

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

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

Projeto Final – Jogo: Magic The Gathering



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

## **1 - Introdução:**

O objetivo deste projeto final é realizar todo o processo de desenvolvimento de um sistema (análise, concepção, implementação), aplicando os conceitos e técnicas vistas durante o curso (POO, UML, padrões de projeto, interface gráfica, testes unitários, etc).

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](http://www.wizards.com/magic) e de lá pode ser baixado o conjunto de regras oficiais, que foram a base para a implementação da nossa versão do jogo para computador.

A idéia geral do jogo é que cada jogador começa com 20 pontos de vida e o objetivo geral é reduzir a vida do adversário até zero. Os jogadores se revezam cada um jogando em um turno, e através das cartas do baralho usam mágicas que o ajudam a vencer, por exemplo invocando Creatures e ordenando-as ao ataque para diminuir os pontos de vida do adversário.

Cabe dizer que Magic the Gathering é um jogo muito grande, que atualmente possui mais de dez mil cartas diferentes na coleção completa. Geralmente não se permite que todas as cartas do jogo sejam usadas, sendo definida uma coleção menor, como por exemplo a coleção de cartas que escolhemos para a implementação desse projeto final. Apesar disso, houve um esforço muito grande para que a implementação fosse abrangente o suficiente para aceitar as mais diferentes cartas e mecânicas do jogo.

Um aspecto muito interessante de Magic é que as cartas são muito diferentes, com efeitos muito variados e proporcionam uma liberdade criativa para os jogadores. Um jogador poderia lançar uma carta que destrói suas próprias Creatures, em uma estratégia que utiliza uma carta que aproveita as Creatures do seu Graveyard. Não é óbvio que um jogador vai fazer uma jogada dessas, mas é importante que nossa implementação seja sólida para permitir esse tipo de jogadas diferentes e divertidas. E fazer isso é bem complexo considerando a quantidade de cartas e variáveis que existem no jogo.

## **2 - Regras básicas do jogo Magic The Gathering**

Parte da dificuldade de se começar a jogar Magic The Gathering é o seu conjunto complexo de regras. Acredito que não convém explicar todas elas na documentação, porém é interessante esclarecer pelo menos o básico.

Algumas decisões de implementação foram feitas baseadas em minúncias mais detalhadas das mecânicas do jogo. Infelizmente não convém detalhar as regras completas, mas o link é: [http://media.wizards.com/2014/downloads/MagicCompRules\\_20140926.pdf](http://media.wizards.com/2014/downloads/MagicCompRules_20140926.pdf).

Na nossa implementação para o computador, algumas regras se tornam mais intuitivas para se aprender a jogar. Por exemplo, o jogador não precisa saber que só pode ser jogado uma Land por turno - o próprio jogo não permite que você jogue errado e faça isso! Porém, algumas regras são seguidas à risca e deixam o jogo mais lento, por exemplo em determinada jogada será necessário que o jogador faça muitos cliques em várias telas de prompt, prejudicando a jogabilidade. Entendemos essa jogabilidade mais lenta como o jogo

sendo jogado da maneira correta, já que o nosso sistema tenta simular de maneira genérica e abrangente o jeito correto de se jogar o jogo.

### 2.1 - Deck e configuração inicial da mesa de jogo:

Cada jogador possui o seu próprio Deck (baralho). O baralho de onde você compra as cartas é chamado Library. No início do jogo, você compra 7 cartas e coloca elas na sua Hand (mão). Cada jogador também tem uma pilha de descartes, chamada Graveyard, e o campo de batalha onde a maioria das cartas são jogadas chama-se Battlefield.

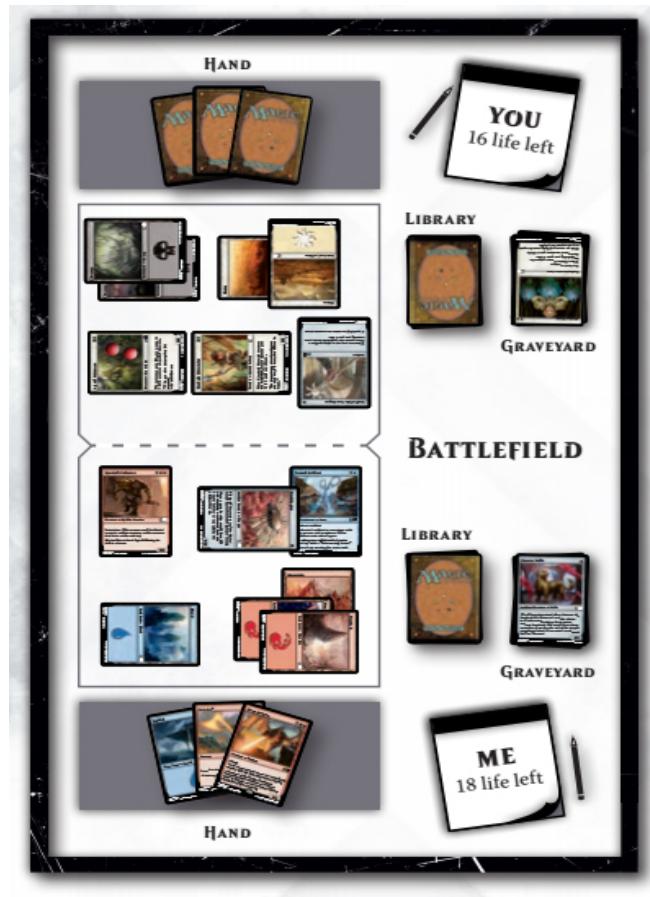


Imagen: Configuração de uma mesa de jogo, com áreas para Hand, Battlefield, Library, Graveyard.

### 2.2 - Objetivo e condições de vitória:

Você e seu adversário começam com 20 de vida, e se essa quantidade for reduzida para zero ou menos, esse jogador perde o jogo. Também se perde o jogo se você tenta comprar uma carta do Library que não possui mais cartas para ser compradas, e certas cartas podem trazer condições adicionais para a vitória.

Os jogadores se revezam cada um jogando em um turno, e através das cartas do baralho usam mágicas que o ajudam a vencer, por exemplo invocando Creatures e ordenando-as ao ataque para diminuir os pontos de vida do adversário.

## 2.3 - Tipos de Cartas:

Seu baralho contém carta de diferentes tipos: Lands, Creatures, Sorceries, Instants, Artifacts, Enchantments. Esses tipos de cartas tem um funcionamento básico que vai ser descrito a seguir, mas a maioria das cartas vai ter um funcionamento único que é descrito no texto da própria carta.

Lands são a fonte de Mana, que é a energia gasta para pagar o custo de ativação de outras cartas (para poder jogá-las). Todas as cartas exceto os Lands tem um custo de mana no canto superior direito. Você pode jogar um Land a cada turno. Para adquirir mana, você vira (tap) os Lands necessários no Battlefield, adicionando Mana ao seu Mana Pool.

Creatures são Permanents (cartas que ficam no Battlefield) que representam criaturas sobre o seu controle. Durante o combate, eles podem atacar os adversários e defender de ataques inimigos. Você pode lançar esse tipo de cartas durante a sua Main phase.

Sorceries e Instants são magias que possuem uma variedade grande de efeitos, e assim que o efeito acontece, essas cartas são colocadas no Graveyard (ao invés do Battlefield). Você pode lançar uma Sorcery na sua Main phase. Já um Instant pode ser lançado praticamente em qualquer momento, até no turno do adversário.

Artifacts e Enchantments são Permanents que representam itens e manifestações mágicas. A maioria dos Artifacts não possuem cor, então não precisam de uma cor de mana específica para ser lançado, e alguns Creatures também são Artifacts. Você pode lançar Artifacts e Enchantments na sua Main phase.



Imagen: Exemplos de cartas de magic. Dois Creatures, um Land, um Sorcery.

Na imagem podemos ver algumas cartas de Magic. As quatro cartas são diferentes, mas partilham algumas características. 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 depois o tipo e subtipo da carta. Observe que apenas as cartas do tipo Creature possuem os números de Power e Toughness no canto inferior direito. Além disso, algumas possuem um texto descrevendo habilidades e regras específicas daquela carta. Por exemplo, quando a carta "Black Cat" é destruída, o jogador alvo descarta uma carta - isso é uma mecânica do Black Cat, e outras Creatures não necessariamente compartilham dessa mecânica. O Strangleroot Geist é um Creature que

possui Haste (pode atacar no mesmo turno que foi colocado no Battlefield) e Undying (quando é destruída, ela volta do Graveyard para o Battlefield com mais Power e Toughness), então apesar dos dois serem Creatures e terem algumas similaridades, eles funcionam bem diferente na prática, abrindo para diferentes estratégias de jogo. Por fim, o texto em itálico presente em algumas cartas contém uma descrição de ambientação que não tem impacto nas regras.

#### 2.4 - Combate:

Durante o Combat Phase, que acontece no meio do seu turno, você decide quais Creatures irão atacar o adversário (e não outros Creatures diretamente). As Creatures que você escolheu são viradas e estão atacando. Então o adversário pode optar defender com os Creatures dele que estão desvirados. Cada Creature pode bloquear um atacante, mas é possível que mais de um Creature defendam um mesmo atacante (nesse caso, o atacante escolhe o quanto de dano será aplicado em cada Creature). O dano é aplicado aos Creatures que estão em combate (em geral o valor igual ao seu Power) e Creatures que levaram mais dano que seu Toughness são destruídos. Se um Creature não for bloqueado, ela aplica o dano ao jogador adversário.

**Obs:** Caso um Creature tenha entrado no Battlefield naquele turno, ele não pode atacar (mas pode defender). Normalmente os Creatures desviram no início do próximo turno, na untap phase, explicada a seguir.

#### 2.5 - Fases do Turno

Já mencionamos que os jogadores agem cada um em um turno. Um turno tem as seguintes partes: Untap, Upkeep, Draw, Main, Combat, Main (segunda parte), End, Cleanup.

- Untap: Desvire todas as suas cartas viradas (tapped).
- Upkeep: (algumas cartas tem efeitos nesse momento do turno).
- Draw: Compra uma carta do seu Library.
- Main: Pode jogar uma Land da sua Hand colocando-a no Battlefield. Você pode lançar Creatures, sorceries e outras magias que puder pagar o custo de ativação.
- Combat: Fase de combate. Inicialmente são declarados os atacantes (e virados os Creatures com as quais você quer atacar o adversário). Depois são declarados quem vai defender - cada Creature desvirado pode bloquear um atacante, e mais de um Creature pode bloquear o mesmo atacante. Não é obrigatório para um Creature bloquear. Em seguida, é calculado o dano de combate (primeiro das Creatures que possuem "First Strike"). Creatures que foram bloqueadas dão dano igual a seu Power umas às outras. Se um Creature toma dano maior que o seu Toughness, ele é destruído. Creatures que não foram bloqueadas, dão dano no adversário, que perde aquele tanto de vida.
- Main: Novamente a fase principal, que você pode jogar uma Land - se ainda não o fez esse turno, e outras magias.
- End: Fim do turno.
- Cleanup: Danos nas Creatures são curados. Jogador deve descartar se estiver com mais de 7 cartas na Hand.

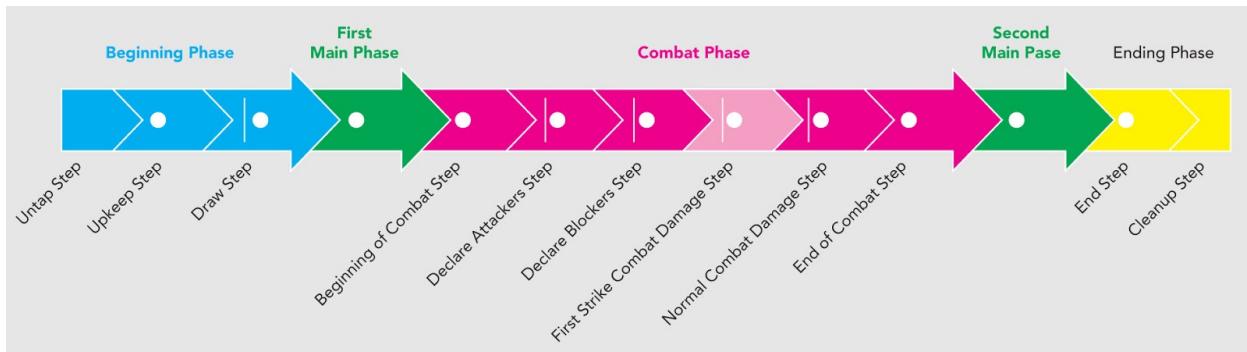


Imagen: Esquema visualizando as etapas do Turno. Geralmente, as grandes decisões são tomadas na "Main Phase".

Na implementação do TP2, o Beggining Phase (Untap, Upkeep, Draw) acontecia automaticamente com o programa desvirando as cartas do jogador e comprando uma nova carta do Library - colocando o jogador direto na Main Phase, que é a etapa importante de decisões do turno. Porém, como a intenção deste TP era de fazer um sistema mais genérico e abrangente que seguia fielmente as regras de Magic, essas etapas não devem ser mais automatizadas (o que deixa o jogo mais lento, porém mais correto).

## 2.7 - Observação

Já foi mencionado, mas convém reforçar já que é uma mecânica importante e definidora do Magic. As diferentes cartas costumam trazer um funcionamento especial diferente do básico. Essas regras especiais são descritas no texto da carta, e tem precedência sobre as regras básicas do jogo. Um exemplo comum é a habilidade “Haste”, que permite que um Creature que tenha acabado de entrar no Battlefield já possa atacar naquele mesmo turno, sendo que Creatures normalmente não poderiam. Algumas regras são mais específicas e modificam ainda mais como o jogo funciona. A combinação de cartas com regras muito diferenciadas pode ter uma interação complexa, por isso a implementação deve ser cuidadosa a fim de acomodar esse tipo de jogabilidade.

## 3- Implementação:

Um objetivo da implementação era fazer um bom código, modularizado e legível, mas ao mesmo tempo complexo o suficiente para adequar-se às mais variadas situações de gameplay.

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. 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. Então sempre que possível, tentamos resolver da maneira mais abrangente, utilizando bem os conceitos de Programação Modular visando criar um bom código e facilitar sua manutenção.

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, suas imagens em .jpg estão no diretório img, 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 minúcias da implementação. Já a classe Player lida com as responsabilidades do jogador, Deck com o baralho, Console com a parte do terminal, e assim em diante. Confira o detalhamento maior das classes nos diagramas UML em anexo ao trabalho prático.

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.

Na versão anterior, as cartas her davam características de uma classe abstrata. Todos os Creatures são Permanents, todos Permanents são Cards, entre outros, estabelecendo uma forte noção de Herança e Generalização. Porém, optamos refatorar essa parte do código e mudamos a forma que essa estrutura foi implementada.

O jogo possui uma Interface Gráfica (GUI) que permite que o jogador visualize facilmente as cartas da sua mão e da mesa. Vários detalhes podem ser observados no layout principal de jogo. Acreditamos que esse layout vai funcionar especialmente bem quando futuramente implementarmos uma versão online. Ainda não implementamos o módulo para montar as imagens das cartas dinamicamente - queremos construir as cartas de maneira que se um efeito aumenta a resistência da criatura, esse número aumente seu valor na imagem da carta. Porém, alguns outros efeitos, como as rotações de cartas ativadas já existem. Não utilizamos mais a TUI, que era a interface de texto que estávamos usando anteriormente.

Foi implementado uma classe chamada LogicSolver. No Magic, existe um certo tipo de carta que produz efeitos toda vez que uma condição acontece. Por exemplo, "toda vez que um Creature com Power maior que 3 entrar no Battlefield, todos os jogadores perdem 1 de vida". Para implementar isso, temos que verificar essa condição, e é nesse momento que o LogicSolver se faz importante: verificando se a carta que está sendo utilizada é um creature E tem power maior que 3 E está entrando no Battlefield.

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.

#### **4. - Decisões de Implementação:**

Acreditamos que a maior decisão de implementação se refere à proposta de tratarmos o Magic the Gathering como um sistema complexo que deve pretendemos futuramente implementar por completo. É um trabalho em progresso, mesmo depois da entrega do trabalho prático, pois ainda existem cartas e funcionalidades a serem implementadas, como suporte para jogar com mais que dois jogadores, jogar online, montar decks.

Querer implementar Magic dessa maneira é um objetivo ambicioso, tendo em vista o seu tamanho e complexidade. Também é preciso notar que geralmente, as versões de Magic para o computador não seguem as regras literalmente - porque elas deixam o jogo mais lento e travado prejudicando a jogabilidade. Tentamos não ultrapassar nenhuma regra, mesmo quando a custo de jogabilidade (como por exemplo, do jogador ter que clicar mais vezes para fazer uma ação simples).

Para a nossa implementação, escolhemos um conjunto limitado de cartas. Isso é perfeitamente normal, como se trata de um jogo de cartas colecionáveis, os jogadores não costumam ter acesso a todas as cartas existentes. Além disso, os formatos de campeonato também limitam as escolhas das cartas a coleções específicas, em que apenas determinadas cartas são permitidas e outras são banidas. Com essa coleção reduzida criamos dois decks com 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 Gravewcrawler, 3x Warpath Ghoul, 1x Army of the Damned, 3x Assassinate, 2x Damnation, 2x Endless Ranks of the Dead e 16x Swamp.

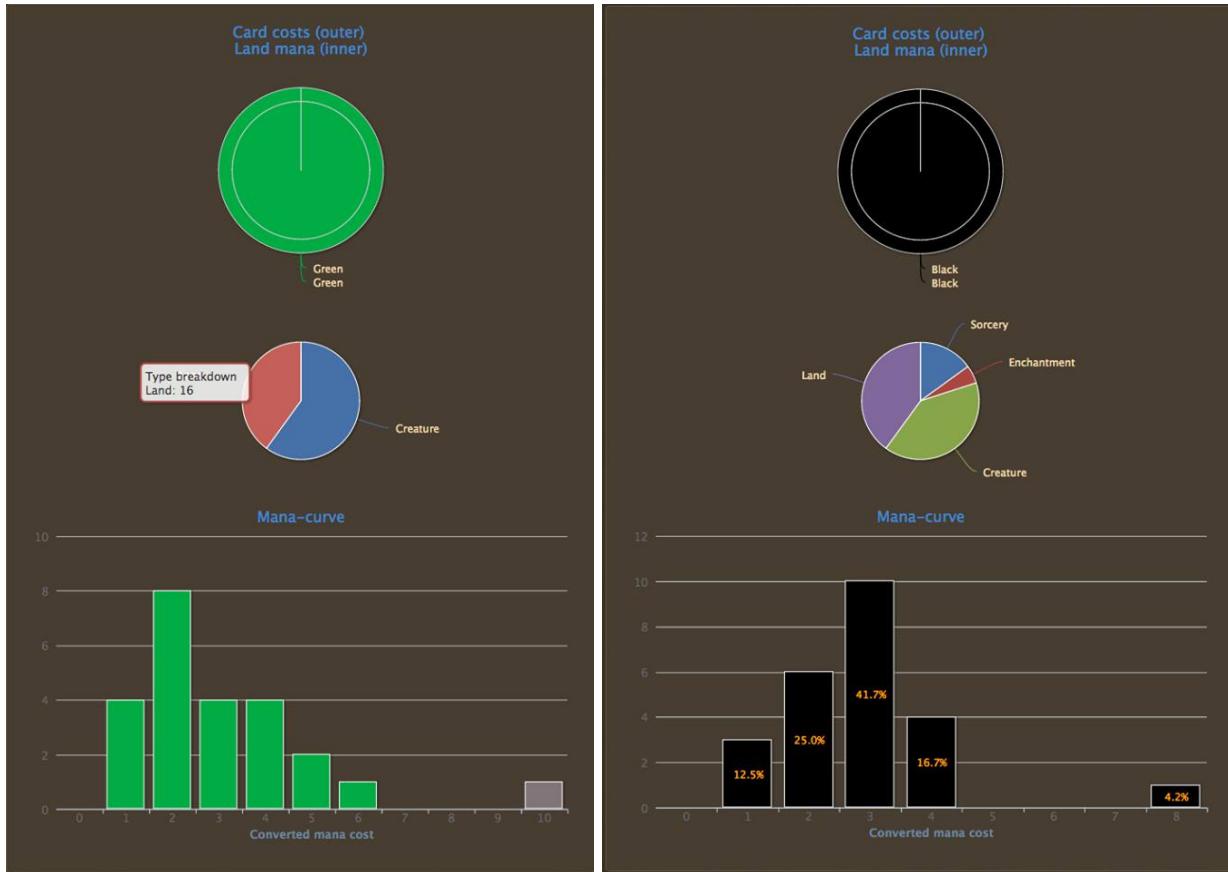


Imagen: Gráficos dos decks implementados. Dados exibidos incluem cores das cartas, tipos de cartas e proporções dos custos convertidos de mana.

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 distribuídas em boa proporção (com uma quantidade de terrenos compatível com o custo das criaturas utilizadas no deck). Ajustes foram feitos e algumas cartas substituídas até encontrarmos o melhor equilíbrio possível.

## 5- Testes



Imagen: Interação com as cartas na interface gráfica: virando a carta com o clique do mouse



Imagen: Layout principal. Em sentido horário: painel de informações, painel de detalhes da carta, Battlefield do adversário, Battlefield do jogador, Hand e Graveyard do jogador.

```
Jean-Lucs-MacBook-Pro:TPPM03 Jean$ java GameCore/Util/LogicSolver
Test 1 passed
Test 2 passed
Test 3 passed
Test 4 passed
Test 5 passed
Test 6 passed
Test 7 passed
Test 8 passed
Test 9 passed
Test 10 passed
Test 11 passed
Test 12 passed
Test 13 passed
Test 14 passed
Test 15 passed
Test 16 passed
Test 17 passed
Test 18 passed
Test 19 passed
Test 20 passed
Test 21 passed
Test 22 passed
Test 23 passed
Test 24 passed
Test 25 passed
Test 26 passed
```

Imagen: Teste unitário do módulo LogicSolver, funcionando corretamente.

## 6- Como usar a Interface Gráfica para jogar



Dado a complexidade do Magic, pode ser intimidador começar a jogar. Então essa parte da documentação ajuda a entender a interface gráfica para que se possa testar o jogo. A interface criada é bem simples e intuitiva, e praticamente toda jogável apenas com o mouse, exceto alguns poucos comandos via o teclado em certas janelas auxiliares.

A distribuição inicial de cartas é feita automaticamente pelo programa. Então você começa com 7 cartas na aba "Hand", na parte de baixo da janela. Quando se passa o mouse

em cima de uma carta qualquer, a sua imagem é mostrada grande na tela (no segundo painel) permitindo ao jogador ver detalhes do texto da carta.

Ao clicar com o mouse em uma carta na sua Hand, o jogo abre uma nova janela perguntando o que você quer fazer. Em geral, se quer usar a carta, que damos o nome de “Cast”, ou descer um terreno que é chamado de “Play Land”. Clique na carta Forest (ou Swamp) para colocar essa carta no Battlefield. Agora ela estará na sua divisão do painel no canto superior direito da janela principal. Só se pode jogar um terreno básico no Battlefield por turno. Para passar para a próxima etapa do turno, se clica no botão “PASS”, no painel mais a esquerda.

Depois que passarem todas as etapas do seu turno, será o turno do seu adversário (e. Quando for novamente o seu turno, você poderá descer um novo terreno. Vamos supor que nesse segundo turno você desce mais uma Forest, e tem em sua mão uma carta chamada Strangleroot Geist. Primeiramente você vai usar as habilidades das Forests que estão no Battlefield para gerar mana. Para fazer isso, clique nas cartas no Battlefield e escolha a habilidade de virar para adicionar uma Green Mana ao seu Mana Pool. Faça isso com as duas Forest, e em seguida, clique com o mouse no Strangleroot Geist na sua Hand. Essa carta custa apenas duas Green Mana para ser cast, então na nova janela que abriu, digitamos esse valor para determinar de onde está vindo a mana para pagar o custo de ativação da carta. Essa janela nova é necessária porque pode se usar mana de qualquer cor para pagar um custo incolor de uma carta, sendo necessária a decisão do jogador sobre qual mana de seu mana pool se quer gastar. O jogo faz uma checagem para verificar se é possível castar a carta, e caso seja, ela é colocada no Battlefield.

Agora que você já tem uma Creature no Battlefield, você pode usar o botão PASS para avançar para a etapa de combate. Como o Strangleroot Geist é um Creature que pode atacar no mesmo turno que entrou no Battlefield (já que ele tem a habilidade “haste”), clique na carta para definir essa Creature como um atacante. Supondo que o adversário não tenha escolhido nenhuma Creature para bloquear, você causa 2 pontos de dano ao jogador adversário. Esse valor é atualizado no painel a esquerda, agora o adversário tem 18 pontos de vida enquanto você ainda tem 20. Quando algum jogador chegar a 0 ou menos pontos de vida, o jogo termina.

## **7- 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.

Considerando o extenso conjunto de regras de Magic, algumas simplificações tiveram que acontecer, em especial se referindo ao número de cartas. Existem mais de dez mil cartas de Magic, então a nossa implementação lida com uma coleção reduzida, apesar de tentar ser abrangente e genérica para qualquer carta poder ser facilmente implementada. As cartas que foram escolhidas para serem implementadas funcionam corretamente de acordo com as

regras do Magic, e considerando as estruturas implementadas, não é difícil implementar mais cartas futuramente.

Tivemos alguns problemas com tempo, pois realmente tentamos fazer a implementação o mais completa e correta possível. Então parte do tempo que poderia ter sido gasto por exemplo em fazer uma interface gráfica mais elaborada foi gasto fazendo módulos que permitissem que certas cartas pudessem futuramente serem implementadas. Isso não é evidente na implementação porque embora essas cartas agora tenham suporte para serem implementadas, elas não estão implementadas de fato. Então várias refatorações de código deixaram a nossa implementação muito mais sólida e estável, porém, essas mudanças não são evidentes sem analisar o código.

Outra dificuldade foi o conhecimento aprofundado das regras do jogo Magic. Ambos os integrantes conhecem bem as regras, mas como a decisão foi basear-se no conjunto mais complexo das regras, algumas vezes soluções implementadas não eram abrangentes o suficiente para adequarem-se a algum detalhe do jogo. "Mas e se essa carta que não está na nossa coleção fosse usada nessa condição?" geralmente era respondido com uma refatoração do código para deixar a implementação mais abrangente.

O objetivo do projeto final era ter uma implementação correta e funcional do jogo de Magic The Gathering, com a aplicação de conceitos e técnicas de Programação Modular. Acreditamos que esse Trabalho Prático foi um grande sucesso, tanto pelo aprendizado, prática, e resultados obtidos.

## **8- Bibliografia**

### **8.1 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.

### **8.2 Sites sobre o jogo Magic The Gathering:**

[www.wizards.com/magic](http://www.wizards.com/magic)

[http://media.wizards.com/2014/docs/EN\\_M15\\_QckStrtBklt\\_LR\\_Crop.pdf](http://media.wizards.com/2014/docs/EN_M15_QckStrtBklt_LR_Crop.pdf)

[http://media.wizards.com/2014/downloads/MagicCompRules\\_20140926.pdf](http://media.wizards.com/2014/downloads/MagicCompRules_20140926.pdf)

<http://magiccards.info/>

<http://tappedout.net/>

### **8.3 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](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/>

#### 8.4- Link para o Repositório no GitHub

<https://github.com/Temennigru/TPPM03/>