MAC0242 - Laboratório de Programação II
Prof. Dr. Alfredo Goldman

Projeto da disciplina - Etapa 2

Visão geral

Nosso projeto da disciplina será a criação de um jogo de batalha, estilo Pokémon, utilizando **Python 3** com orientação a objetos para isso. Ele será composto de três etapas:

1. Modelagem da batalha em modo texto
2. Comunicação ponto a ponto via rede para batalhas
3. Inteligência artificial para batalhas

O objetivo é ao fim do semestre organizar um campeonato entre os grupos. Então desde o começo pensem na melhor modelagem possível para maximizar o aproveitamento de código entre as fases.

Em todas as etapas será esperada a manutenção de **diagramas de classe** descrevendo a modelagem do seu programa, bem como uma **cobertura de testes** completa.

Tecnologias

Obrigatoriamente, os programas devem ser desenvolvidos com Python 3. O uso de bibliotecas já existentes do Python é livre, mas devem ser claramentes indicadas em um arquivo *requirements.txt* [1].

Opcionalmente, mas altamente recomendado, é o uso do pacote virtualenv[2] para isolar essas dependências e garantir que seu ambiente de desenvolvimento seja facilmente replicável em outras máquinas (lembrando que precisamos executar seu programa nas nossas máquinas para poder corrigir!).

Para o servidor da web da segunda etapa uma boa opção é o Flask[3]. Esta é apenas uma sugestão e não uma determinação. Façam como julgarem mais adequado.

---

[1] https://pip.pypa.io/en/latest/user_guide.html#requirements-files
[2] http://virtualenv.readthedocs.org/en/latest/virtualenv.html#usage
[3] http://flask.pocoo.org/

Esclarecimentos e ajustes sobre a Etapa 1

- Lembrando que, exceto onde especificado, estamos usando a geração 1 (*Red/Blue/Green/Yellow*) como base.
- Níveis de prioridade dos ataques: o ataque acontece no turno do pokémon, ou seja, ele irá atacar e dar dano independente do ataque escolhido pelo oponente.
    - Ataques que consumam mais de um turno, da mesma forma, estão fora do escopo.
- Modificadores de tipo: usaremos a tabela de modificadores de tipo da geração 1[4], mesmo que ela contenha erros que depois foram arrumados.
- Itens e ataques que mudem algum atributo do pokémon continuam fora do escopo do projeto.
- Sigam as sugestões de implementação e definições dos enunciados.

Etapa 2

Com o conhecimento adquirido na etapa 1 sobre a modelagem de uma batalha Pokémon, ainda que rudimentar em diversos detalhes, agora finalmente podemos partir, depois de pequenos ajustes pedidos pelos alunos durante a primeira etapa, para a segunda etapa, que é a batalha via rede, mas ainda com humanos tomando as decisões de batalha pela interface do terminal definida na etapa 1. Para isso, seu programa deverá ser tanto um cliente quanto um servidor.

Então utilizaremos um padrão bastante difundido para WEB chamado de RESTful APIs[5]. Para que estes servidores funcionem entre si, agora precisamos definir o formato das mensagens esperadas e quais serão os caminhos para acessar os recursos do servidor.

Para o formato de comunicação utilizaremos arquivos XML que sigam o *schema* no arquivo **pokemon.xsd** anexado junto a este enunciado. Do ponto de vista do servidor, ele deve responder as seguintes requisições de um oponente:

[4] http://bulbapedia.bulbagarden.net/wiki/Type/Type_chart#Generation_I
[5] http://en.wikipedia.org/wiki/Representational_state_transfer#Architectural_constraints

- Iniciar batalha
    - Método HTTP: POST
    - PATH: /battle/
    - Parâmetros: um objeto battle_state, com apenas um Pokémon (o que o oponente usará na batalha)
    - Resposta: Se estiver disponível, um objeto battle_state, com dois Pokémons (o que o oponente usará na batalha e o que o servidor usará). Caso contrário um erro[6] número 423.
    - Observação: se o servidor for quem começa a jogar, ele já devolve o battle_state modificado por seu ataque
- Ataque
    - Método HTTP: POST
    - PATH: /battle/attack/<id>
    - Parâmetros: nenhum
    - Resposta: um objeto battle_state, com os dois pokémons que o oponente e o servidor estão usando na batalha.
    - Observação 1: a resposta é o battle_state com o ataques do cliente e do servidor já contabilizados.
    - Observação 2: <id> é o número que identifica qual ataque será utilizado.

Da mesma forma, seu programa deve possuir um modo cliente, mantendo a interface de entrada e saída da etapa 1, além receber pela entrada padrão o endereço do servidor de batalha. Esse cliente gerará requisições para o servidor de batalha de acordo com os comandos do usuário. Por exemplo, se o usuário executar um comando de ataque (que tem id = 1), seu programa deve gerar uma requisição para o servidor do tipo POST, com PATH */battle/attack/1*.

Tanto o cliente quanto o servidor continuam, assim como na etapa 1, recebendo os dados do Pokémon que será usado na batalha pela entrada padrão linha a linha, como foi definido então:

[6] http://en.wikipedia.org/wiki/List_of_HTTP_status_codes#4xx_Client_Error

<NOME>
<LVL>
<HP>
<ATK>
<DEF>
<SPD>
<SPC>
<TYP1>
<TYP2>
<NUM ATKS>
<NOME>
<TYP>
<ACU>
<PWR>
<PP>
…
<NOME>
<TYP>
<ACU>
<PWR>
<PP>

Conselho

Procure seus colegas para testar seus clientes e servidores uns contra os outros. Além disso, será que seu programa consegue jogar contra ele mesmo?

Assim que entregue, você já pode pensar em como deixar seu cliente e servidor inteligentes para a terceira fase. Nesta fase seu programa será inicializado como servidor ou como cliente, recebendo o endereço do servidor de batalha, e a partir daí deve batalhar sozinho.

O que deve ser entregue

- Todo o código fonte compactado em um arquivo **zip** ou **tar.gz**, inclusive testes de unidade que devem cobrir o código implementado
- Diagrama de classes descrevendo a implementação nos formatos **pdf** ou **png**
- Relatório
    - Nome e número USP de todos os integrantes do grupo
    - Descrição das dificuldades e desafios

Importante

- O trabalho é estritamente em grupos de no máximo 4 pessoas. Discussões entre os grupos serão mais do que bem vindas, mas plágio ou cola não serão tolerados. Veja a política do Departamento de Ciência da Computação para casos de plágio ou cola.
- Escreva de forma clara e estruturada todos os seus códigos e relatórios. Organize e nomeie todos os arquivos entregues de forma que eles possam ser facilmente identificados. A avaliação levará em conta todas essas questões! Uma apresentação ruim, ou a falta de clareza, poderá prejudicar sua nota.
- O programa deve ser entregue por meio do sistema Paca em um arquivo zip ou tar.gz
- Apenas **um** integrante do grupo deve fazer a entrega
- Enquanto o prazo de entrega não expirar, você poderá entregar várias versões do mesmo exercício-programa. Apenas a última versão entregue será guardada pelo sistema. Encerrado o prazo, você perderá 1 ponto para cada hora de atraso. Não deixe para entregar seu exercício na última hora!
- Guarde uma cópia do seu exercício-programa pelo menos até o final do semestre.
- Caso existam problemas de relacionamento entre os grupos, o professor e os monitores devem ser comunicados prontamente, para que medidas possam ser tomadas a tempo.