Universidade Federal de Minas Gerais
Instituto de Ciências Exatas - Departamento de Ciência da Computação

Programação Modular
2º Semestre de 2014
Prof. Douglas G. Macharet

Trabalho Prático 2 – Cartas na mesa

Jean-Luc Nacif Coelho - 2011049207
João Francisco Moreira Penna - 2011049231

1 - Introdução:

O objetivo deste trabalho prático é familiarizarmos com as etapas de análise e projeto de um sistema referente a um jogo de cartas. Para isso, devemos implementar as entidades necessárias para a implementação de um jogo de cartas (Cartas, Mão, Baralho, etc), assim como as funcionalidades relacionadas à um jogo específico.

No caso, optamos por trabalhar com o jogo de cartas Magic The Gathering, um jogo de cartas bastante complexo. Esse é um jogo de cartas colecionáveis inventado em 1994. O site oficial é www.wizards.com/magic e de lá pode ser baixado o conjundo de regras oficiais. Cada jogador começa com 20 pontos de vida e o objetivo geral é reduzir a vida do adversário até zero. Geralmente isso é feito invocando criaturas e ordenando-as ao ataque, mas as várias cartas do jogo trazem outras possibilidades.

Cabe dizer que Magic the Gathering é um jogo muito grande, que atualmente possui mais de dez mil cartas diferentes que adicionam cada vez mais variedade ao jogo, com novas variáveis, regras específicas e exceções que mudam como o jogo funciona. Para os propósitos desse Trabalho Prático, implementaremos um conjunto reduzido de cartas. Porém, esse conjunto que escolhemos é funcional e equilibrado, abordando todos os aspectos interessantes que cativam os fãs da série, e proporcionando uma boa experiência de jogo.

De maneira geral, a implementação traduz um jogo tradicional de Magic The Gathering. Cada jogador tem o seu baralho, e durante o seu turno, faz suas ações tentando vencer o adversário. O funcionamento geral do programa é controlado pelo GameCore, que chama as várias funções para desencadear as ações do jogo. Por exemplo, após o jogador digitar no console que quer invocar determinada criatura, é checado qual carta está sendo usada, de onde essa criatura está sendo invocada (nas cartas presentes na mão do jogador ou no cemitério), qual o custo de ativação de mana (e se é possível invocá-la com aquela quantidade de terrenos), qual o momento do turno. Considerando que é possível, a criatura é colocada no campo de batalha, e o console está pronto para receber um novo comando do jogador através do prompt, por exemplo, usando essa criatura para atacar o adversário e reduzindo seus pontos de vida. Quando o jogador finaliza seu turno, o jogador seguinte joga, e assim sucessivamente até que um dos jogadores tenha seus pontos de vida reduzidos a zero, caso em que o adversário é declarado vencedor.

2- Implementação:

Procuramos manter o máximo possível de modularização no nosso código. Pretendemos continuar e aperfeiçoar o projeto, adicionando mais cartas e funcionalidades, portanto evitamos ao máximo soluções “chumbadas”, cujas implementações funcionariam apenas visando um espectro menor do jogo. Sempre que usamos de uma solução simplificada desse tipo, que resolve apenas no universo reduzido presente no trabalho prático, deixamos explicitado nos comentários do próprio código. Mas sempre que possível, tentamos resolver da maneira mais abrangente possível. Por exemplo, será que compensa criar uma função só para tratar esse caso específico de uma única carta que ainda nem foi implementada? Talvez não

para o escopo do trabalho prático, mas como pretendemos dar continuidade ao projeto, tentamos reconhecer as muitas sutilezas das regras do jogo e criar implementações que prevêem esses casos que não são suportados na versão atual, mas serão implementados no futuro. Novamente, a boa utilização dos conceitos de Programação Modular é de grande ajuda, facilitando a manutenção do código.

Cada carta de Magic foi implementada em uma classe própria com seu nome. Isso porque cada carta é única em suas inúmeras características. Porém os conceitos de Programação Modular ainda se aplicam. Um exemplo disso é que as cartas herdam características de outras classes: todas as cartas de “Creature” tem um valor de força e resistência (power e toughness), e todas as criaturas são “Permanents”, todos “Permanents” são “Cards”, entre outros, estabelecendo assim uma forte noção de Herança e Generalização.

Imagem: Exemplos de cartas de magic.

Na imagem podemos ver bem essas características em comum. A cor das bordas indica que cor que é aquela carta. Todas tem o nome no topo, mas somente algumas possuem custo de ativação de mana (em cima, à direita). Em seguida a ilustração, e o tipo da carta. Observe que apenas as cartas do tipo Creature possuem os números de poder e resistência no canto inferior direito. Além disso, algumas possuem um subtipo, e um texto descrevendo habilidades e regras específicas daquela carta. Algumas cartas possuem também um texto em itálico com uma descrição de ambientação que não tem impacto nas regras.

Tivemos o cuidado de obedecer uma disposição de packages de maneira a agrupar as classes que fazem sentido juntas, em um formato claro e organizado. Por exemplo, as cartas estão dentro do package Cards, e as abstrações das quais as cartas herdam ficam em GameObjectCore. Além disso, prestamos atenção nos nomes escolhidos e suas funcionalidades. Por exemplo a classe Turn gerencia o andamento da rodada. Sua implementação é bem direta e simples, mas um entendimento das regras do jogo se faz necessário para a compreensão das minúncias da implementação (Magic é um jogo complexo! Mas o diagrama de atividades ilustra o funcionamento do Turno implementado e as decisões e comandos pertinentes ao jogador no Terminal de comando). Já a classe Player lida com as responsabilidades do jogador, Deck com o baralho, Console com a parte do terminal, e assim em diante.

O GameCore é o módulo principal para controle do jogo. Ele é implementado como uma facade, então se alguma implementação é modificada, tudo que é necessário é fazer mudanças nessa classe. Além disso, o GameCore é um singleton e existe também uma função que permite salvar o estado do jogo, para continuar uma partida em outro momento. Utilizamos uma interface para facilitar a utilização do GameCore.

A interface com o usuário foi dividida em GUI (interface gráfica) e TUI (interface de texto). A parte de texto acontece via terminal de comando, com o usuário usando o teclado para determinar suas ações no jogo. Algumas informações estão sempre exibidas na tela, e tomamos o cuidado de fazer o terminal se limpar para deixar a interface mais limpa. O jogador pode interagir com comandos simples que são explicados no próprio terminal: por exemplo digitando Battlefield (ou simplesmente B), e utilizando as teclas W,S,A,D é possível percorrer as cartas que estão na mesa. É aberta também uma janela com a parte da interface gráfica, ajudando a visualização ao mostrar a imagem da carta que está selecionada.

Futuramente, vamos ampliar as funcionalidades da interface gráfica, coisa que ainda não foi possível devido aos inúmeros detalhes que tivemos que tratar durante a implementação deste trabalho. Porém, já estamos esboçando soluções que vão permitir que o jogo seja jogável completamente a partir da interface, movendo as cartas e mostrando todas as informações na tela, inclusive montando as imagens das cartas dinamicamente (para quando algum efeito aumentar a resistencia de uma criatura, veremos esse valor aumentar na imagem da carta), as cartas rotacionadas quando ativadas, entre outras.

Existe no código tratamento de exceção para casos em que o jogo tenta fazer algo que não é permitido, como usar uma carta que você não tenha controle, ou uma criatura que não está indo para a zona de jogo adequada. Perda de jogador atual também é tratado como uma exceção. Isso foi decidido porque no Magic, quando um jogador perde (principalmente em um jogo com vários jogadores), muitas consequências são desencadeadas, como exílio de efeitos da pilha, as cartas que o jogador é dono são exiladas, as cartas que ele ganhou controle voltram para os seus donos, e as cartas que entraram no campo sob o controle dele são exiladas. Assim é possível tratar todos esses casos.

2. - Decisões de Implementação:

Consideramos que a maior decisão de implementação está ligada a quais cartas seriam implementadas. Como já foi dito anteriormente, Magic possui uma quantidade imensa de cartas e efeitos, e foi necessário um esforço para decidir qual seria a melhor combinação delas.

Os decks implementados são baralhos de 40 cartas. O primeiro é um deck verde de criaturas, pensado para ter uma bom ritmo de jogadas e colocando criaturas no campo de batalha e atacando frequentemente. Caso o jogo demore um certo número de turnos, esse deck coloca em jogos criaturas muito grandes e perigosas para o adversário. Acreditamos que esse deck é direto o suficiente para uma inteligência artificial conseguir executá-lo bem. Suas cartas são: 4x Barishi, 4x Garruk's Companion, 2x Indrik Stomphowler, 1x Kozilek, Butcher of Truth, 4x Leatherback Baloth, 4x Llanowar Elves, 4x Strangleroot Geist, 1x Terra Stomper e 16x Forest.

O outro deck é um deck preto de zumbis. Ele apresenta algumas opções maiores de liberdade de mecânica para o jogador, com mais variedade de tipos de cartas e opções criativas. As suas cartas são: 2x Black Cat, 4x Butcher Ghoul, 4x Geralf's Messenger, 3x Gravecrawler, 3x Warpath Ghoul, 1x Army of the Damned, 3x Assassinate, 2x Damnation, 2x Endless Ranks of the Dead e 16x Swamp.

Algumas cartas infelizmente ainda não estão com todas as funcionalidades implementadas. Black Cat, Damnation, Endless Ranks Of The Dead, Geralf's Messenger, Gravecrawler, Indrik Stomphowler e Kozilek, Butcher of Truth não tem as suas mecânicas mais avançadas. Não são cartas de difícil implementação, mas gastamos mais tempo resolvendo outros bugs no código, e os decks da versão atual são os que não apresentam erros inesperados. Futuramente, implementaremos corretamente essas e muitas outras cartas, com decks mais complexos de mais de uma cor, e cartas com efeitos diferenciados.

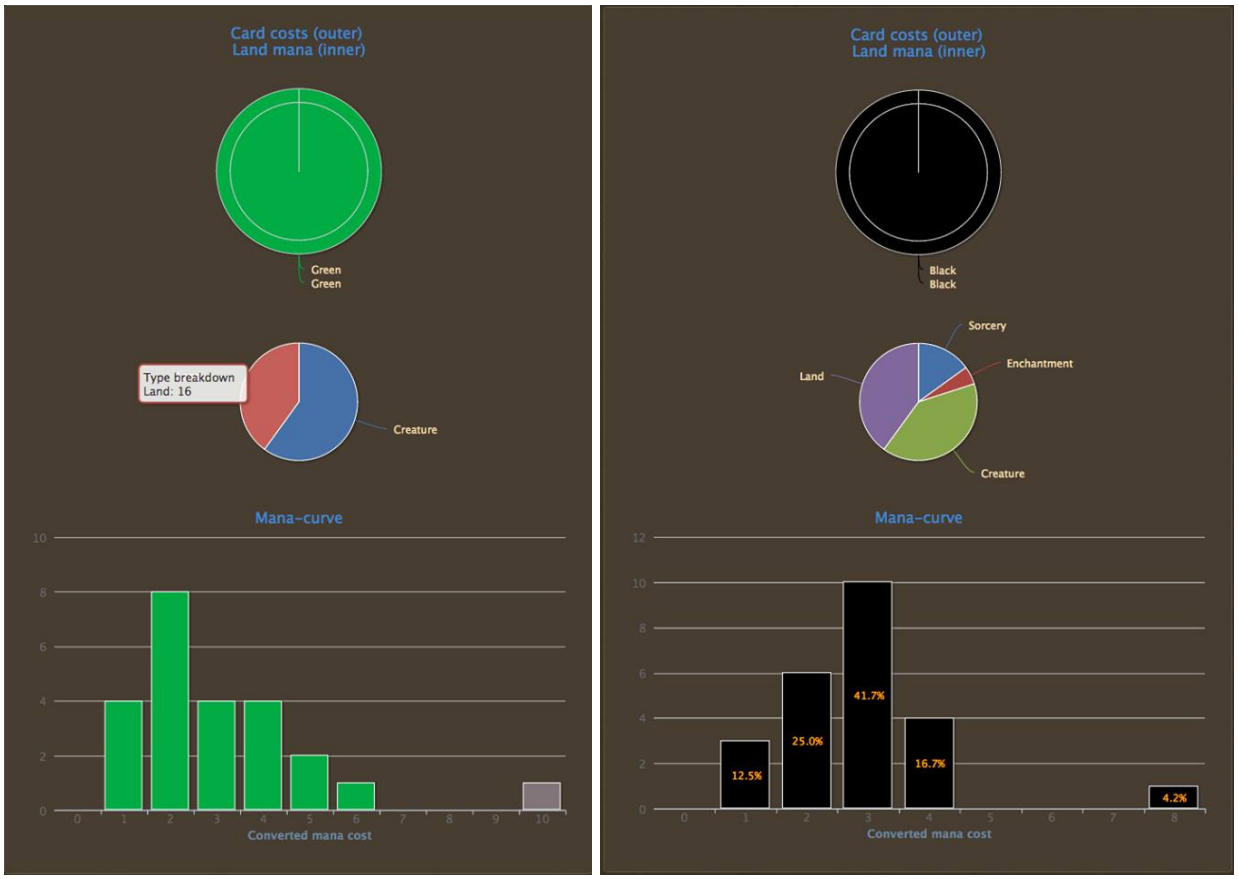

Imagem: Gráficos dos decks implementados. Dados exibidos incluem cores das cartas, tipos de cartas e proporções dos custos convertidos de mana.

Informações sobre as cartas podem ser encontradas no site http://magiccards.info/, e um construtor de decks do site http://tappedout.net/ foi utilizado para garantir que as cartas estariam distribuidas em boa proporção (com uma quantidade de terrenos compativel com o custo das criaturas utilizadas no deck). Ajustes foram feitos e algumas cartas substituídas até encontrarmos o melhor equilíbrio possível.

3- Testes

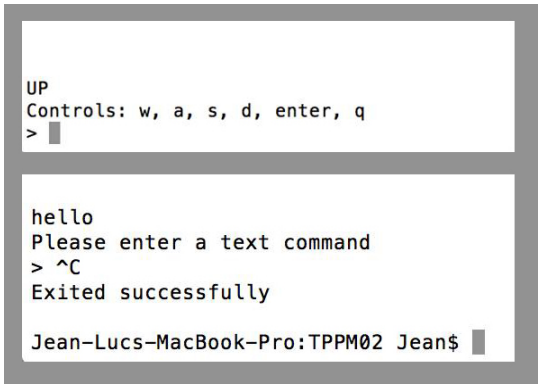

Imagem: Teste de interface baseada em texto e console.

Para a Interface baseada em texto, implementarmos uma maneira do prompt se limpar, deixando a interface mais limpa mas registrando todos os comandos em um log. Foi necessário também implementar um 'shutdownhook' pra impedir o terminal de se comportar de maneira inesperada no caso do programa ser interrompido durante o processamento dos comandos.

```
270 Controls: w, a, s, d, enter, q
271 >
272 User pressed 100
273 Swamp -
274 Basic Land - Swamp
275 (T: Add B to your mana pool)
276
277
278 Controls: w, a, s, d, enter, q
279 >
280 User pressed 100
281 Strangleroot Geist - GG
282 Creature - Spirit
283 Haste
284 Undying (when this creature dies, if it had no +1/+1 counters on it,
285 Geists of the hanged often haunt the tree on which they died.
286 2/1
287
288 Controls: w, a, s, d, enter, q
289 >
290 User pressed 100
291 Forest -
292 Basic Land - Forest
293 (T: Add G to your mana pool)
294
295
296 Controls: w, a, s, d, enter, q
297 >
298 User pressed 10
299 Chosen card: Forest
```

Imagem: Log mostrando o correto funcionamento do console e das cartas Swamp, Forest e Strangleroot Geist.

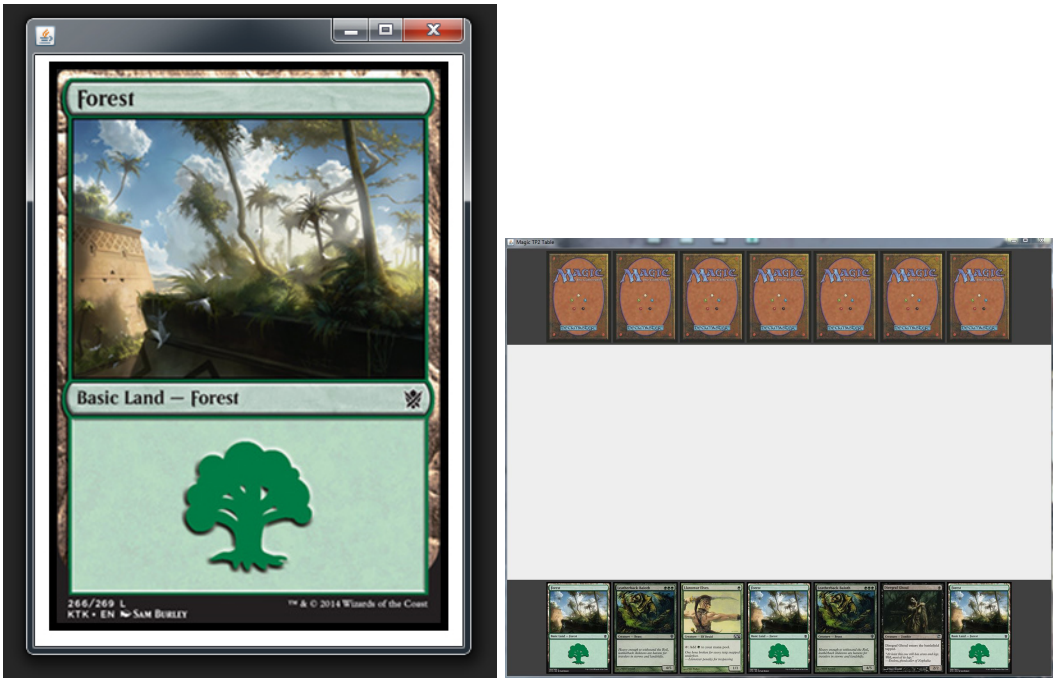

Imagem: Interface gráfica mostrando a carta. Na direita temos um trabalho em andamento.

Pretendemos investir bastante na Interface Gráfica em breve, pois acreditamos que se trata do ideal para uma boa experiência do jogo, facilitando a visualização e jogabilidade. Além de criar uma representação da mão dos jogadores e do campo de batalha, queremos deixar visível todos os dados necessários para que o jogo seja possível completamente através dessa interface. Isso inclui a quantidade de pontos de vida, quantidade de cada tipo de Mana, o cemitério e cartas exiladas. Queremos que as ações dentro da interface desencadeiem comandos, por exemplo, arrastar uma carta ou clicar nela para ativar uma habilidade. Outra idéia que queremos colocar em prática é conseguir montar as cartas dinâmicamente. Assim, quando algum efeito acontecer na carta, por exemplo, mudando sua cor ou alterando seus atributos, a mudança acontecerá visualmente e a carta terá uma nova imagem.

4- Conclusão

De maneira geral, esse trabalho propiciou um maior contato com a linguagem de programação Java, com as técnicas de programação orientada a objetos e técnicas de programação modular, além de um amadurecimento do processo de trabalho. Utilizamos diferentes padrões de projeto, e dado a complexidade do projeto, confirmamos a importância de um bom planejamento e o peso e importância das decisões de implementação.

O objetivo do trabalho era implementar as entidades necessárias para a implementação de um jogo de cartas, assim como as funcionalidades relacionadas ao jogo específico, no caso Magic The Gathering. Considerando o extenso conjunto de regras de Magic, algumas simplificações tiveram que acontecer, em especial se referindo ao número de cartas existentes (existem mais de dez mil cartas diferentes). Dessa maneira, o jogo é completamente funcional

dentro de o seu universo reduzido, ou seja, funciona perfeitamente com os decks e cartas que já foram implementados.

Infelizmente, não conseguimos implementar tudo que queríamos já nesse trabalho prático. Algumas cartas estão com as funcionalidades reduzidas, prejudicando o funcionamento que planejávamos para os Decks. Agora seguimos buscando aprimorar o projeto, ampliando as funcionalidades do programa, desde a implementação de novas cartas com efeitos e interações mais ousados, uma inteligência artificial, o desenvolvimento da interface gráfica, funcionalidades em rede, possibilidade de criação de decks e outros modos de jogo. Pretendemos reestruturar e refatorar algumas parte do código utilizando os padrões de projeto decorator e factory, e ter um código ainda mais bem estruturado e sólido.

Estamos felizes com a base obtida durante esse trabalho prático. Conseguimos com as classes implementadas obter resultados bem sucedidos como observados nos testes. Além disso, foi uma grande prática dos conceitos aprendidos em sala de aula. Por causa disso, consideramos esse Trabalho Prático um grande sucesso.

5- Bibliografia

5.1 Livros:

- MEYER, Bertrand. **Object-Oriented Software Construction.** Prentice Hall, 2000.
- WU, Thomas C. **An Introduction to Object-Oriented Programming with Java.** McGraw-Hill, 2009.
- SAVITCH, Walter. MOCK, Kenrick. **Absolute Java.** Addison-Wesley, 2012.
- DEITEL, Paul. DEITEL, Harvey. **Java: How to Program.** Prentice Hall, 2011.

5.2 Sites sobre o jogo Magic The Gathering:

www.wizards.com/magic
http://media.wizards.com/2014/docs/EN_M15_QckStrtBklt_LR_Crop.pdf
http://media.wizards.com/2014/downloads/MagicCompRules_20140926.pdf
http://magiccards.info/
http://tappedout.net/

5.3 Outros sites

http://docs.oracle.com/
http://docs.oracle.com/javase/tutorial/uiswing/events/actionlistener.html
http://stackoverflow.com/questions/11171021/java-use-keystroke-with-arrow-key
http://stackoverflow.com/questions/2347770/how-do-you-clear-console-screen-in-c
http://stackoverflow.com/questions/18669245/how-can-i-pipe-the-java-console-output-to-file-without-java-web-start
http://stackoverflow.com/questions/2533227/how-can-i-disable-the-default-console-handler-while-using-the-java-logging-api
http://stackoverflow.com/questions/7522022/how-to-delete-stuff-printed-to-console-by-system-out-println

http://stackoverflow.com/questions/3776327/how-to-define-custom-exception-class-in-java-the-easiest-way
http://stackoverflow.com/questions/6520914/defining-custom-gnu-make-functions
http://stackoverflow.com/questions/6038040/printing-variable-from-within-makefile
http://stackoverflow.com/questions/2701182/call-a-method-of-subclass-in-java
http://stackoverflow.com/questions/26434094/java-makefile-how-to-dynamically-compile-files-inside-packages/26444307#26444307
http://stackoverflow.com/questions/1909188/define-make-variable-at-rule-execution-time
https://www.gnu.org/software/make/manual/html_node/Appending.html
http://mrbook.org/tutorials/make/
http://www.cs.swarthmore.edu/~newhall/unixhelp/javamakefiles.html
http://www.darkcoding.net/software/non-blocking-console-io-is-not-possible/

5.4- Link para o Repositório no GitHub
https://github.com/Temennigru/TPPM02/