Geração Procedural de Níveis para o Super Mario Bros. com Algoritmos Genéticos

Lucas Nascimento Ferreira



Sumário

- Procedural Content Generation (PCG)
- Super Mario Bros. (SMB)
- Representação de um nível no SMB
- Função de fitness com Design Patterns
- Algoritmo Genético
- Avaliação dos níveis gerados
- Tutorial

Procedural Content Generation

- PCG é o processo de geração automática ou semi-automática de conteúdo para jogos.
  - Vantagens:
    - Economia de tempo (ex: *Darwinia*)
    - Economia de dinheiro (ex: *Just Cause*)
    - Explorar variações de conteúdo (ex: *The Sentinel*)
    - Criação de conteúdo único (ex: *Minecraft*)
    - Economia de memória (ex: *Elite*)

Procedural Content Generation

- PCG + *Game Design*
  - Geração de mapas em tempo de execução (Civilization, Diablo II)
  - Geração de uma grande variação de armas (Borderlands)
  - Adaptação do cenário para experiência dinâmica (Left for Dead)
  - Criação de cenários naturais (*Just Cause*)

Que Mario?

Super Mario Bros.

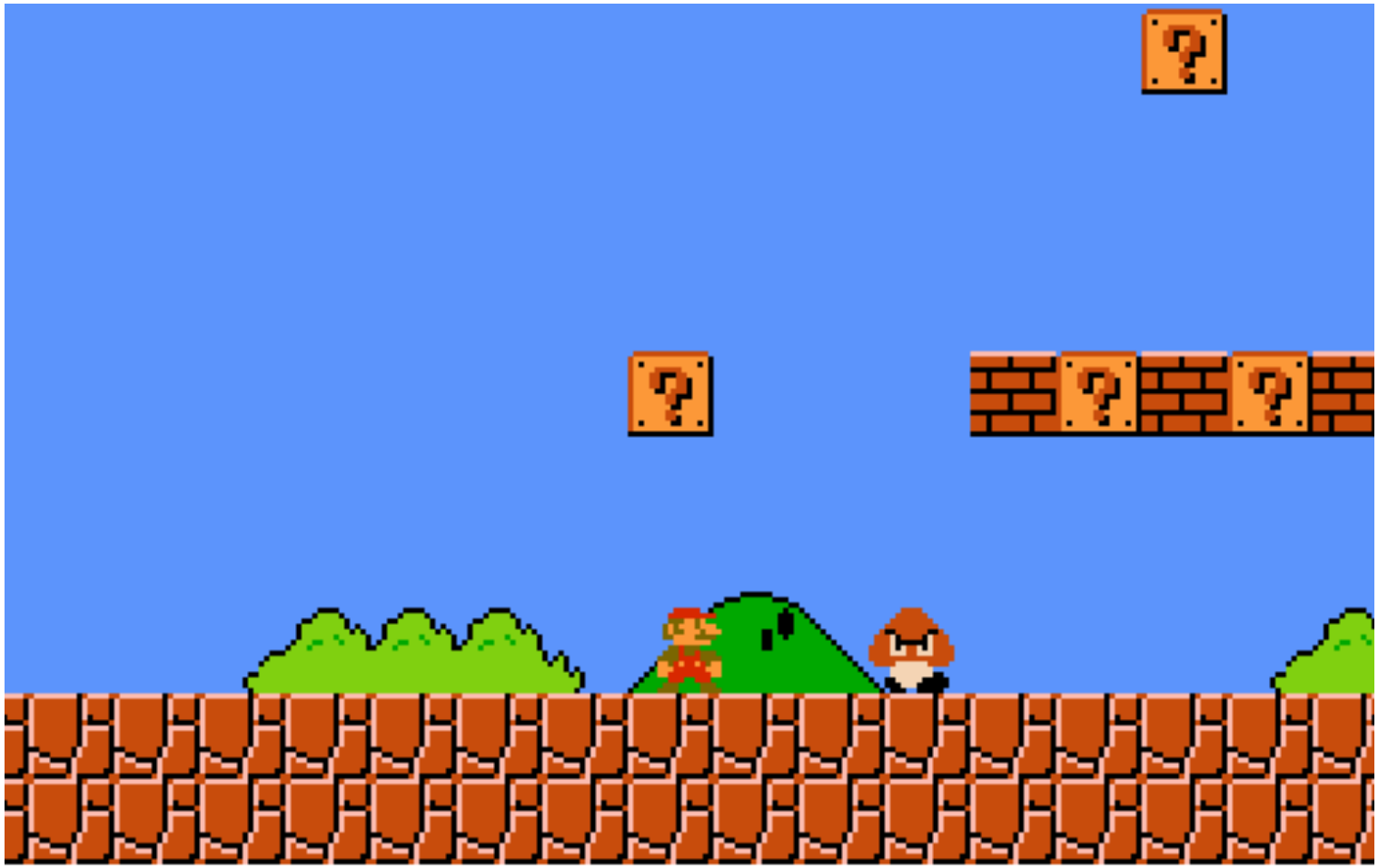


Super Mario Bros.

- Lançado pela Nintendo em 1985
- 8 mundos com 4 níveis cada
- Último nível de cada mundo é uma luta contra o “chefão” (Bowser)

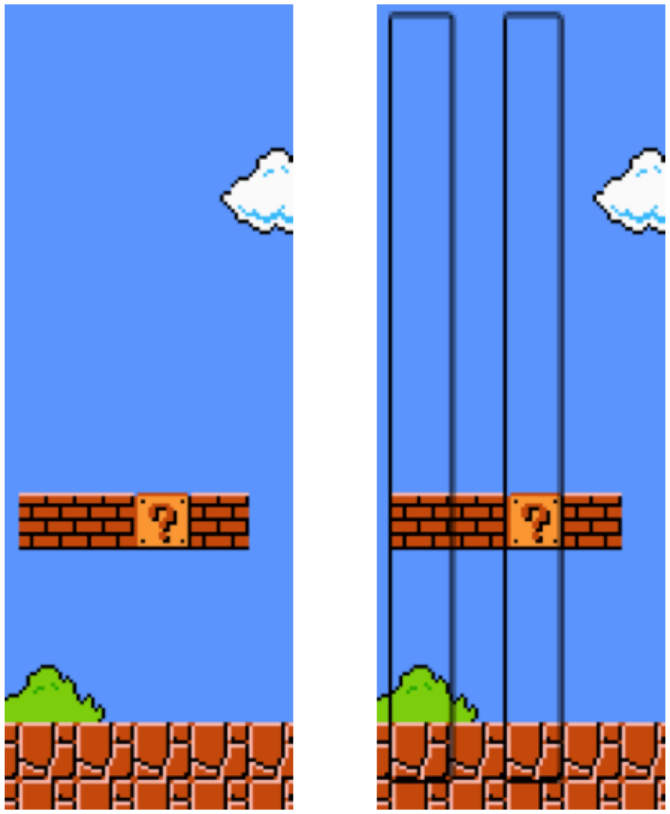


Super Mario Bros.



Representação de um nível no SMB

- Um level pode ser representado por meio de um vetor de ***fatias verticais***



Representação de um nível no SMB

- Uma ***fatia vertical*** tem largura de 1 bloco e altura de 13 blocos
- [*Dahlskog and Togelius, 2013*] utilizaram um vetor com 200 fatias de 24 tipos
- Cada tipo de fatia é representada por uma letra do alfabeto

Design Patterns

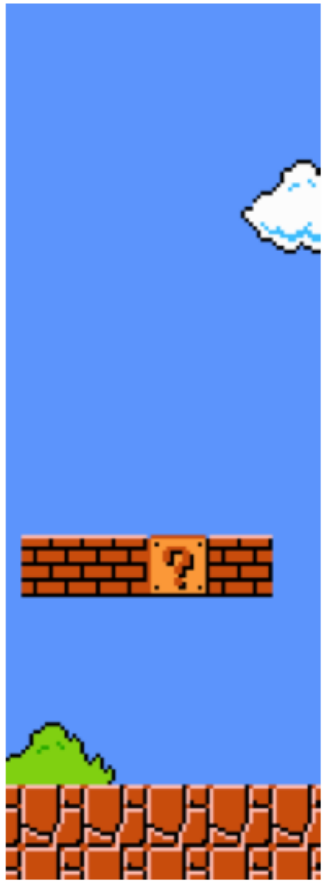
- A ideia de padrões possibilita a utilização de soluções pré-definidos para resolver problemas recorrentes. [Alexander et al., 1977]
- [Gamma et al., 1994] propuseram um conjunto de padrões para soluções de problemas comums de design de POO

Game Design Patterns

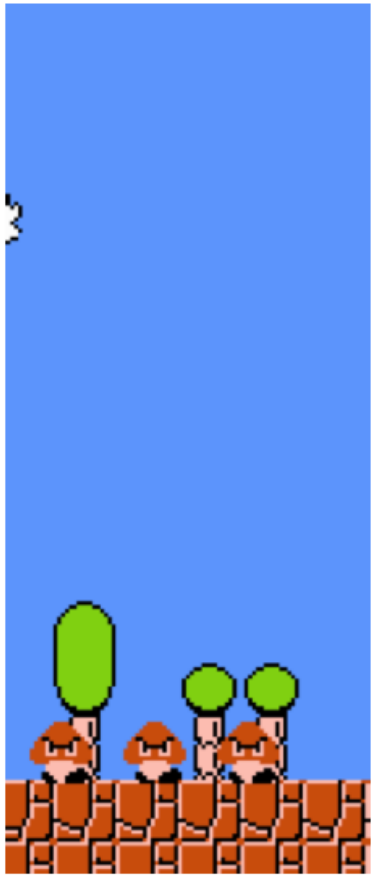
- *Design Patterns* em jogos podem ser usados por diferentes motivos:
  - Inspiração criativa
  - Compartilhar conhecimento entre designers
  - Ferramenta de análise de game design
  - Ferramenta educativa para ensino de game design



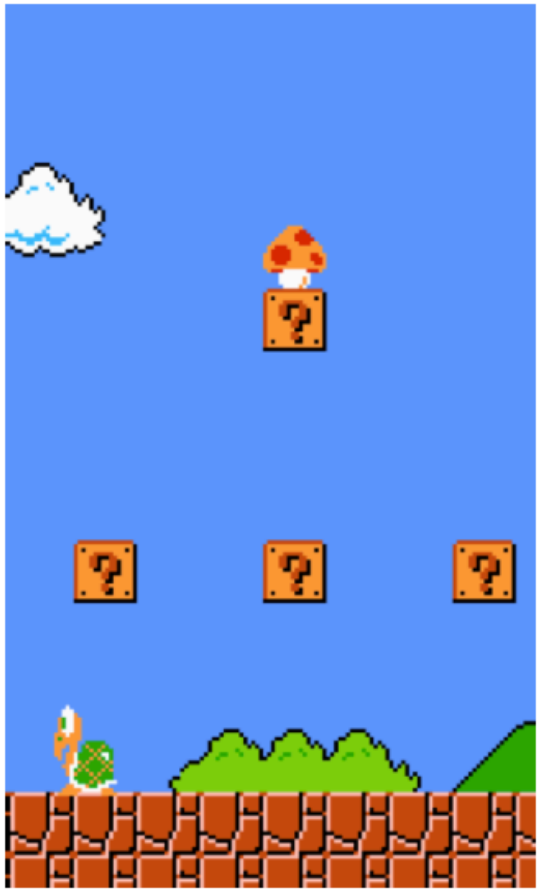
SMB Design Patterns



2-path pattern



3-horde pattern



3-pathpattern

Função de fitness com Design Patterns

- Busca linear em cada indivíduo por padrões pré-definidos
- Cada padrão possui um valor qualitativo (baixo, médio, alto, …)
- Quanto maior a qualidade dos padrões encontrados, maior o ***fitness*** do indivíduo
- [*Dahlskog and Togelius, 2013*] propuseram e utilizaram 23 padrões

Função de fitness com Design Patterns

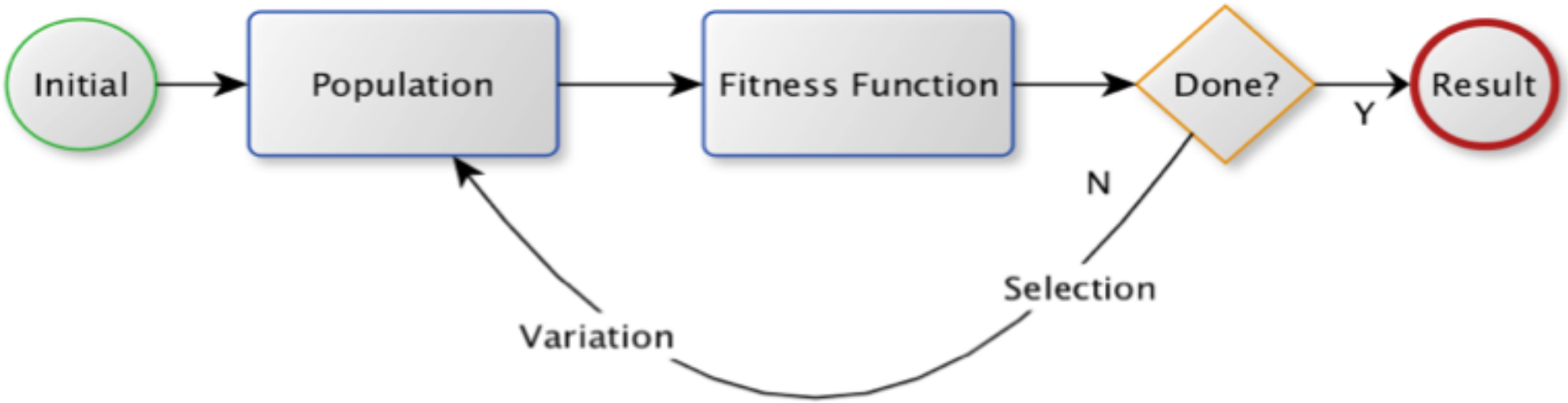
Enemies	
Enemy	Low
2-Horde	Low
3-Horde	Low
4-Horde	Low
Roof	Medium
Gaps	
Gaps	Low
Multiple gaps	By stacking
Variable gaps	By stacking
Gap enemy	Low–Medium by stacking
Pillar gap	Pillar High
Valleys	
Valley	Low
Pipe valley	Medium
Empty valley	By stacking
Enemy valley	By stacking
Roof valley	By stacking
Multiple paths	
2-Path	Medium–High
3-Path	Medium–High
Risk and Reward	By stacking
Stairs	
Stair up	Low
Stair down	Low
Empty stair valley	Low
Enemy stair valley	By stacking
Gap stair valley	By stacking

Função de fitness com Design Patterns

- Alguns padrões são suportados apenas por empilhamento
- Sequencias com jogabilidade impossível atribuem um valor negativo muito alto ao fitness
- Sequencias com jogabilidade desinteressante atribuem um valor negativo baixo ao fitness

Algoritmo Genético

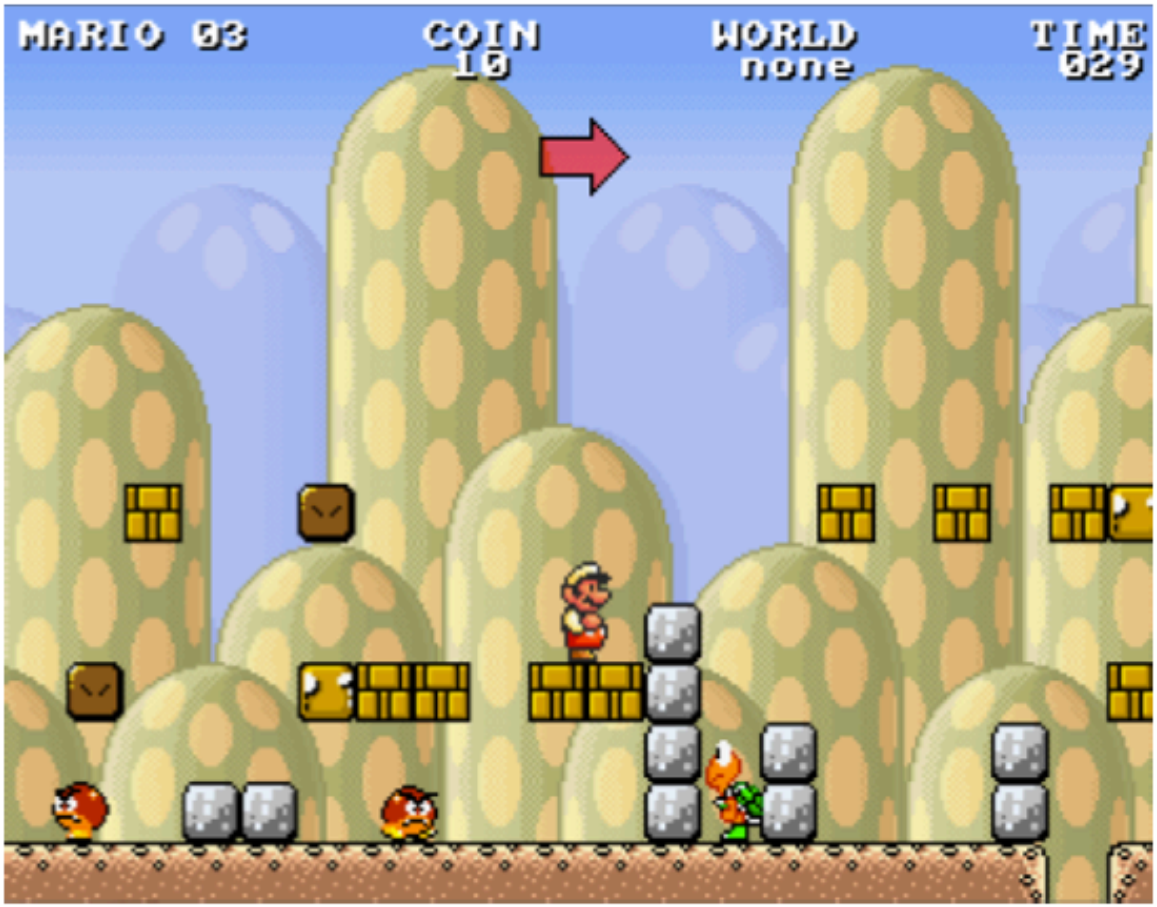
- Algoritmo proposto por [*Dahlskog and Togelius, 2013*]



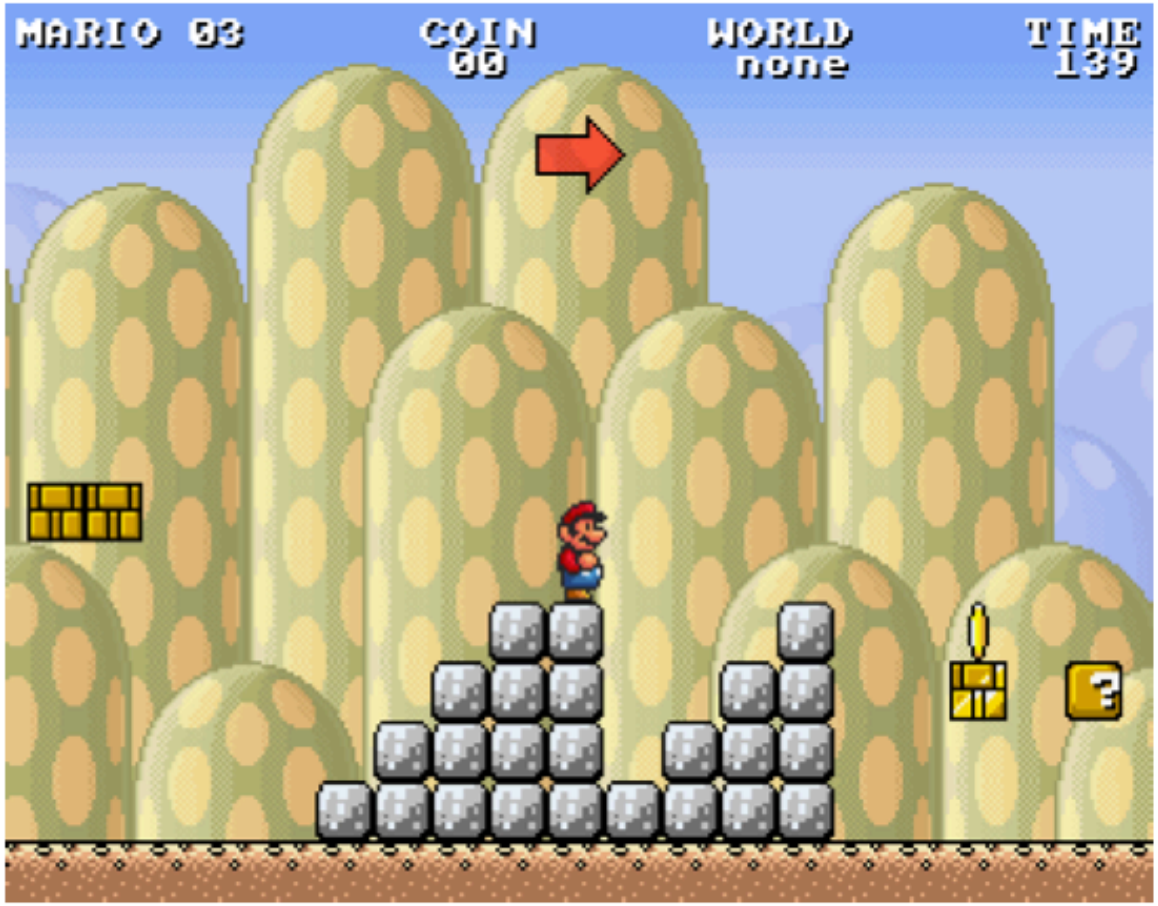
Algoritmo Genético

- População de tamanho 200
- Critério de parada: 10.000 gerações
- Elitismo de tamanho 50
- Crossover de 1 ponto
- Mutação uniforme

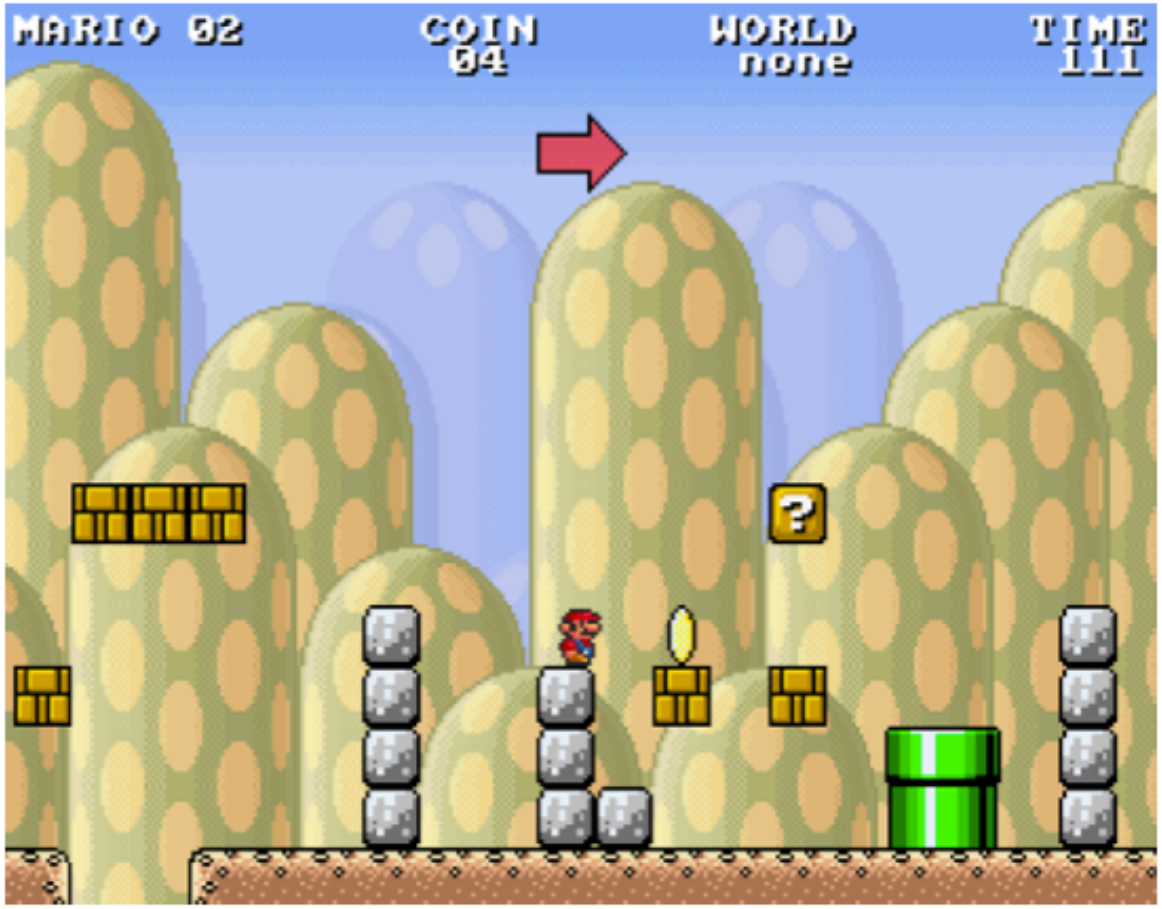
Exemplos de níveis gerados



Exemplos de níveis gerados



Exemplos de níveis gerados



Avaliação dos níveis gerados

- Experimentos com 3 grupos de 8 pessoas
- Os níveis foram avaliados por 3 critérios:
  - 1) *Boring-fun*
  - 2) *Not similar-similar*
  - 3) *Easy-Hard*
- Notas de 0 a 6

Avaliação dos níveis gerados

Version	Avr.	Median	Standard deviation
α Fun	3.75	4	1.041
α Similar	4.125	4	1.062
α Hard	3.5	3	0.791
β	3.792	4	1.159
β	4.625	5	0.906
β	2.375	2	0.989
γ	3.708	4	0.923
γ	4.833	5	0.875
γ	3.292	3	1.006

Tutorial

- Download: http://www.marioai.org/LevelGeneration/source-code
- Testar um nível gerado aleatoriamente:

  $ java dk.itu.mario.engine.Play
- Testar um nível gerado por PCG

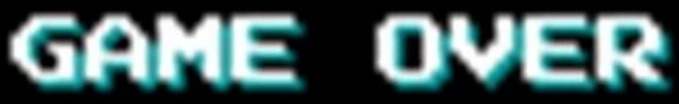
  $ java dk.itu.mario.engine.PlayCustomized

Tutorial

- Classes exemplo para construção de nível:
  - *dk.itu.mario.level.RandomLevel*
  - *dk.itu.mario.level.CustomizedLevel*
- Escreva sua classe para construção de nível implementando a seguinte interface:
  - *dk.itu.mario.MarioInterface.LevelInterface*
- Escreva sua classe para o gerador de nível implementando um dos métodos da classe:
  - *dk.itu.mario.MarioInterface.LevelGenerator*

Tutorial

- Para testar seu nível, altera a seguinte linha no método *init()* da classe dk.itu.mario.scene.LevelSceneTest:
  - CustomizedLevelGenerator clg = new CustomizedLevelGenerator()


GAME OVER