

# Rule-Based Godot: um arcabouço de Sistemas Baseados em Regras para uma *game engine*

Aluno: **Rodrigo Volpe Battistin** (rv.battistin@usp.br)

Orientador: Dr. Wilson Kazuo Mizutani

## Introdução

Dentre inúmeras aplicações de **Inteligência Artificial**, bots em *videogames* são um dos usos mais perceptíveis para o público geral há muitas décadas, desde *Pac-Man* (Bandai Namco, 1980) até *Valorant* (Riot Games, 2020).

A técnica de Sistemas Baseados em Regras se destaca em jogos como *Final Fantasy XII: The Zodiac Age* (Square Enix, 2017), no qual a mecânica de *gambits* (Figura 1) permite definir o comportamento de personagens por meio de **regras "se-então"**.

Sistemas desse tipo sofrem com dificuldade de implementação, assim, a proposta deste trabalho é fornecer **ferramentas que facilitem sua elaboração e integração** com uma *game engine*.



Figura 1: Regras (*Gambits*) em *Final Fantasy XII: The Zodiac Age* (Square Enix, 2017). Fonte: Nevada Dru em *Bits & Pieces* (10/01/2018)

## Metodologia

1. Estudo da teoria, tecnologias e exemplos de jogos
2. Elicitação de requisitos e proposta de arquitetura
3. Princípios de **software livre**
4. **Metodologia Ágil**
  - Iterações de duas semanas
  - Cenas de teste
  - Entregas parciais (*Alpha, Beta*)
5. Métrica com base em jogos de referência

## Resultado

**Rule-Based Godot**, um *plugin* para *Godot Engine 4.1+*. Escrito em *GScript*, sob licença MIT.

- Declaração de regras com variáveis
- Módulos prontos reutilizáveis
- Interface textual (Figura 2)
- Código aberto
- Integração com Editor nativo
- Extensível pelo usuário

```
1  { "Rules": [
2    { "if": [ "AND", [
3      { "Distance", "^../Ashe", 10, 20, "?target", ["Foes"] },
4      { "String", "Slow", "?target", [], "status" }
5    ] },
6    { "then": [
7      { "CallMethod", "^../Ashe", "attack", ["?target"] }
8    ] }
9  ] }
```

Figura 2: Interface textual para declaração de regras com sintaxe JSON

O arcabouço contém a estrutura básica de execução (Figura 3), além de recursos prontos:

- |  |   |
|--|---|
| <b>Condições Atômicas</b> (parte "se", ou "if")  | <b>Ações</b> (parte "então", ou "then")   |
| <ul style="list-style-type: none"><li>• Variáveis distintas</li><li>• Hierarquia na cena da <i>Godot</i></li><li>• Distância entre objetos</li></ul> | <ul style="list-style-type: none"><li>• Detecção de área</li><li>• Numérica</li><li>• <i>String</i></li><li>• Atribuir propriedade</li><li>• Chamar método</li><li>• Emitir sinal da <i>Godot</i></li></ul> |

### Funcionamento:

1. Adiciona-se um *RuleBasedSystem*
2. Criam-se as regras
  - i. A **condição** é uma **árvore** (operadores *booleanos* são conectores) que define um predicado lógico
  - ii. A lista de **ações** segue a **ordem de execução**
3. Faz-se uma varredura periódica do sistema, seguindo uma frequência configurada
4. Coletam-se todas as regras cuja condição é satisfeita
5. Aplica-se um método de resolução de conflito (chamado de **arbitragem de regras**) para decidir qual regra satisfeita será acionada
6. Acionam-se as ações da regra escolhida

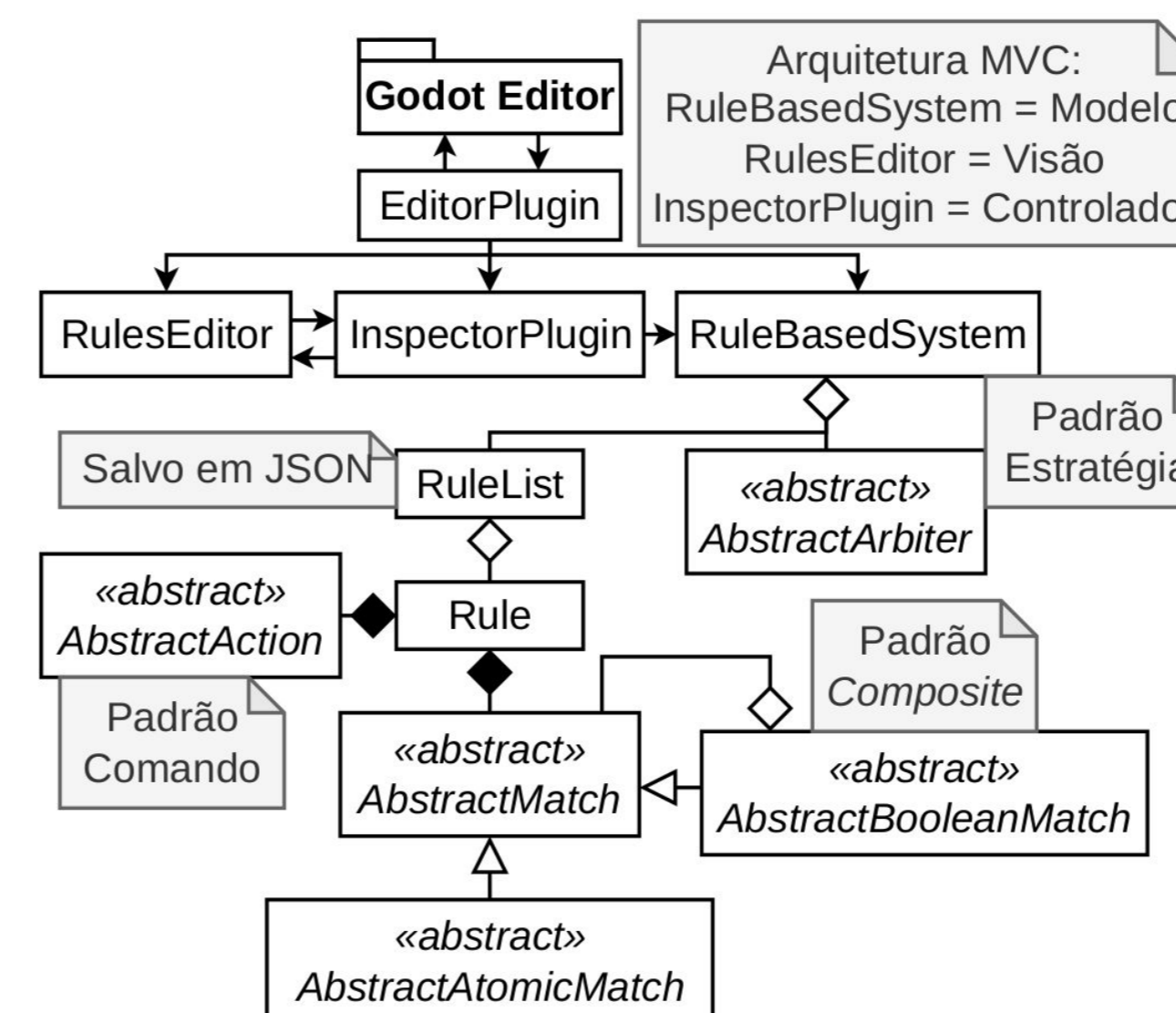


Figura 3: Diagrama de classes do sistema. Repare no uso dos padrões de projeto (Referência 2.)

## Conclusões

Com a instalação do *plugin*, desenvolvedores que utilizam a *Godot game engine* conseguem usar a tecnologia de Sistemas Baseados em Regras para definir o **comportamento de personagens** automatizados.

O uso da interface e dos módulos prontos permite que pessoas **sem muita experiência em programação** utilizem o sistema. Isso é relevante especialmente para *game designers*.

O programa é de código aberto e está disponibilizado para ser baixado, instalado e estendido por qualquer pessoa. A estrutura de classes abstratas e o fornecimento de *templates* permitem que o **usuário adicione novos recursos**.

- Cumpriu-se o objetivo de definir comportamentos semelhantes aos jogos de referência

## Referências

1. Bourg, D. e Seemann, G. (2004). *AI for game developers*. O'Reilly Media
2. Gamma, E., Helm, R., Johnson, R. e Vlissides, J. (1994). *Design Patterns: elements of reusable object-oriented software*. Addison-Wesley Professional
3. Millington, I. e Funge, J. (2009). *Artificial Intelligence for Games*. Morgan Kaufmann Publishers

Mais informações:

<https://linux.ime.usp.br/~rvbatt/mac0499/>

