

DCC192

2025/2



Desenvolvimento de Jogos Digitais

A25: Efeitos de Pós-Processamento

Prof. Lucas N. Ferreira

Plano de Aula



- ▶ Efeitos Comuns
- ▶ Filtros
- ▶ Convoluções
- ▶ Framebuffer Auxiliar

Efeitos de Pós-Processamento



Efeitos de Pós-processamento são filtros e efeitos aplicados às imagens produzidas por cada quadro do jogo, antes de serem mostradas na tela. Por exemplo:

- ▶ **Ajuste de Cores**

- ▶ Depth of Field
- ▶ Grain
- ▶ Bloom
- ▶ Motion Blur
- ▶ Anti-aliasing
- ▶ ...



Ajustes de cores como Temperatura, Tint, Saturação, etc.

Efeitos de Pós-Processamento



Efeitos de Pós-processamento são filtros e efeitos aplicados às imagens produzidas por cada quadro do jogo, antes de serem mostradas na tela. Por exemplo:

- ▶ Ajuste de Cores
- ▶ **Depth of Field**
- ▶ Grain
- ▶ Bloom
- ▶ Chromatic Aberration
- ▶ Motion Blur
- ▶ ...



Simula foco de uma câmera real: focando ou desfocando os objetos que estiverem a uma certa distância das lentes

Efeitos de Pós-Processamento



Efeitos de Pós-processamento são filtros e efeitos aplicados às imagens produzidas por cada quadro do jogo, antes de serem mostradas na tela. Por exemplo:

- ▶ Ajuste de Cores
- ▶ Depth of Field
- ▶ **Grain**
- ▶ Bloom
- ▶ Chromatic Aberration
- ▶ Motion Blur
- ▶ ...



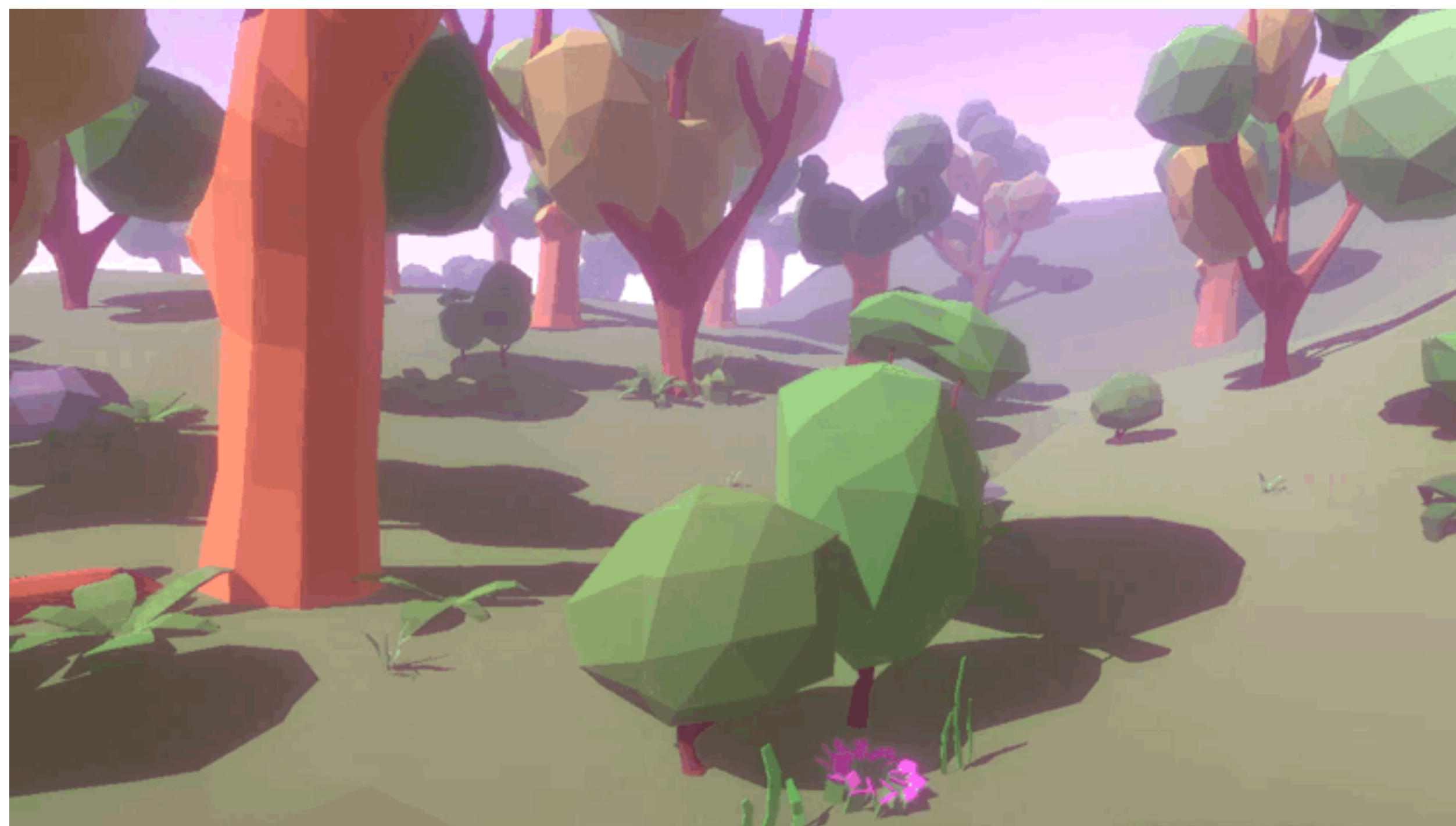
Simula o efeito que as câmeras real produzem quando pequenas partículas na câmera do filme dão à imagem um efeito grosseiro e não processado

Efeitos de Pós-Processamento



Efeitos de Pós-processamento são filtros e efeitos aplicados às imagens produzidas por cada quadro do jogo, antes de serem mostradas na tela. Por exemplo:

- ▶ Ajuste de Cores
- ▶ Depth of Field
- ▶ Grain
- ▶ **Bloom**
- ▶ Chromatic Aberration
- ▶ Motion Blur
- ▶ ...



Deixa os pontos de luz mais brilhantes e levemente “borrados”

Efeitos de Pós-Processamento



Efeitos de Pós-processamento são filtros e efeitos aplicados às imagens produzidas por cada quadro do jogo, antes de serem mostradas na tela. Por exemplo:

- ▶ Ajuste de Cores
- ▶ Depth of Field
- ▶ Grain
- ▶ Bloom
- ▶ **Chromatic Aberration**
- ▶ Motion Blur
- ▶ ...



Cria espectros de luz ao redor dos objetos, é mais utilizada para efeitos de intoxicação do player

Filtros



Um **filtro** (ou **kernel**) k é uma pequena matriz de pesos, tipicamente 3x3, usada para transformar um **pixel** pela soma ponderada dos pixels da sua vizinhança m .

	206	205	247	
	144	161	137	
	192	154	75	

Pixel original (161) e
seus vizinhos

*

0,0625	0,125	0,0625
0,125	0,25	0,125
0,0625	0,0125	0,0625

Filtro (blur)

$$= \sum_{i=1}^3 \sum_{j=1}^3 = m_{i,j} * k_{i,j} =$$

$$206 * 0.0625 + 205 * 0.125 + 247 * 0.0625 + \\ 144 * 0.125 + 161 * 0.25 + 137 * 0.125 + \\ 192 * 0.0625 + 154 * 0.125 + 75 * 0.0625 =$$

178

	206	205	247	
	144	178	137	
	192	154	75	

Pixel Transformed (178) e
seus vizinhos

Convoluções



Convolução é a operação que aplica um filtro k de tamanho f a cada pixel de uma imagem m , começando em $(\lfloor f/2, f/2 \rfloor)$ e indo até $(\lfloor w - f/2, h - f/2 \rfloor)$, de cima para baixo, da esquerda para a direita.

206	205	247	245	244
244	161	137	244	254
192	154	75	200	249
90	109	96	143	223
67	69	107	196	236

Imagem Original
(5 × 5)

0,0625	0,125	0,0625
0,125	0,25	0,125
0,0625	0,0125	0,0625

Filtro (blur)
 $f = 3$

$$= \sum_{i=1}^3 \sum_{j=1}^3 = m_{i,j} * k_{i,j} =$$

178	175	216
141	133	183
106	117	167

Imagem Transformada
(3 × 3)

Tamanho da imagem resultante



Uma convolução em um imagem original de tamanho $(n \times n)$ produz uma imagem transformada com tamanho reduzido, que pode ser calculado usando a regra geral abaixo:

206	205	247	245	244
244	161	137	244	254
192	154	75	200	249
90	109	96	143	223
67	69	107	196	236

Imagem Original
(5 × 5)

*

Filtro
(3 × 3)

=

178	175	216
141	133	183
106	117	167

Imagem Transformada
(3 × 3)

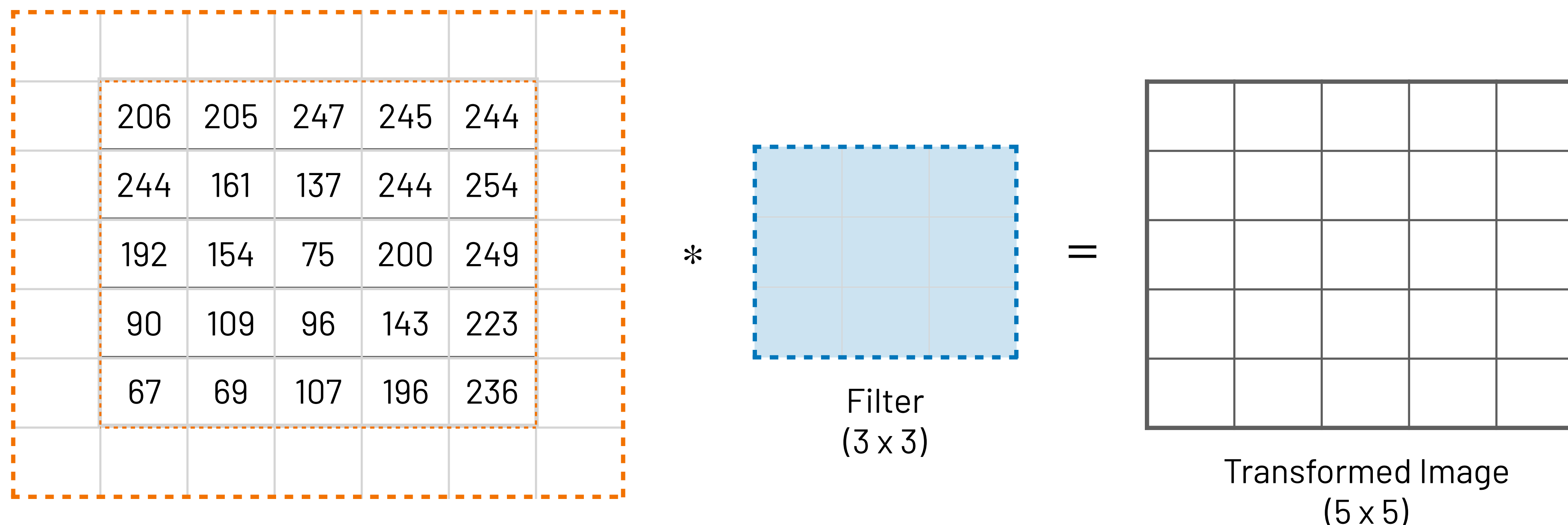
Regra Geral

$$(n \times n) * (f \times f) = (n - f + 1 \times n - f + 1)$$

Preenchimento (Padding)



Preenchimento (*padding*) consiste em adicionar uma **borda** com p pixels a imagem original. Com padding, podemos manter o tamanho original da imagem após uma convolução:



Regra Geral com Padding

$$(n \times n) * (f \times f) = (n + 2p - f + 1 \times n + 2p - f + 1)$$

Preenchimento (*Padding*)



Para encontrar o valor de p que mantém o tamanho $n \times n$ original de uma imagem depois de uma convolução com filtro de tamanho f (ímpar), podemos resolver a seguinte equação:

$$n + 2p - f + 1 = n$$

$$2p - f + 1 = 0$$

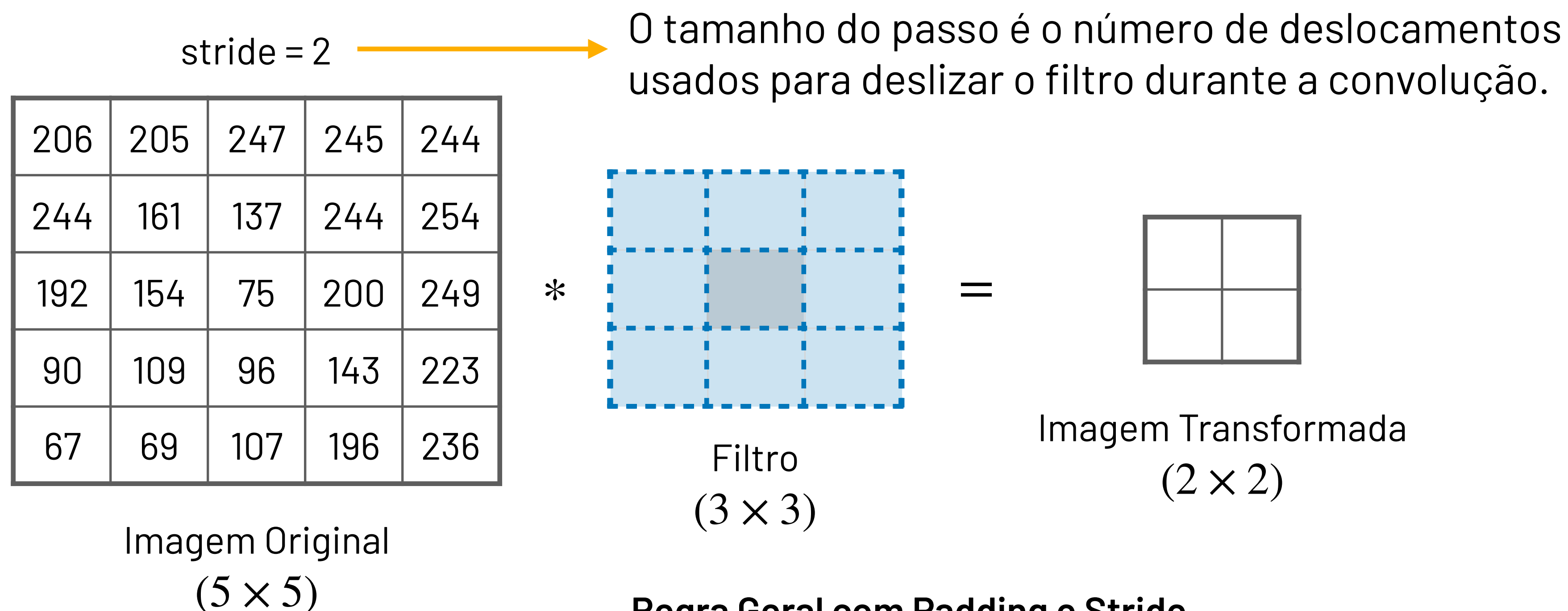
$$2p = f - 1$$

$$p = \frac{f - 1}{2}$$

Convoluções com Passo (*stride*)



Convoluções com passo deslizam o filtro mais de um passo por vez:



Regra Geral com Padding e Stride

$$(n \times n) * (f \times f) = \left(\frac{n + 2p - f}{s} + 1 \right) \times \left(\frac{n + 2p - f}{s} + 1 \right)$$

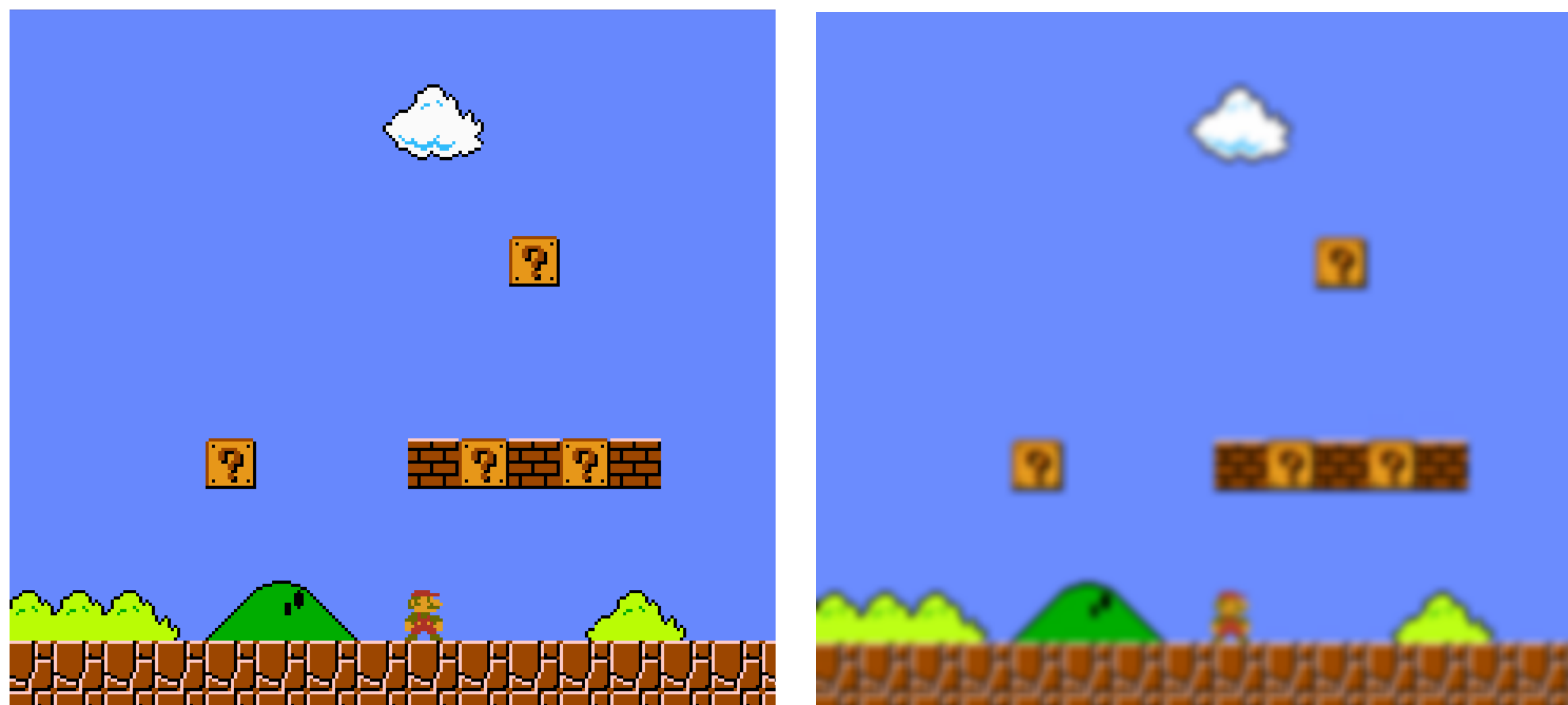
Exemplos de Filtros



Diferentes filtros foram desenvolvidos cientificamente pela comunidade de pesquisa em processamento de imagens para alcançar resultados visuais diferentes.

0,0625	0,125	0,0625
0,125	0,25	0,125
0,0625	0,0125	0,0625

Blur



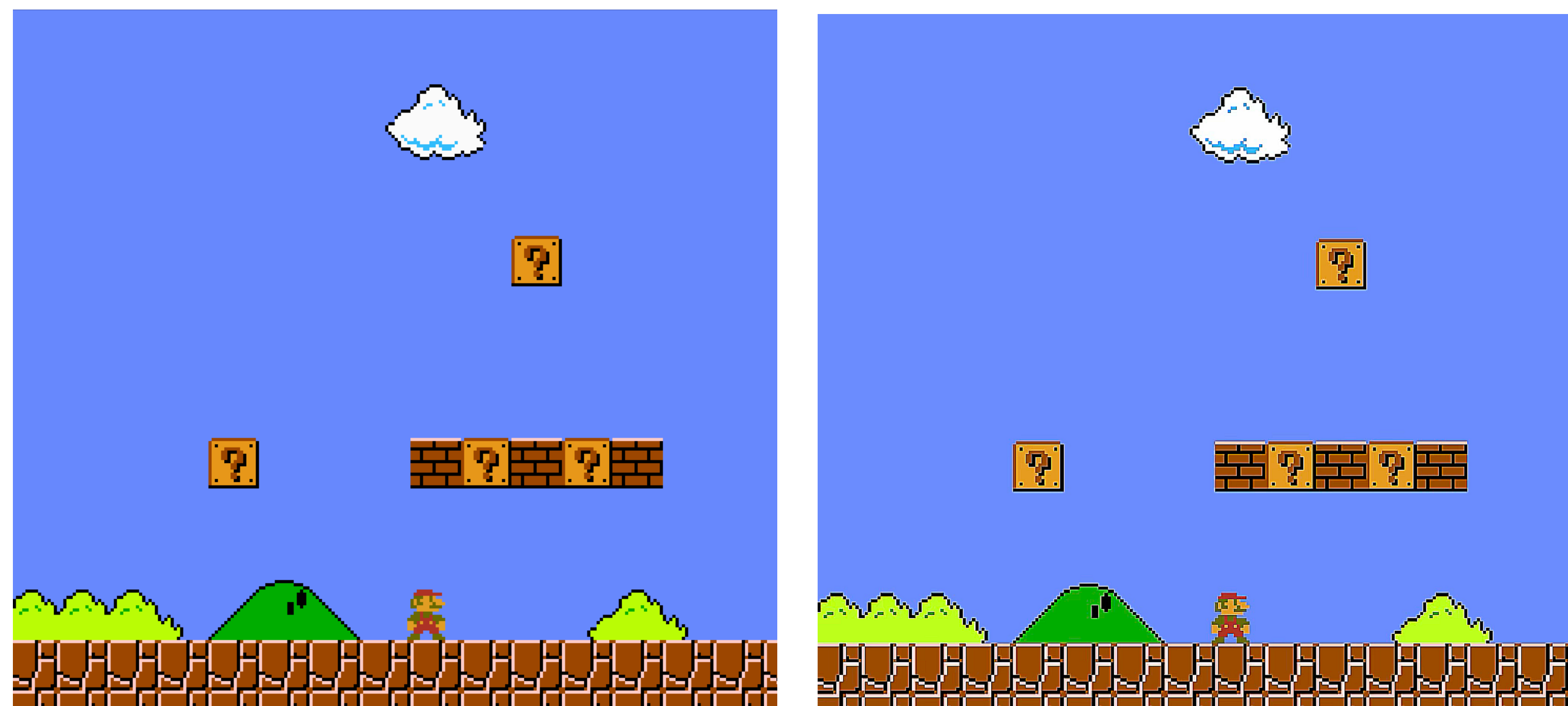
Exemplos de Filtros



Diferentes filtros foram desenvolvidos cientificamente pela comunidade de pesquisa em processamento de imagens para alcançar resultados visuais diferentes.

1	0	-1
2	0	-2
1	0	-1

Sharpen



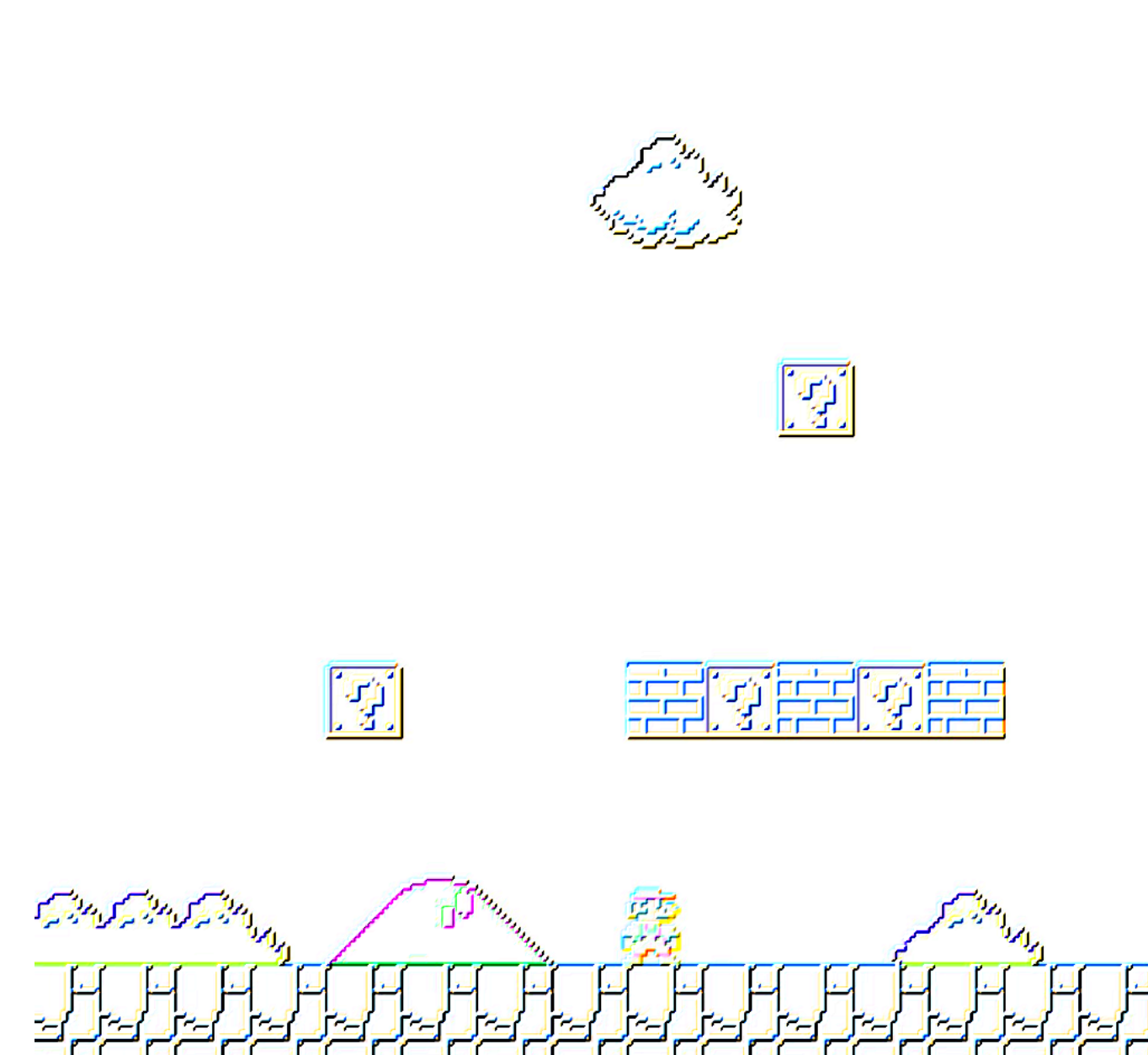
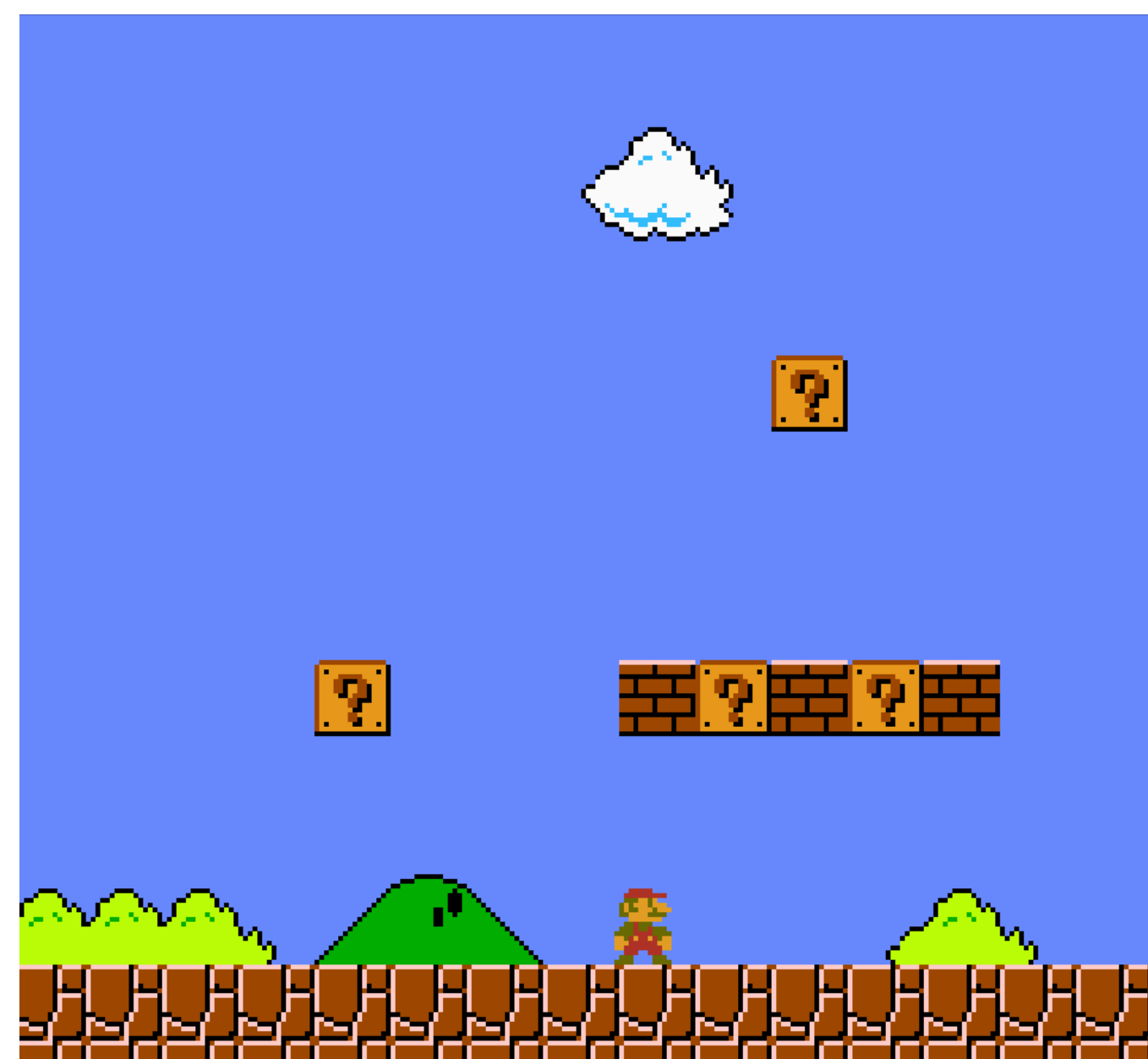
Exemplos de Filtros



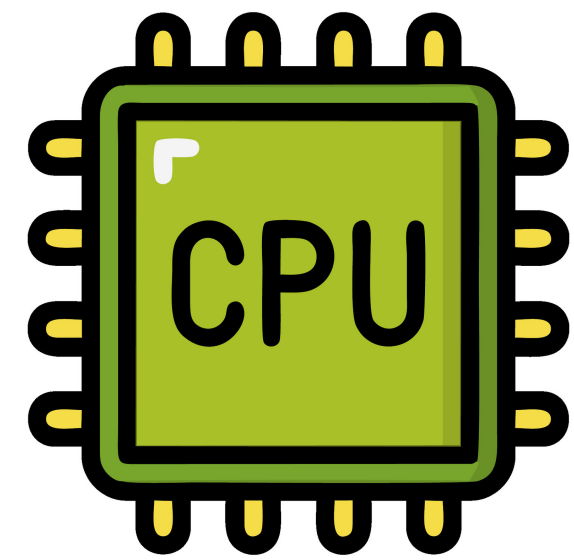
Diferentes filtros foram desenvolvidos cientificamente pela comunidade de pesquisa em processamento de imagens para alcançar resultados visuais diferentes.

0	-1	0
-1	5	-1
0	-1	0

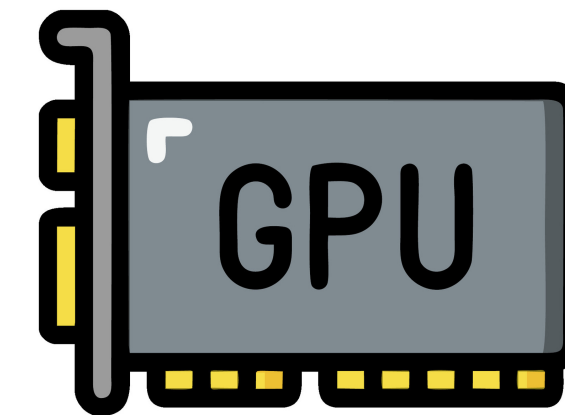
Detecção de Bordas
(Sobel)



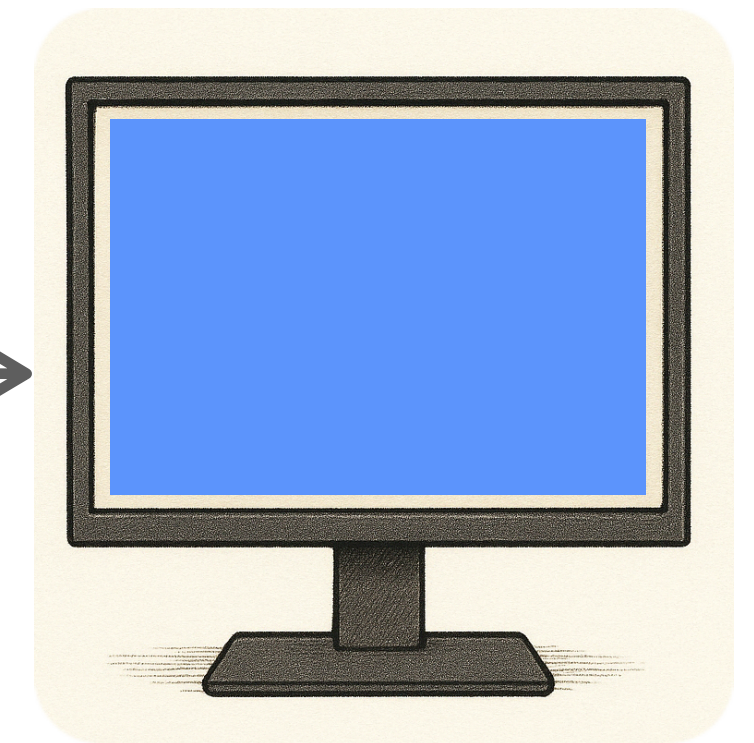
Framebuffer Auxiliar



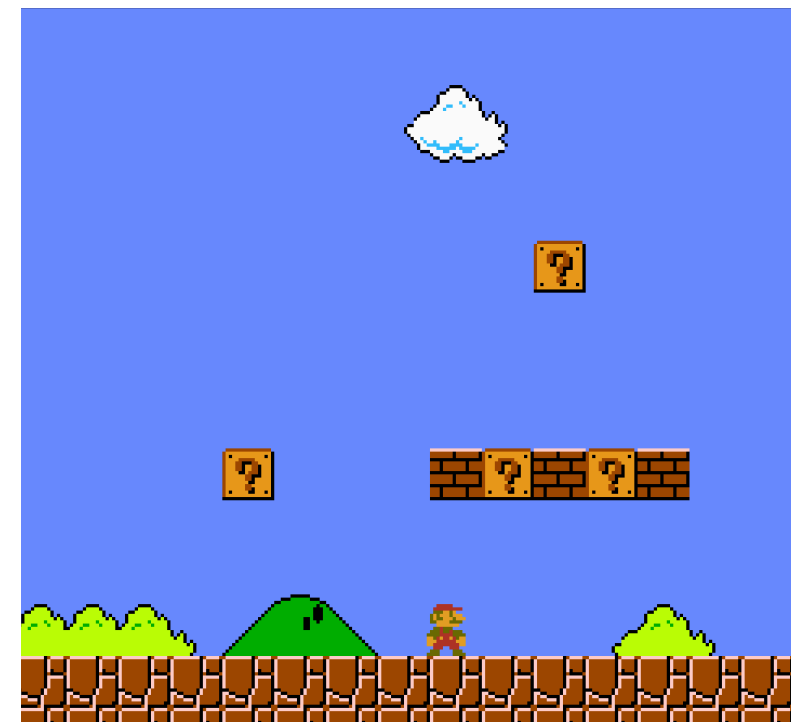
```
while (mIsRunning) {  
    ProcessInput();  
    UpdateGame(deltaTime);  
    GenerateOutput();  
}
```



Monitor



1. Ao invés de renderizar a cena direto para o framebuffer, vamos renderizá-la em um buffer auxiliar



Buffer Auxiliar

2. Aplicamos um filtro ou efeito à imagem renderizada no buffer auxiliar

(Ex. Blur)



Framebuffer

3. Movemos a imagem com filtro para o frame buffer,

Próxima aula



A26: Prática