INF216

2023/2

UFV

Projeto e Implementação de Jogos Digitais

A6: Gráficos 2D

Logística

Avisos

- Os projetos P4 e P5 trocaram de ordem. O próximo projeto será o Super Mario Bros

Última aula

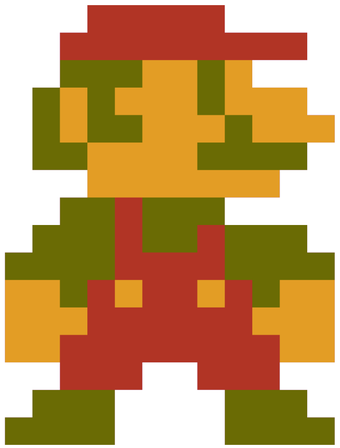
- Geomtrias de Colisão
- Detecção de Colisão
- Resolução de Colisão

Plano de Aula

- Sprites
- Animações
- Sprite Sheets
- Mapas de blocos (*Tilemaps*)
- Rolagem de Câmera

Sprite

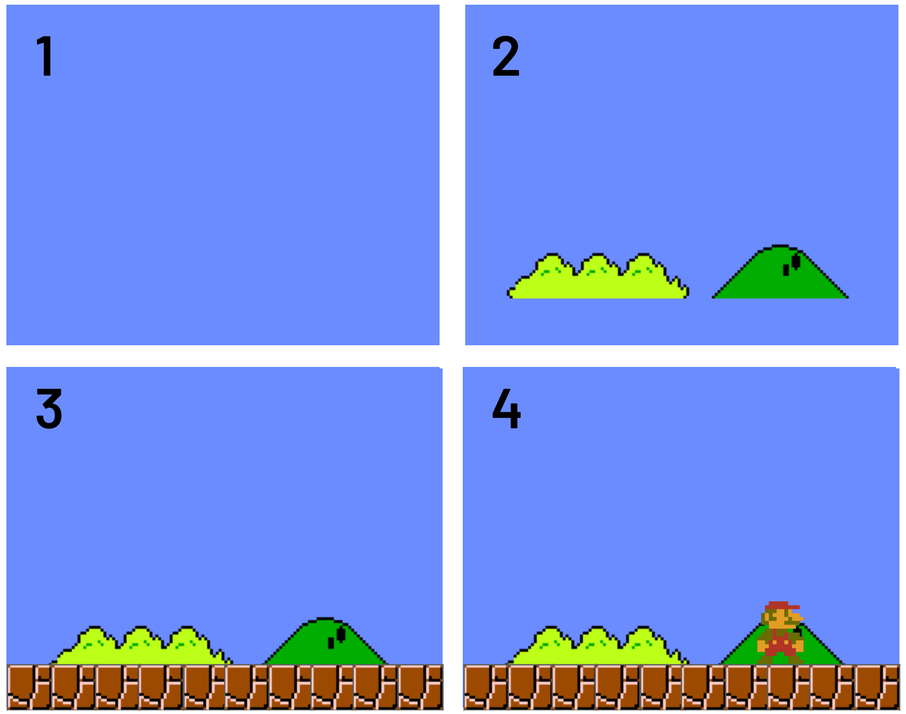
Um **sprite** é um objeto 2D que pode ser desenhado usando uma única imagem em qualquer quadro do jogo.



```
class Sprite {
    Vector2 position;
    int drawOrder;
    void Draw();
}
```

Desenhando Sprites

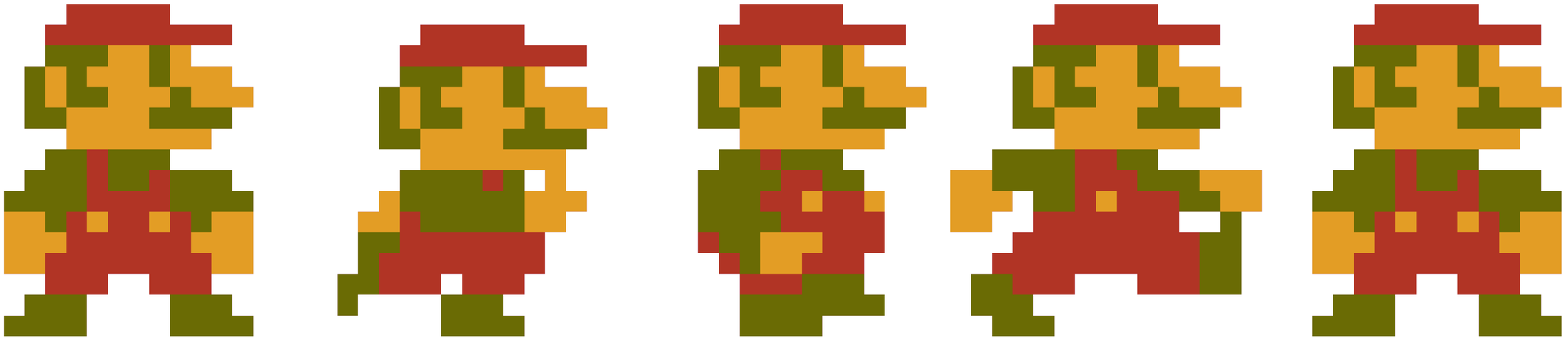
Algoritmo do pintor: manter uma lista ordenada de sprites e desenhá-la de trás pra frente.




```
SortedList spriteList

// When creating a new sprite...
Sprite newSprite = specify image and desired x/y
newSprite.drawOrder = set desired draw order value

// Add to sorted list based on draw order value
spriteList.Add(newSprite.drawOrder, newSprite)

// When it's time to draw...
foreach Sprite s in spriteList
  s.Draw()
```

Animação

Uma série de imagens estáticas 2D reproduzidas em rápida sucessão para criar uma ilusão de movimento.



Armezando Animações

Manter uma lista de imagens com todos os quadros de um personagem:

0 1 2 3 4 5 6 7 8

Manter uma lista com os indices dos quadros de cada animação do personagem:

Idle	[0]
Jump	[1]
Run	[2, 3, 4]
Stomp	[5, 6]
Turn	[7]
Dead	[8]

Tocando Animações

Não podemos assumir que a taxa de quadros da animação seja mais lenta que a taxa de quadros do jogo

- FPS do Jogo: 30
- FPS de uma animação com 24 quadros: 48

Isso significa que muitas vezes precisaremos pular vários quadros na animação.

Tocando Animações

Precisamos de dois floats para tocar uma animação:

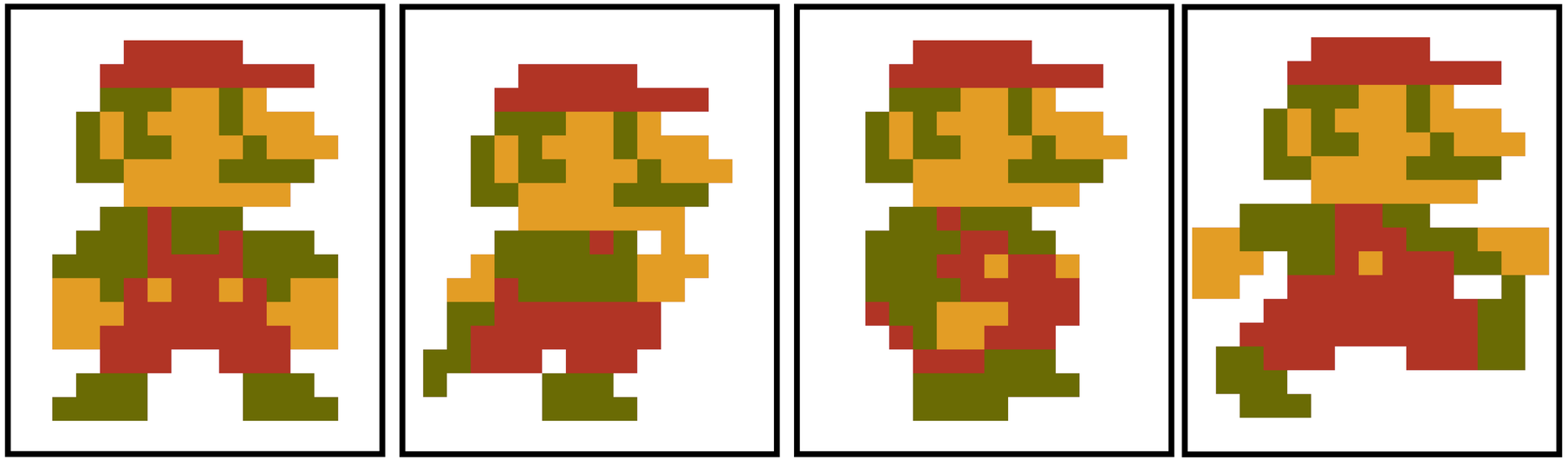
- `AnimTimer`, para armazenar o tempo corrente da animação
- `AnimFPS`, para amazenar a taxa de atualização da animação

Transformar (cast) `AnimTimer` para inteiro para acessar o índice da animação

```
frameTime += deltaTime

// verificar se está na hora de alterar o sprite
if frameTime > (1 / animFPS) {
   animTimer = frameTime * animFPS // frameTime / (1 / animFPS) -> frameTime * animFPS
   if animTimer >= animData.frameInfo[animNum].numFrames:
      animTimer = (int)animTimer % animData.frameInfo[animNum].numFrames
}

int imageNum = animData.frameInfo[animNum].startFrame + frameNum
```

Sprite Sheet

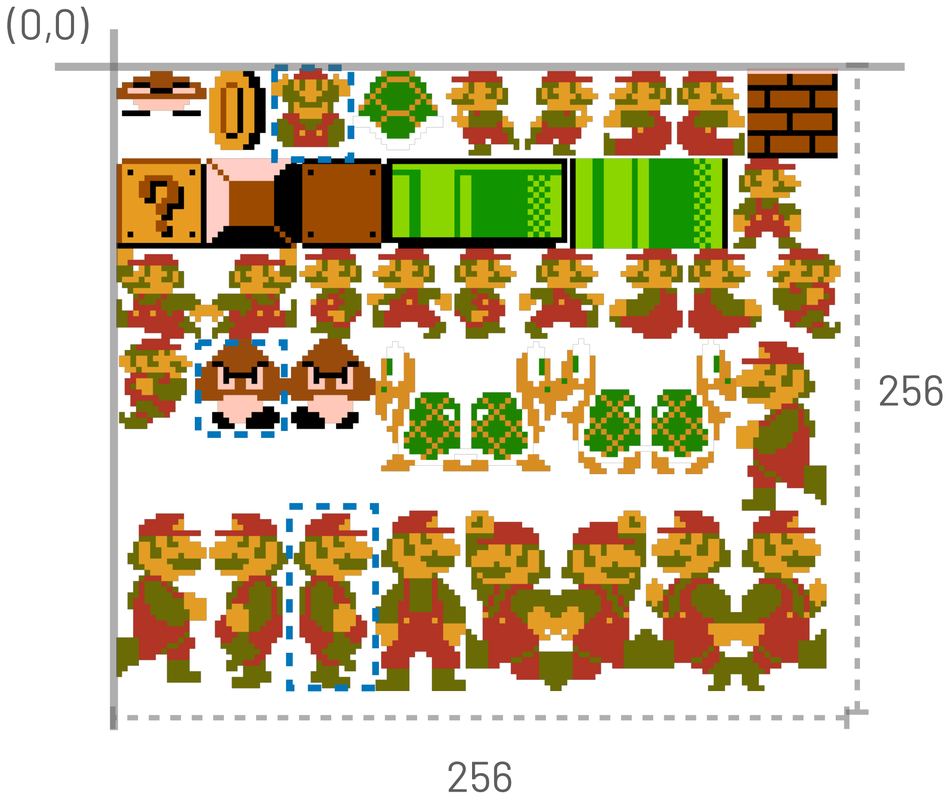
Armazenar sprites em arquivos separados pode acabar desperdiçando muita memória e processamento (considerando sprites com imagens de tamanhos iguais).



100 sprites de 100Kb, 15% área branca/sprite, 1.5MB de desperdício

Sprite Sheet

Compactar todos os sprites em uma única imagem (geralmente com dimensões em potência de 2), onde cada sprite é descrito por um retângulo.



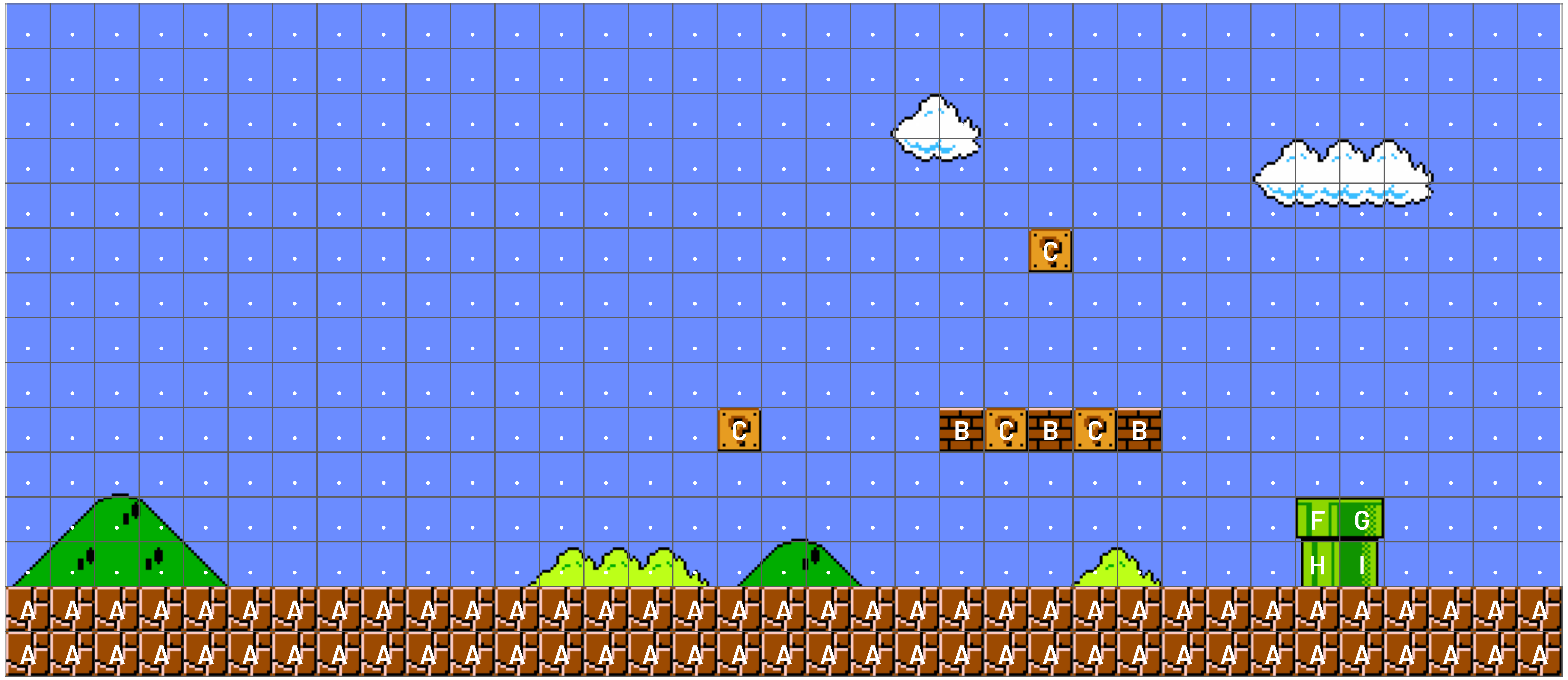
{x: 64, y: 0, w: 32, h: 32}

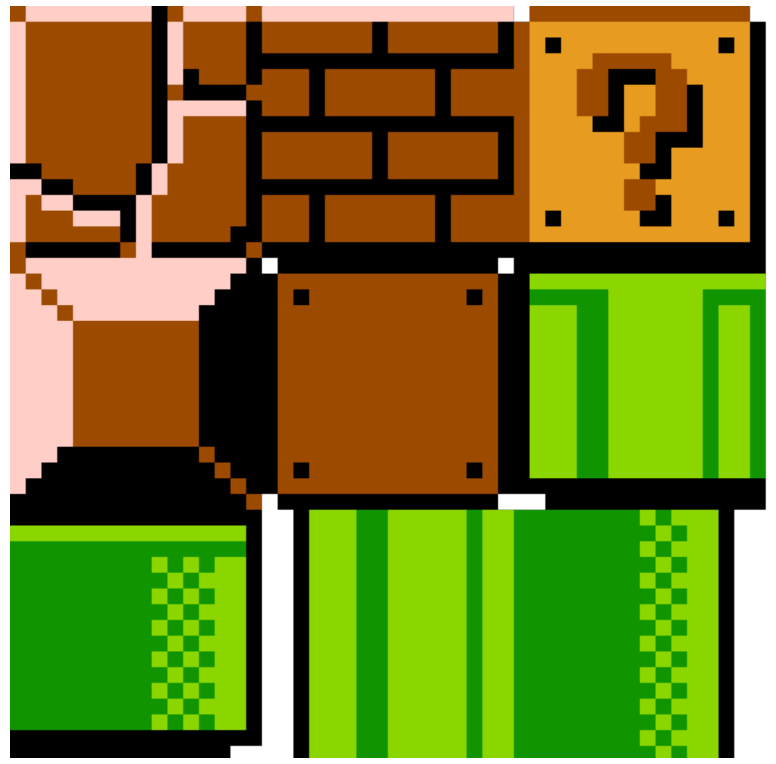
{x: 32, y: 128, w: 32, h: 32}

{x: 96, y: 192, w: 32, h: 96}

Tilemaps

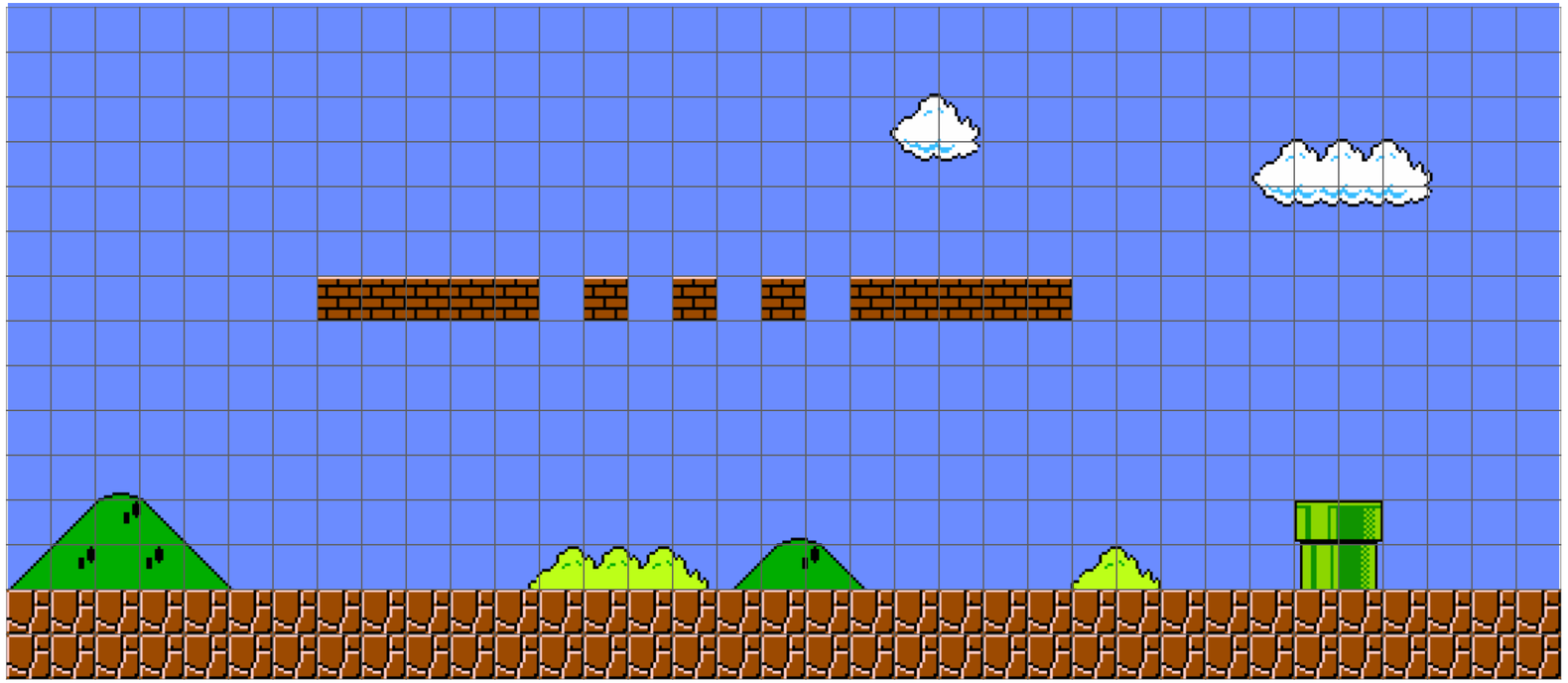
Tilemap é uma forma de organizar níveis (levels) em uma grade (grid) com partes quadradas de tamanhos iguais (tiles), cada um com número identificador, visando maximizar os tiles que se repetem.

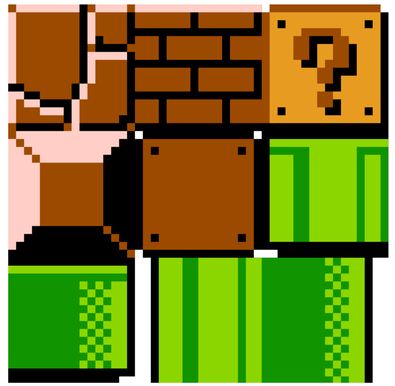




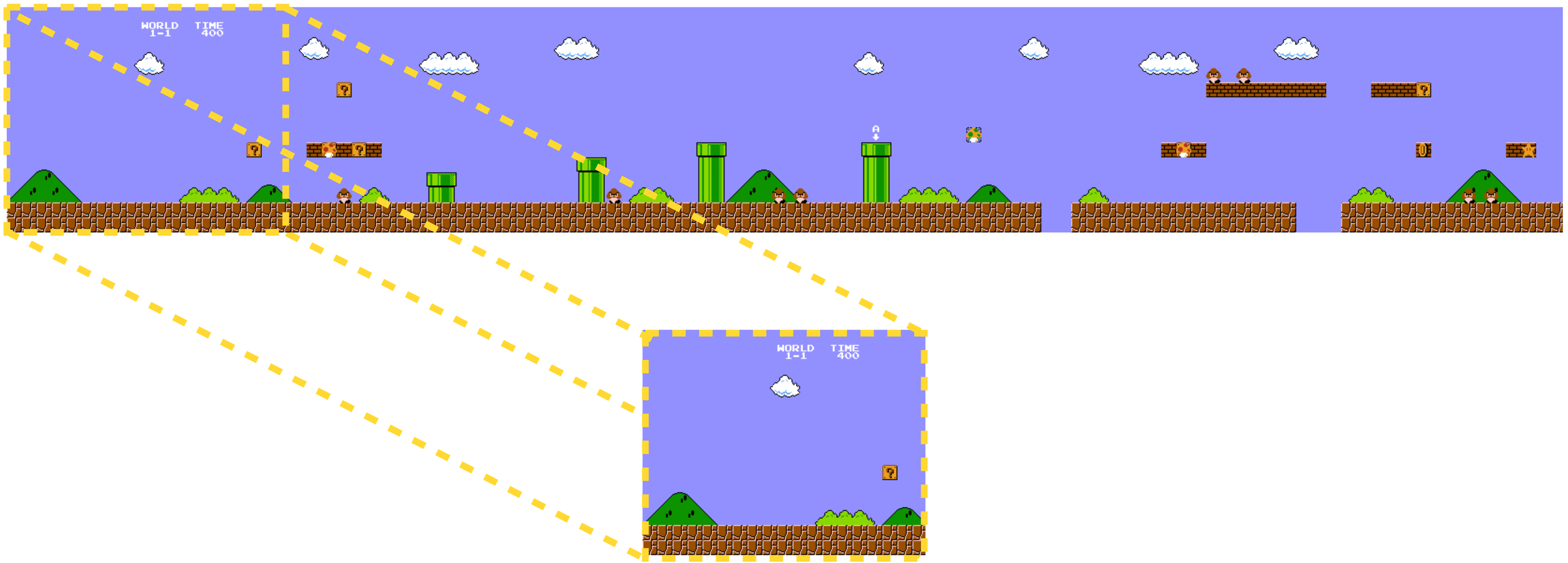
Tilemaps

A estrutura de grade (grid) dos tilemaps facilita a editação de níveis, pois a posição dos tiles é limitado a coordenadas discretas.

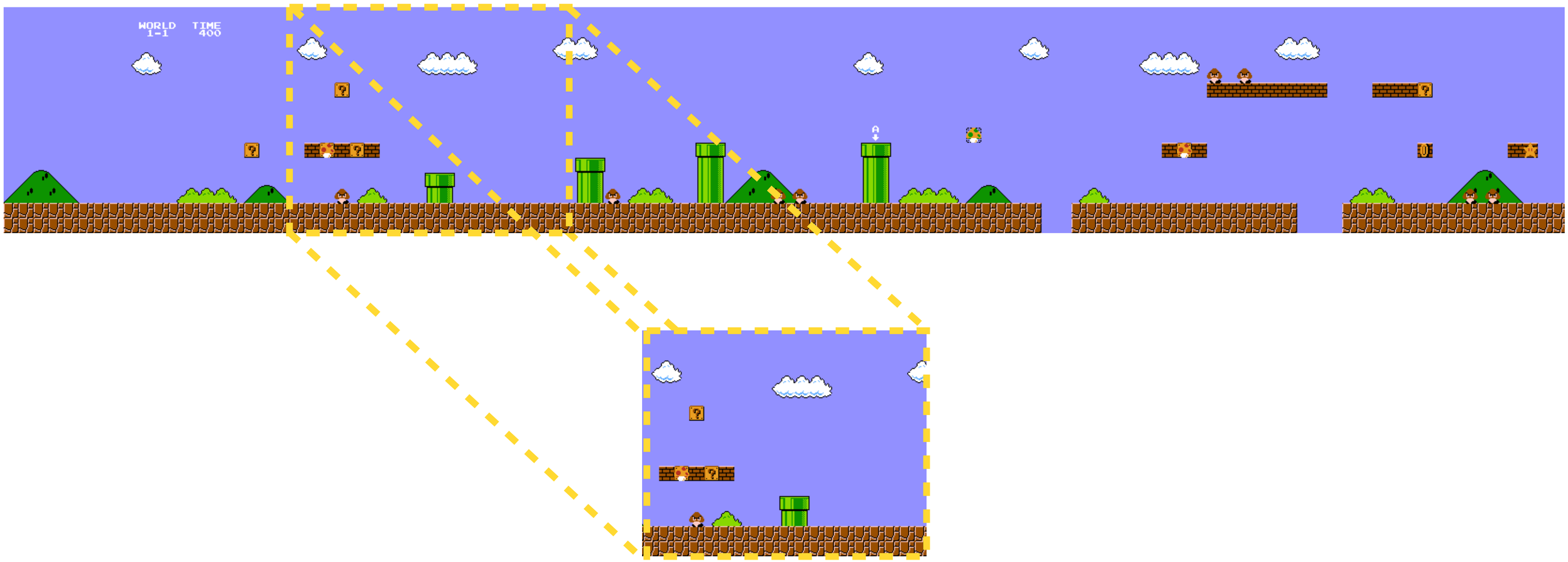




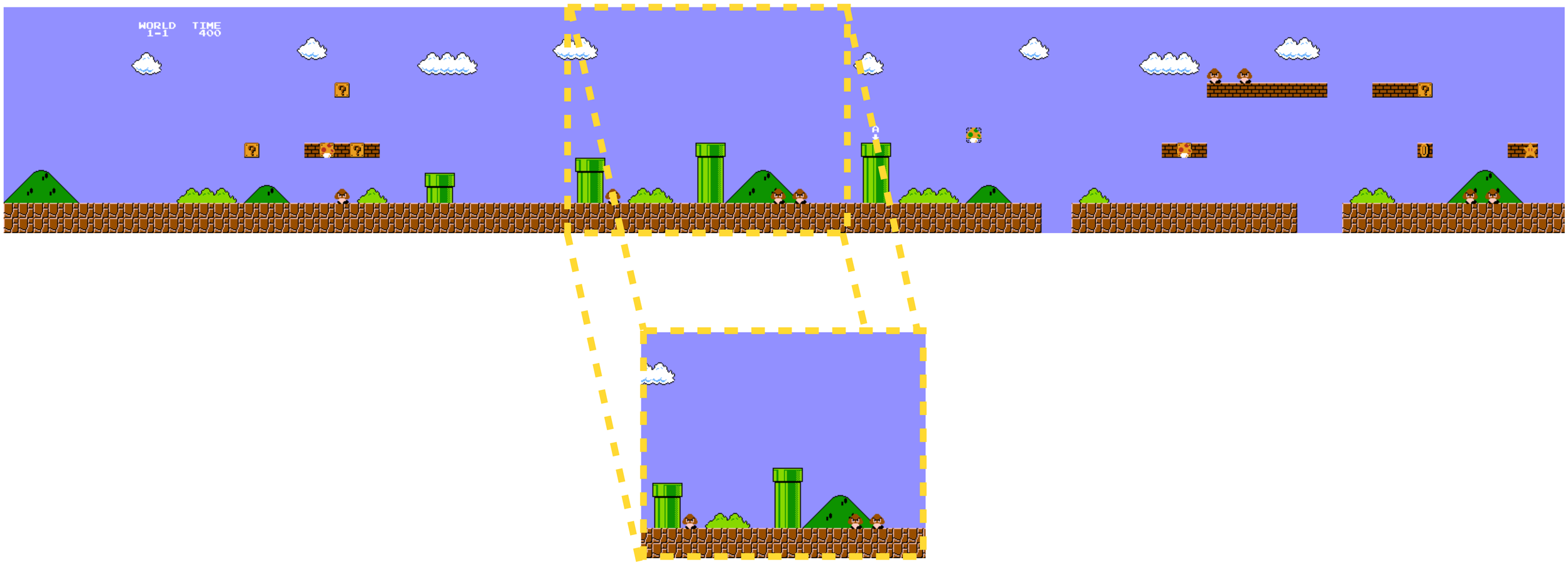
Rolagem de Câmera



Rolagem de Câmera

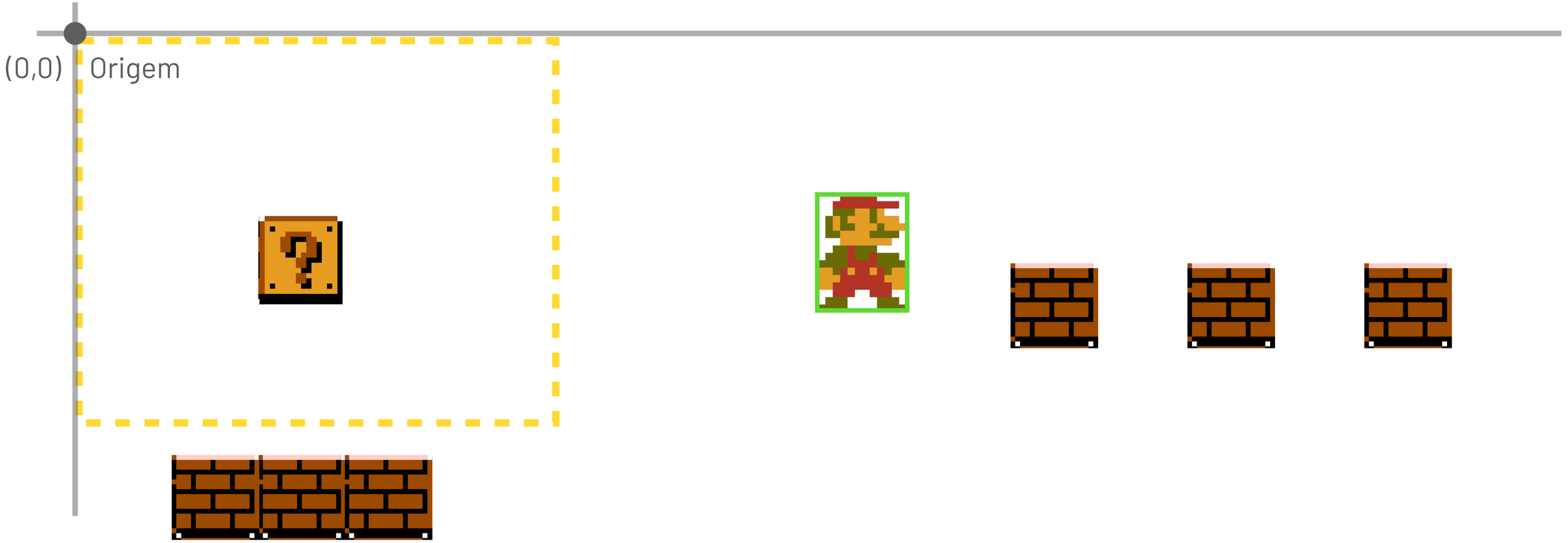


Rolagem de Câmera



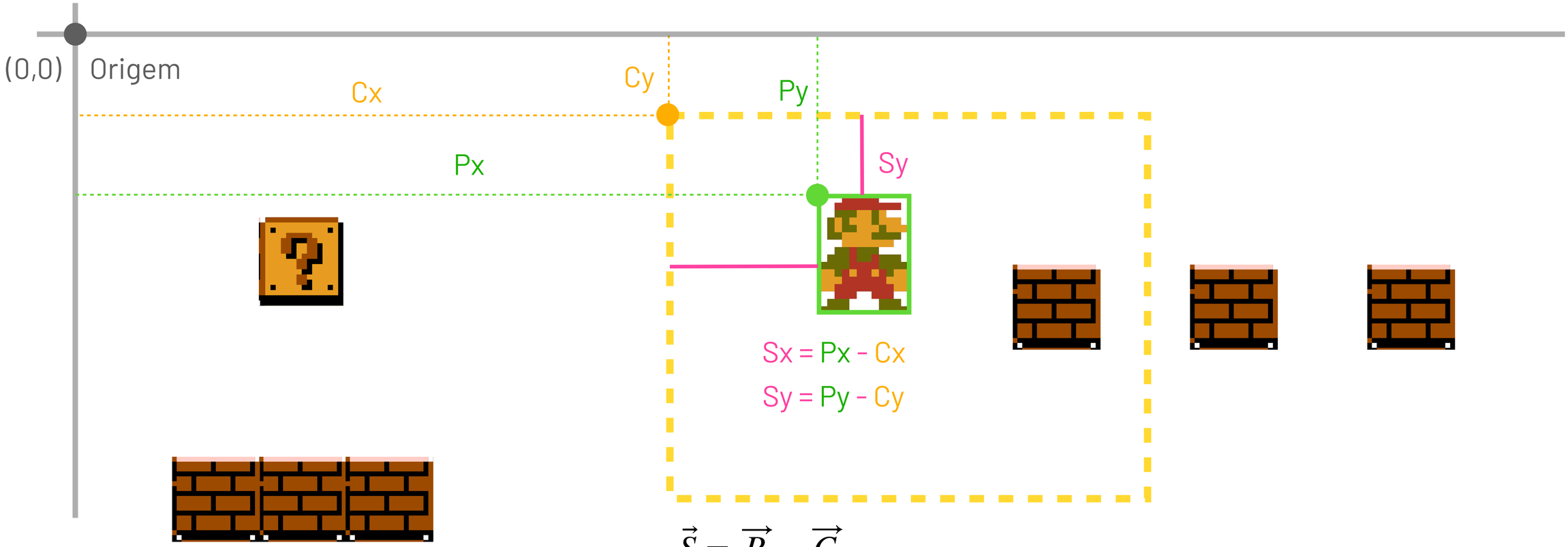
Rolagem de Câmera

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!



Rolagem de Câmera

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



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

screenPosition = worldPosition - cameraPosition

Efeito Parallax

Para implementar o efeito parallax, basta multiplicar a posição da câmera por um fator de parallax p.

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

Por exemplo:

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

Próximas aulas

A7: Câmeras 2D

Técnicas para rolagem de câmeras em jogos 2D.

L7: Super Mario Bros - Parte 1

Implementar um componente SpriteRenderer para animação 2D com spritesheets.