra frente)

Jazz Jackrabbit 2, 1998 Epic Games

Seguir com Física

Aplicar forças na câmera em direção ao ponto focal para movimentá-la com aceleração:

Horizontal

Super Mario Bros, 1985 Nintendo

Horizontal/Vertical

Never Alone, 2014 Upper One Games

Regras para Evitar Movimentos

Outra forma de deduzir desconforto é **criar regras para minizar** a movimentação da câmera. Por exemplo, seguir o jogador no eixo vertical, mas apenas quando ele aterriza numa plataforma.

Super Mario World, 1990 Nintendo

Rayman, 1995 Ubisoft

Combinando Controle e Suavização

Na prática, essas técnicas são combinadas para balancer (1) o que jogador deve ver, (2) o que o jogador quer ver e (3) o conforto visual.

Cave Story

Fez

Efeito de Paralaxe

O **efeito de paralaxe** faz com que objetos mais distantes se movem mais lentamente do que objetos mais próximos:

Para implementar esse efeito, basta multiplicar a posição da câmera $\vec{C}$ por um fator de paralaxe p:

$$\vec{S} = \vec{P} - p\vec{C}$$

Por exemplo:

- $p = 1.0$ (camada do jogador)
- $p = 0.5$ (camada do meio)
- $p = 0.25$ (camada do fundo)

Camera Shake

O **chacoalhamento (*shake*) de câmera** é uma técnica usada para transmitir impacto (explosão, colisão, dano, etc.) adicionando pequenas variações temporárias na posição da câmera

Para implementar esse efeito, basta somar uma deslocamento $\Delta\mathbf{C}_k$ à posição original da câmera $\overrightarrow{C}$, para cada quadro k que o efeito estiver ativo:

$$\vec{S} = \overrightarrow{P} - p\overrightarrow{C} + \Delta\mathbf{C}_k$$

Deslocamento aleatório:

$$\Delta\mathbf{C}_k = \begin{bmatrix} r_x^{(k)} \\ r_y^{(k)} \end{bmatrix}, \quad r_x, r_y \sim U(-\alpha, \alpha)$$

Camera Shake

O **chacoalhamento (*shake*) de câmera** é uma técnica usada para transmitir impacto (explosão, colisão, dano, etc.) adicionando pequenas variações temporárias na posição da câmera.

Deslocamento oscilatório:

$$\Delta \mathbf{C}(t) = \begin{bmatrix} A(t)\ \sin(\omega_x t) \\ A(t)\ \cos(\omega_y t) \end{bmatrix}$$

- $A(t)$ = amplitude decrescente com o tempo

$$A(t) = \alpha \cdot \left(1 - \frac{t}{T}\right), 0 \leq t \leq T$$

  - α = intensidade inicial
  - T = duração do efeito
  - ω_x e ω_y = frequências em cada eixo

Próxima aula

A13: Game Design

- Definições de Jogo
- Mecânicas e Dinâmicas
- Técnicas de Design
- Equipes de Desenvolvimento
- Playtesting