Relatório: Etapa 3

André Meneghelli Vale - 4898948
andredalton@gmail.com

Taís Pinheiro - 7580421
tais.aparecida.pinheiro@usp.br

09 de Dezembro

1 Introdução

O projeto da disciplina *Laboratório de Programação 2* consiste no desenvolvimento de um sistema de batalhas entre dois jogadores de pokemom baseado no jogo homônimo. A batalha se desenvolve entre dois pokémons diretamente e não um grupo de pokémons como ocorre no jogo.

Para tanto foi escolhida a linguagem Python versão 3.4 e o desenvolvimento orientado a objetos.

Foram determinadas inicialmente três etapas:

- **Etapa 1**: Modelagem da batalha em modo texto;
- **Etapa 2**: Comunicação ponto a ponto via rede pra batalhas;
- **Etapa 3**: Inteligência artificial para as batalhas;

Este relatório é focado apenas na *etapa 3*.

1.1 Desenvolvimento

Nesta etapa, além de se inserir a Inteligência Artificial para batalhas (*IA*) também foi melhorada a apresentação do programa em relação a segunda etapa. Foram incluídas três novas funcionalidades (*auto*, *host* e *port*). Permitindo que o cliente possa se comunicar com qualquer servidor e também que o servidor aceite conexões de uma maneira mais controlada.

Embora todos os módulos utilizados nas batalhas tenham sofrido alterações, das classes utilizadas no pokémon apenas a classe *Pokemon* passou por alterações. Onde foi inserido um novo atributo booleano *auto*, que identifica se o pokémon é automático, assim como os métodos de acesso à este atributo:

- **set_auto()**: Marca o pokémon como automático;
- **get_auto()**: Retorna a identificação de automático deste pokémon.

E também o atributo **auto_attack** para guardar qual ataque foi considerado mais eficiente pelo método responsável pela IA, **on_my_own(other)**, que será tratado na próxima subseção.

2 Dinâmica da IA

Devido a baixa complexidade da batalha, e a batalha ser resumida a um duelo e não uma competição entre grupos. Não foi necessário criar uma técnica avançada de escolha do ataque.

O ataque é selecionado uma única vez de forma a resolver o duelo para o qual este pokémon foi carregado e armazenado no atributo auto_attack.

Para isso, é escolhido o ataque cujo ataque médio multiplicado pela acurácia seja maior.

3 Modo de execução

Este programa carrega um pokemon para uso como cliente ou servidor. Tanto o cliente (`battle_client.py`) como o servidor (`battle_server.py`) realizam o mesmo tratamento da linha de comando.

- **sem argumentos**: Modo default de execução, recebe os dados pela entrada padrão do sistema.
- **-h**: Imprime uma ajuda, `-h` pode ser substituído por `--help`;
- **-f**: Carrega as informações de um arquivo separado por linhas, `-f` pode ser substituído por `--file`;
- **-x**: Carrega as informações de um arquivo xml, `-x` pode ser substituído por `--xml`;
- **-a**: Inicializa com o pokemon em modo automatico, `-a` pode ser substituído por `--auto`;
- **-p**: Permite a passagem da porta de acesso do servidor por linha de comando, `-p` pode ser substituído por `--port`;
- **-H**: Permite a passagem da URL principal do programa por linha de comando, `-H` pode ser substituído por `--host`.

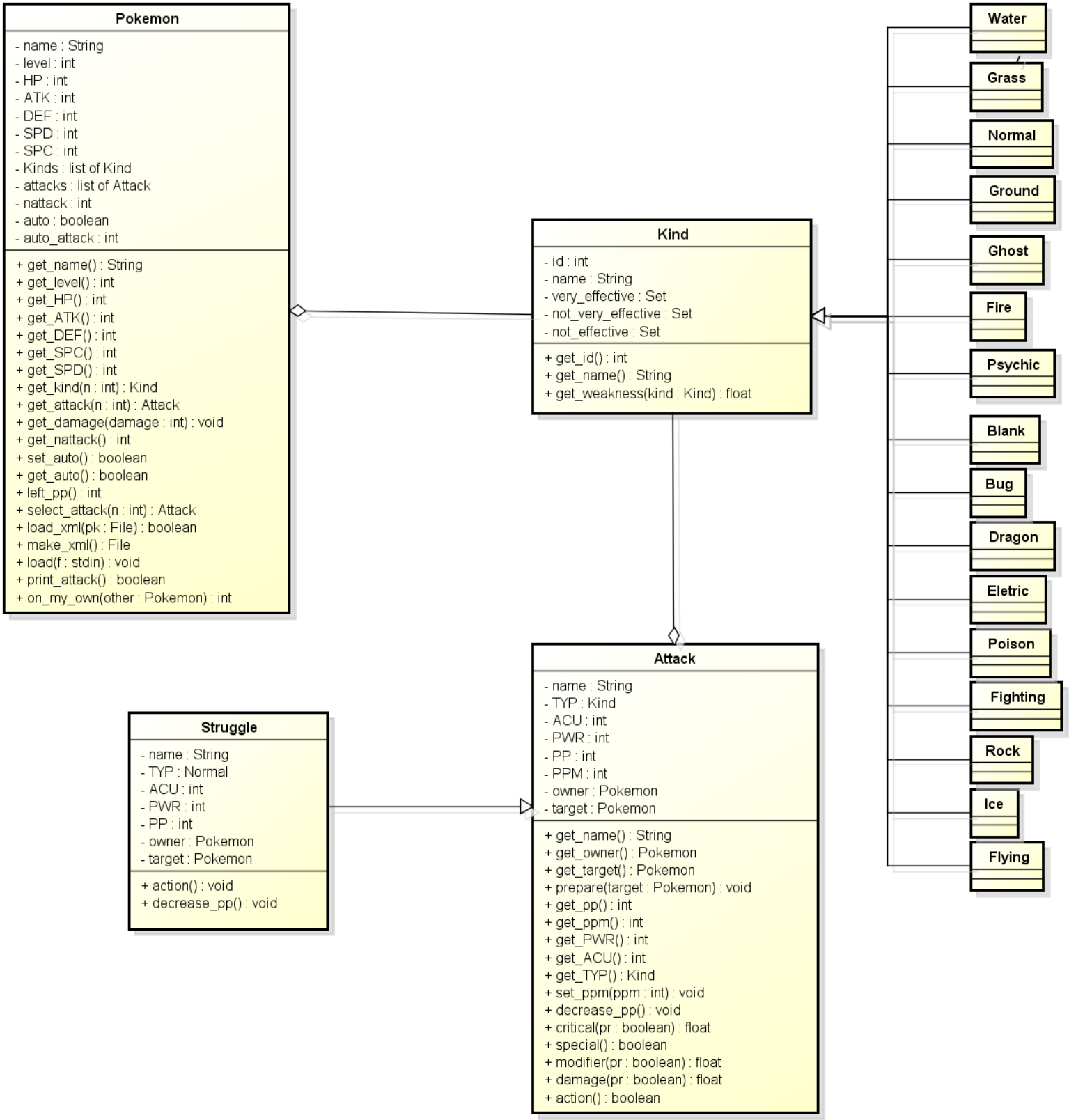
É possível selecionar o *host* e a porta tanto no servidor quanto no cliente, porém no servidor, o DNS precisa ser válido, por exemplo:

- **python battle_server.py -H localhost -f billpc/rattata**
  Funciona
- **python battle_server.py -H 127.0.0.1 -f billpc/rattata**
  Funciona somente para acesso local
- **python battle_server.py -H estranho -f billpc/rattata**
  Não funciona

Por padrão as configurações host:port são:

- **Server**: 0.0.0.0:5000, permitindo que o acesso seja feito de qualquer máquina;
- **Client**: localhost:5000, permitindo que o acesso seja feito apenas ao servidor local.

4 Diagrama de Classes



5 Limitações

Devido a forma de comunicação entre cliente e servidor algumas dificuldades foram encontradas, principalmente quanto a apresentação da ordem exata de ataques utilizados no cliente.

A implementação exigia que o servidor, quando mais rápido, realizasse o ataque e já respondesse com o *battle_state* contabilizando o seu próprio ataque. Como nenhuma lista de ataques é passada, saber quais ataques o servidor realiza depende de identificar qual dos ataques teve o PP reduzido. Tal fato impossibilita a impressão correta do primeiro ataque quando o servidor ataca primeiro. Fazendo com que o primeiro ataque impresso no cliente sempre seja o dele.

Também é impossível verificar a efetividade do ataque, e os casos onde o ataque não atinge o oponente foram identificados quando o HP do oponente não diminui. Permitindo que o cliente saiba quando o ataque atingiu o oponente.

6 Teste

Também foi desenvolvido um teste, que verifica se dois pokémons originados do mesmo arquivo selecionado aleatóriamente de `./billpc/` realmente montam objetos distintos mas possuem alguns atributos instânciados no mesmo lugar.

Seria necessário o desenvolvimento de testes para garantir a integridade tanto dos servidores quanto dos clientes. No entanto não soubemos como administrar testes adequados para estes aspectos.

7 Conclusão

Embora existam limitações dadas a forma de comunicação cliente/servidor não ser completa, é possível realizar batalhas de maneira adequada. Tanto manualmente como automáticamente.

A inteligência artificial desenvolvida foi bastante simplificada, como foi descrito na sessão específica. O que torna a boa escolha do pokémon algo muito mais importante do que a própria IA.

A falta de gerenciamento de um grupo de pokémons também impede que o sistema seja utilizado de maneira similar ao que ocorre no jogo. Já que um pokémon vencedor de um duelo não se mantém em batalha contra o próximo adversário.

De maneira geral o projeto se mostrou bastante útil para identificar as necessidades de uma boa organização de orientações a objeto adequadas e também ao desenvolvimento de um servidor web bastante simplificado.

A utilização do *Flask* permitiu que os códigos do cliente/servidor ficassem bastante simplificados sem necessidade de grande configuração como ocorre quando se desenvolve usando *Tomcat*, *Django* e etc.

Referências

[1] Enunciado

[2] Bulbapedia

[3] Serebii

[4] Pokémon Red e Blue - Wikipedia

[5] Modules - Python

[6] Flask - Python