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

Ronan de Castro Paula Lana

MONOGRAFIA DE PROJETO ORIENTADO EM COMPUTAÇÃO II

Desenvolvimento de uma engine para o jogo ZTCG:HS

Belo Horizonte
2016 / 1

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Desenvolvimento de uma engine para o jogo ZTCG:HS

por

Ronan de Castro Paula Lana

Monografia Projeto Orientado em Computação II

Apresentado como requisito da disciplina de Projeto Orientado em Computação II do Curso de Bacharelado em Ciência da Computação da UFMG

Prof. Dr. *Renato Antônio Celso Ferreira*
Orientador

Belo Horizonte
2016 / 1

À Deus,
aos professores,
aos colegas de curso e
aos meus familiares,
dedico este trabalho.

AGRADECIMENTOS

Inicialmente quero agradeçer a Deus, pelos dons recebidos.

Agradeço aos meus pais, pelo amor incondicional.

Aos meus professores, pelos conhecimentos adquiridos.

E finalmente aos colegas de curso pela convivência e trocas.

“Jamais considere seus estudos como uma obrigação, mas como uma oportunidade invejável para aprender a conhecer a influência libertadora da beleza do reino do espírito, para seu próprio prazer pessoal e para proveito da comunidade à qual seu futuro trabalho pertencer.”

Albert Einstein

RESUMO

Para a edição passada, foi proposto e desenvolvido um sistema completamente automatizado para um jogo de cartas colecionáveis (*Trading Card Game - TCG*) inédito. Cartas que implementavam certas mecânicas quando ativadas em partida, coleções de cartas separadas por entidades definidas como *profiles* (contendo todos os registros do usuário, bem como *deck*, seu desempenho nas partidas, etc.), elaboração de uma UI (*User Interface*) amigável aos usuários; estes foram os focos abordados durante o desenvolvimento do POC I.

Dito isto, o foco que foi definido para este projeto consiste em incrementar o projeto desenvolvido na etapa anterior: tornar flexível a *engine* deste sistema de jogo de cartas. O usuário do *framework* poderá definir novas cartas, com mecânicas próprias, para incrementar o leque de possibilidades de formação de novos *decks*.

A implementação do projeto conta com a biblioteca gráfica Allegro (Allegro 5) para a plotagem de imagens, utilizando a linguagem C para gerar a *engine*. A escolha destes está implicada na necessidade em obter um programa final ágil e um código onde o programador possa ter maior controle sobre os dados que se passam no ambiente.

A implementação do gerador de *cards* do usuário é feita usando a linguagem Lua (simples e eficiente). Usuários poderão criar novos *cards*, bastando para isso conhecer apenas o básico de programação Lua e analisando as estruturas internas dos *cards* de exemplo.

Com a conclusão deste projeto espera-se obter um produto final de qualidade, com pelo menos as seguintes características: mecânicas que permitam uma boa experiência para os usuários (loja virtual, nivelamento de conta) e um *framework* de edição de cartas bem polido, de entendimento simples e que gere os resultados esperados para qualquer cenário esperado.

Palavras-chave: *TCG, gerador de cards, jogo inédito*.

ABSTRACT

On the last edition, a fully automated engine for a brand-new Trading Card Game (TCG) was proposed and developed. Cards that, when activated in a match, would generate certain game conditions; a card collection for each entity defined as *profiles* (containing every user data, such as *deck*, match performance, etc.), development of a friendly UI (*User Interface*); these were the main objectives approached under the labors at POC I.

That being said, the main objetive for this iteration of the project is to add a new feature on the ongoing system: make this game engine more flexible. The user of the framework will be able to define new cards, with effects of interest, to further expand the range of possibilities on how new decks can be formed.

This project implementation is equipped with the Allegro 5 game programming library for the image rendering task, under the C programming language for engine codes. This choice of implementation was made under the need of a agile final product and a code on which the coder can rely on for control on the environment data.

The card generator implementation uses the Lua scripting language (lightweight and effective). Users will be able to create new cards, requiring only the basics of Lua and analysing the internal file structures of the sample cards provided.

At the end of this project, it is expected to obtain a good quality end-product, having at least these features: mechanics that can bring entertainment for the users (card shop, profile leveling) and a well-developed card editor framework, with straightforward learning curve and generates the expected outcomes for any expected scenery.

Keywords: *TCG, card generator, brand-new game*.

LISTA DE FIGURAS

LISTA DE SIGLAS

TCG	Trading Card Game
ZTCG:HS	Zelda Trading Card Game: Hyrule Showdown

LISTA DE ATRIBUIÇÕES

POINTCUTS	Pontos de entrada de código do usuário (cards customizados)

SUMÁRIO

1 INTRODUÇÃO

Conhecimento, para muitos, é definido como a capacidade de estar continuamente se adaptando a novas experiências. É importante tomar decisões quanto a qual sub-área do conhecimento se quer aprofundar, pelo bem da concisão. Contudo, ao invés de ficar focado em algum ponto (por mais promissor que ele seja), é ainda mais importante tentar abordar e permitir-se contemplar sempre algo novo. Afinal, são as perturbações no ambiente que permitem a inovação e, consequentemente, a evolução do conhecimento. É natural. E dentro do ambiente de desenvolvimento de softwares não é diferente.

No cenário do desenvolvimento de software atual, qualquer aplicativo que permita ou proporcione ao usuário a capacidade de inovar possui grandes vantagens se comparado a um programa equivalente, cujo conteúdo não permita tal flexibilidade.

E em desenvolvimento de jogos também não é diferente. Os *mods*, que permitem aos jogadores elaborarem seus próprios conceitos dentro de um jogo e assim aumentar potencialmente a chamada *replayability* (ou seja, a capacidade de um jogo ser jogado por um período além do esperado pelos desenvolvedores, sem perder a "graça"), são os maiores exemplos de metodologias que, quando aplicadas a um *software*, faz subir a popularidade do mesmo radicalmente.

Grandes exemplos de jogos populares são World of Warcraft, DotA, Quake, TES: Skyrim. Estes são remarcáveis pelo uso de *mods*, que têm garantido continuamente novidades na jogabilidade de cada um deles.

A proposta para este projeto é incrementar o trabalho desenvolvido na edição passada (POC I), elaborando um *framework* que permita *mods* de cards no sistema. Os primeiros 200 *cards* implementados são fixos (não alteráveis), mas novos cards a serem adicionados são completamente livres para implementação pelo usuário do *framework.*

2 DISCLAIMER

O ZTCG:HS é baseado no *best-seller The Legend of Zelda®: Ocarina of Time®*, da Nintendo®. Imagens e nomes “emprestados” são creditados aos seus respectivos donos, *design* dos atributos e habilidades das cartas são de minha autoria.

3 DESENVOLVIMENTO DO TRABALHO

O desenvolvimento deste trabalho se deu em quatro etapas principais:

- Normalização do código-fonte
- Elaboração da API do gerador de cards
- Elaboração do parser do gerador de cards
- Elaboração dos pointcuts

1. Normalização do Código-fonte

Ao término do desenvolvimento do jogo (final do POC I), haviam trechos do código que foi implementado de forma *hard-coded*, ou seja, haviam trechos de código que implementava o sistema de forma não-flexível (atendia somente os propósitos específicos do jogo).

Para resolver, parte do sistema foi profundamente reanalisado e medidas vitais foram tomadas. Dentre estas medidas estão: criar uma tabela de enums (de codificação própria, pois Lua não implementa bem enumeradores), destacar locais de códigos onde pointcuts devem ser inseridos, uso de variáveis globais e de *defines* para pontos do código onde um valor fixo deve ser inserido (normalmente trechos de alocação estática), entre outras medidas.

2. Elaboração da API do gerador de cards

Nesta etapa, funções do sistema original que implementavam comportamentos numa partida (ações como comprar carta, atacar oponente) foram extraídas e reusadas em um *wrapper* em Lua implementado. Basicamente, aquele que for implementar novos *cards* referenciarão em seu *script* somente funções definidas no *wrapper*. O que estas funções farão, ao serem chamadas pelo usuário, é ler o estado atual do jogo e modificá-lo de acordo suas próprias regras internas.

Existem mais de 70 funções nesta API, cada uma implementando uma mecânica esperada do jogo original. Utilizando uma combinação deste leque de funções, é possível implementar quase qualquer tipo de card esperado. No entanto, algumas regras ainda

precisam ser observadas ao implementar: alguns dos *pointcuts* só podem ser acessados por um dado tipo de *card*. Isto não será problema na grande maior parte dos casos.

3. Elaboração do parser do gerador de cards

Nesta etapa foi definido o "esqueleto" dos scripts de cards em Lua. O arquivo que define cards não segue código Lua original (facilitando a vida do usuário menos experiente). Ele tem a natureza de um arquivo XML, somente a implementação das funções são em Lua (ver detalhes nos cards de exemplo).

Todas as pendências e unificações de código Lua e C foram realizadas nesta etapa. Diversas passadas pelo programa intermediário (combinação do parser com o jogo do POC I) foram efetuadas, até garantir que não houvesse falha alguma que tivesse surgido com a unificação.

4. Elaboração dos pointcuts

A etapa final da implementação consistiu em determinar os trechos do código onde implementação *hard-coded* próprio de um card foi realizada. Nenhuma modificação foi realizada para retirar a implementação *hard-coded*, somente referências a *pointcuts* foram inseridas nas posições detectadas.

O usuário do *framework* precisa ter uma noção de como estes *pointcuts* atuam e onde eles se situam, para que os resultados esperados sejam atingidos. Por isso, todos os *pointcuts* tiveram seus nomes definidos de forma que facilite o entendimento do usuário sobre como e onde estes atuam.

Ver a seção "Documentação do Framework" para maiores detalhes a respeito dos nomes e conceitos utilizados pelo *framework*.

4 CARD DESIGN

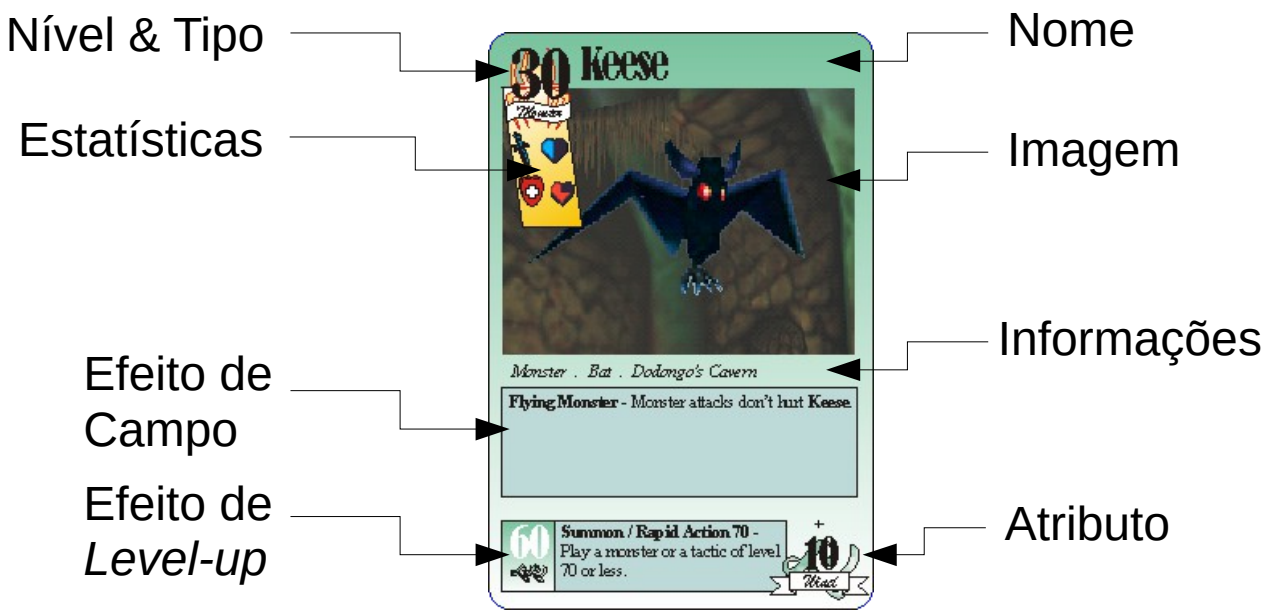

Figura 1: *Card Design.*

4.1.1 Atributos do *Card*

4.1.1.1 Nível & Tipo

Nível: métrica que define o quão simples (ou complexo) será colocar este card em jogo. Geralmente cards de níveis mais baixos são fáceis de chamar, contudo seu impacto no jogo é menor.

Tipo: define a família da qual o *card* pertence. A seção “Tipos de Card” detalhará os tipos presentes.

4.1.1.2 Nome

Identificador do card.

4.1.1.3 Imagem

Figura do card, representa alguma entidade do jogo na qual este TCG foi baseado.

4.1.1.4 Estatísticas

Presentes nos cards do tipo *Mob, Jr. Boss* e *Boss*. Mostram ataque e vida bases do *card*. Importante notar: em certos *cards* (como naquele mostrado na Figura 1), informações referentes à contabilização de dano e cura podem ambos estar marcados como frações de coração OU múltiplos de 10. Neste caso, considera-se que cada ¼ de coração é igual a 10 HP (Pontos de Vida).

4.1.1.5 Atributo

O elemento no qual o *card* é baseado. Ele define vários aspectos do *gameplay* para cada carta no *deck*. Detalhado na seção "Atributos".

4.1.1.6 Informações

Fornece em sequência: tipo do *card*, subtipo (geralmente é alguma característica peculiar a um grupo seleto de *cards*) e, para certos casos, o local dentro do jogo onde este TCG se baseia que pode-se encontrar a instância representada pela carta.

4.1.1.7 Efeito de Campo

Efeito(s) que toma(m) posse no jogo quando o *card* é chamado em campo (note que *chamar um card* é diferente de *consumí-lo* para aumentar o nível do seu personagem).

4.1.1.8 Efeito de *Level-up*

Habilidade extra que o seu personagem ganha ao consumir um *card* durante a fase de "Level-up". Esta fase será melhor detalhada adiante.

4.1.2 Tipos de Card

4.1.2.1 Character

Personagem que representa o jogador em uma *match*. A sua importância reside na necessidade de sua prevalência ao final do jogo, para que o jogador seja declarado vencedor.

Inicialmente não possui nenhuma habilidade relevante, mas a medida que a partida progride, suas habilidades básicas ou novas habilidades vão sendo destravadas. Habilidades como: comprar carta, jogar carta em campo, atacar o oponente, etc.

4.1.2.2 Mob

Personagens secundários, que, quando em campo, ativamente ajudam seu mestre. Não há necessidade direta de mantê-lo vivo até o final do jogo, contudo a sua presença constante garante um bom desempenho no jogo para seu dono. A característica comum dos mobs é retomar o HP original em todo início de turno.

4.1.2.3 Equipment

Ferramentas cuja finalidade é suportar seu usuário em combate. Não pode ser destruído por meios comuns pelo adversário, nem (geralmente) colaboram ativamente para a derrota de seu adversário.

4.1.2.4 Tactic

Ações instantâneas. O efeito que tal carta proporciona geralmente tem um impacto vital no desempenho dos jogadores, podendo até garantir viradas em uma partida aparentemente já decidida.

4.1.2.5 Field

Campo. Define o “local” onde a partida está ocorrendo para o jogador. Analíticamente falando, o campo garante certos bônus para *cards* que estão sobre sua influência.

4.1.2.6 Bosses

Mobs especiais. Têm um impacto bem mais importante no jogo que os *mobs* normais, contudo não compartilha da recuperação de vida a todo início de turno; porém sua vida usualmente alta compensa esta falta.

4.1.3 Atributos

Elementos primários de toda carta do jogo. Para o jogador, a escolha de *cards* no *deck* precisa ser coerente com os atributos principais do deck. É aconselhável manter um *deck* com 2 atrbutos principais, para se adequar às seguintes restrições referentes a atributos:

- Nível 2 ou 3 em um respectivo atributo para que a maioria das habilidades sejam ativadas.
- Para jogar uma carta em campo, o jogador precisa ter pelo menos UM nível no atributo primário da carta requisitada.

Atributos & especialidades:

- *earth* – mob / ataque básico
- *wind* – equip / ataque básico
- *fire* – tactic / dano
- *water* – equip / defesa
- *dark* – mob / dinamismo de jogo
- *light* – tactic / cura

Figura 2: Atributos.

5 MENUS IN-GAME

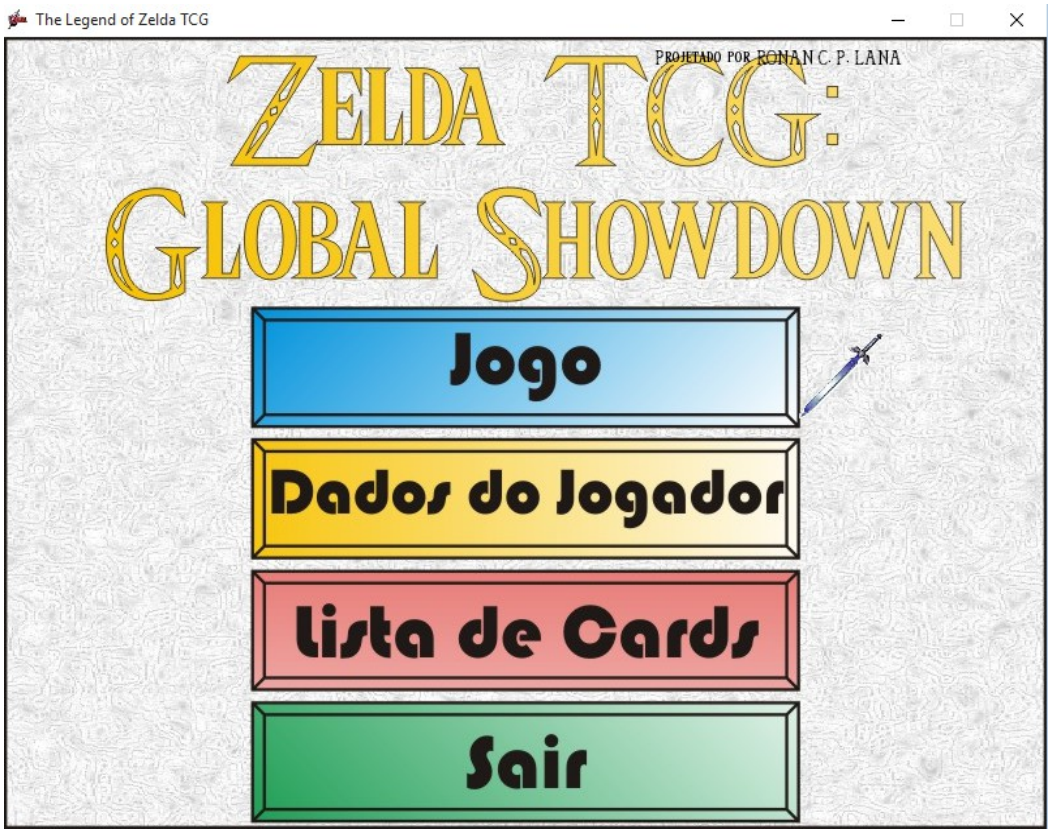

Figura 3: Menu inicial.

5.1.1 Menu Inicial

A tela inicial do jogo. Consiste nos 4 (quatro) botões descritos abaixo:

5.1.1.1 Iniciar Jogo

Abre um lobby local, onde dois jogadores irão se confrontar trocando turnos numa mesma máquina. Não requer acesso à Internet.

5.1.1.2 Dados do Jogador

Abre um próximo menu cuja finalidade é auto-descritivo. Nela há opções como “Criar Profile”, “Administrar Profile” e “Deletar Profile” (bem como retornar ao Menu Inicial, mas é irrelevante continuar citando isto, pois todo menu precisa ter algum botão de “escape”).

5.1.1.3 Lista de Cards

Abre uma janela que lista todos os cards existentes de um *Profile*.

5.1.1.4 Sair

Encerra o aplicativo.

5.1.2 Dados do Jogador

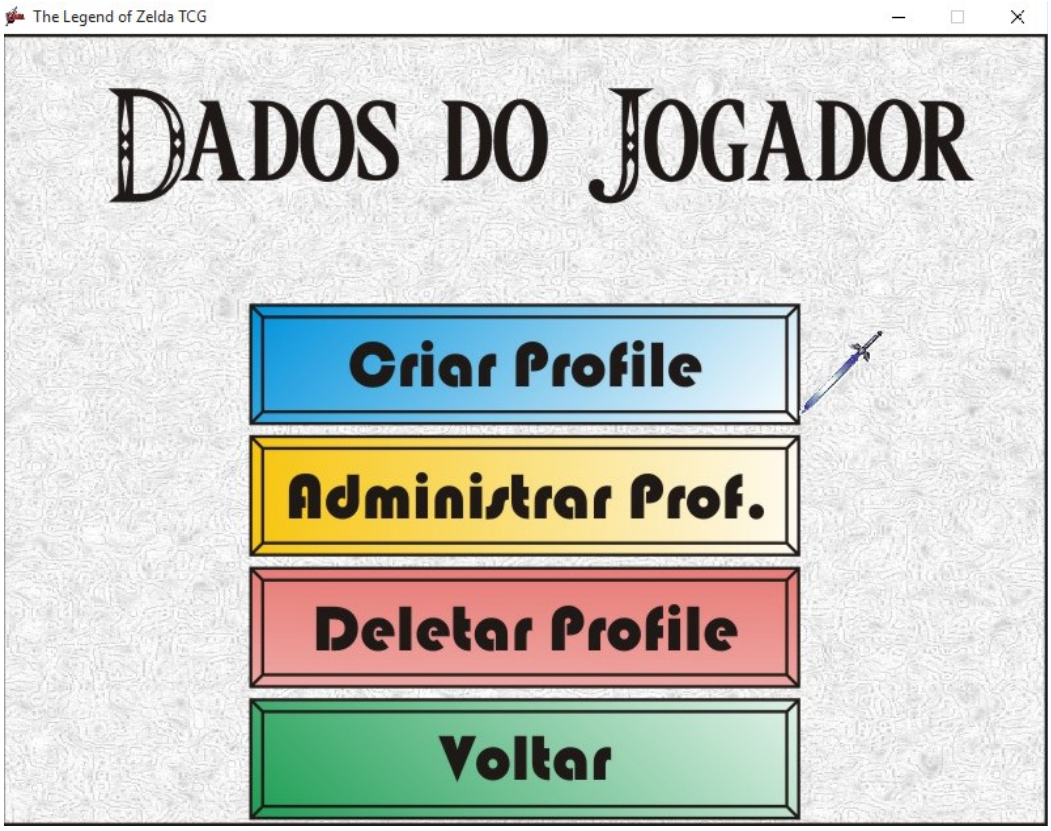

Figura 4: Dados do Jogador.

Este menu permite realizar operações tais como criar um *profile* local, gerenciar os dados de um *profile* e deletar um *profile* local.

5.1.2.1 Criar Profile

Instala no disco um novo *profile* local. Serão requisitados do usuário parâmetros como:

- nome (chave única, 20 caracteres alfanuméricos no máximo);
- dupla de atributos característica para o *deck*.

O botão “Voltar” ignora o processo a ser realizado e retorna para o Menu Inicial. Já o botão “Concluir” salva o processo, dado uma combinação viável de parâmetros foi definida pelo usuário.

5.1.2.2 Administrar Profile

Após selecionado um *profile*, uma tela mostrando todas as cartas do *Side Deck*, *Main Deck* e *Character Deck* é aberta.

O *Side Deck* contém todas as cartas que o jogador possui mas que não estejam no *Main Deck* e nem no *Character Deck* dele, portanto espera-se que a maior parte dos cards existentes esteja concentrado neste setor. Desta forma, o *Side Deck* é fixo para todos os própositos da administração do *profile*.

O botão "Trocar" está disponível para transferir cards em ambos os sentidos do *Side Deck* para o setor escolhido, contudo é preciso estar atento para não transferir carta inválida para o tipo do deck; do contrário nada acontece. Clicar uma vez sobre uma carta irá selecioná-la (borda dourada), indicando que ela está pronta para ser transferida; clicar de novo irá retirar a seleção.

As abas em cor dourada permitem o acesso aos outros tipos de deck do *profile* e à loja virtual.

Finalizada todas as ações, clique em "APLICAR" para registrar as alterações realizadas, ou em "Voltar" para descartar todas as alterações feitas.

5.1.2.3 Deletar Profile

Abre uma janela que lista todos os *profiles* locais. Selecione um e clique sobre "Deletar" para efetivar a ação. Uma vez deletado, não será mais possível recuperar os dados do *profile*.

5.1.3 Lista de Cards

Ao acessá-la, uma tela de escolha de profile se abrirá. Nela, o usuário escolhe um entre a lista de todos os profiles instalados no disco da máquina ou o profile associado à conta dele.

Após feita a escolha, o usuário poderá contemplar todas as cartas que tal profile possui (cartas não reveladas indicam que não existe tal carta no repertório do profile). As cartas listadas são filtradas por Tipo, bastando clicar sobre as abas na parte superior da tela para acessar os variados tipos.

Clicando sobre uma carta revelada permite o usuário ler na íntegra as informações relevantes da carta selecionada, disponível na porção lateral esquerda da tela.

Figura 5: Lista de *Cards*.

5.1.4 Loja Virtual

É considerada uma das *features* especiais deste jogo. Observe que um dos objetivos mais proeminentes de todo jogo de cartas colecionáveis (TCG) é possuir pelo menos uma cópia de todas as cartas do jogo.

A loja virtual é um mecanismo de apoio para atingir este objetivo: a medida que um *profile* vai realizando partidas, o nível dele sobe, bem como as oportunidades de adquirir novas cartas e moedas ao longo do tempo.

As moedas podem ser trocadas por cartas disponibilizadas na loja para aquele *profile* (ou seja, cartas presentes na loja estão vinculadas por *profile*, o que permite que *profiles* de alto nível consigam cartas mais raras por preços mais acessíveis). Além disso, é possível vender cartas para a loja (embora que por metade do preço).

Figura 6: Loja Virtual.

5.1.1 Iniciar Jogo

É o menu de “matchmaking” para partidas em máquina local, onde novamente não é necessário ter conexão à Internet para jogar.

5.1.1.1 Escolha dos Players

Cada jogador entra com um *Profile* existente e escolhe um card do tipo Character que irá representá-lo durante a partida. Não é possível iniciar uma partida enquanto esta etapa não for realizada com sucesso para ambos jogadores.

Para escolher, primeiro clique sobre o nome de um *profile* existente. O nome dele deverá aparecer no interior do *box* “*Profile* Atual”. Em seguida, clique neste *box* para autenticar a sua escolha.

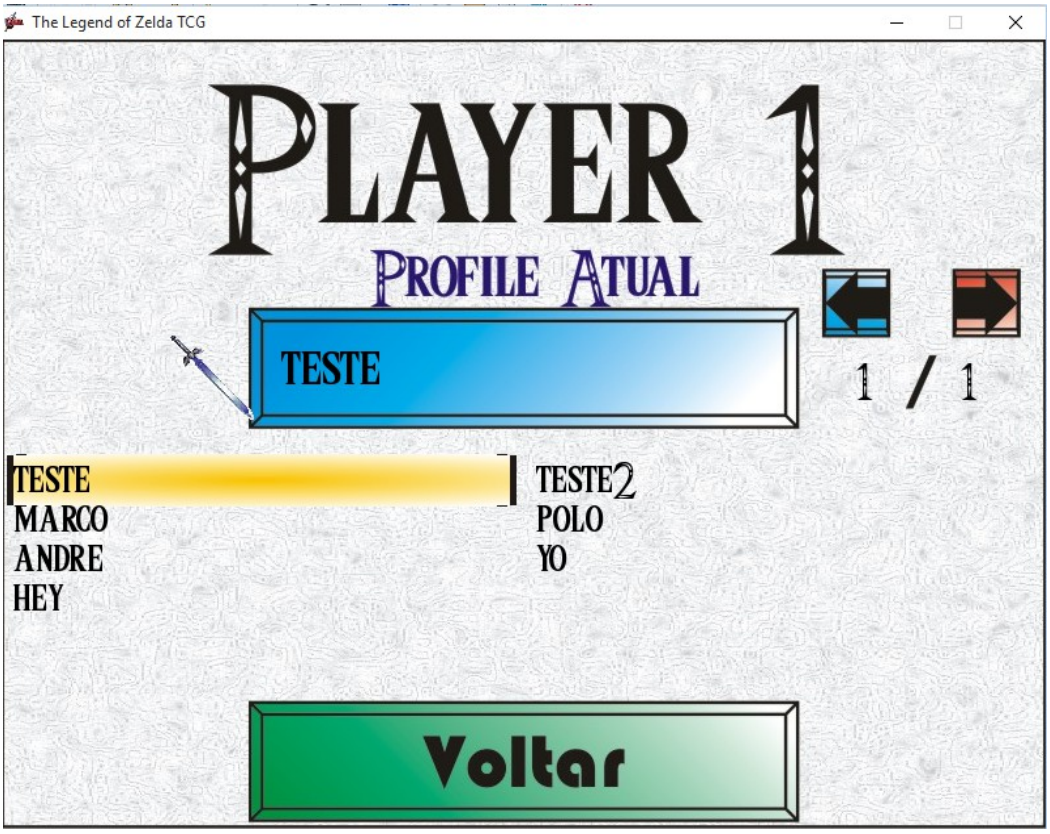

Figura 7: Escolha de *Profile*.

5.1.1.2 Iniciar Partida

Ao clicar este botão dá-se início a uma nova partida, e quaisquer premiações ou punições decorrentes do resultado da mesma é aplicada para os *Profiles* atores.

6 REGRAS DO JOGO

6.1.1 Penalizações por Abandono

Logo ao observar a tela da partida, é possível observar um botão “Sair” no canto superior. Deve-se notar, no entanto, que uma vez que o jogador sai da partida ele é declarado derrotado pelo sistema e é concedido ao outro jogador a vitória. Além disso, diferentemente de perder o jogo atingindo 0 *Hit Points* (HP), o *Profile* do jogador terá alguns dos pontos de experiência deduzidos (sem, no entanto, cair de nível) e o contador de abandono subirá uma unidade, além da de derrota.

6.1.2 Premiações da Partida

Premiações serão efetuadas para os jogadores que perseverarem até o final da partida, na forma de ganho de experiência (Exp) e possibilidade de “encontrar” uma nova carta.

6.1.3 Interface do Jogo

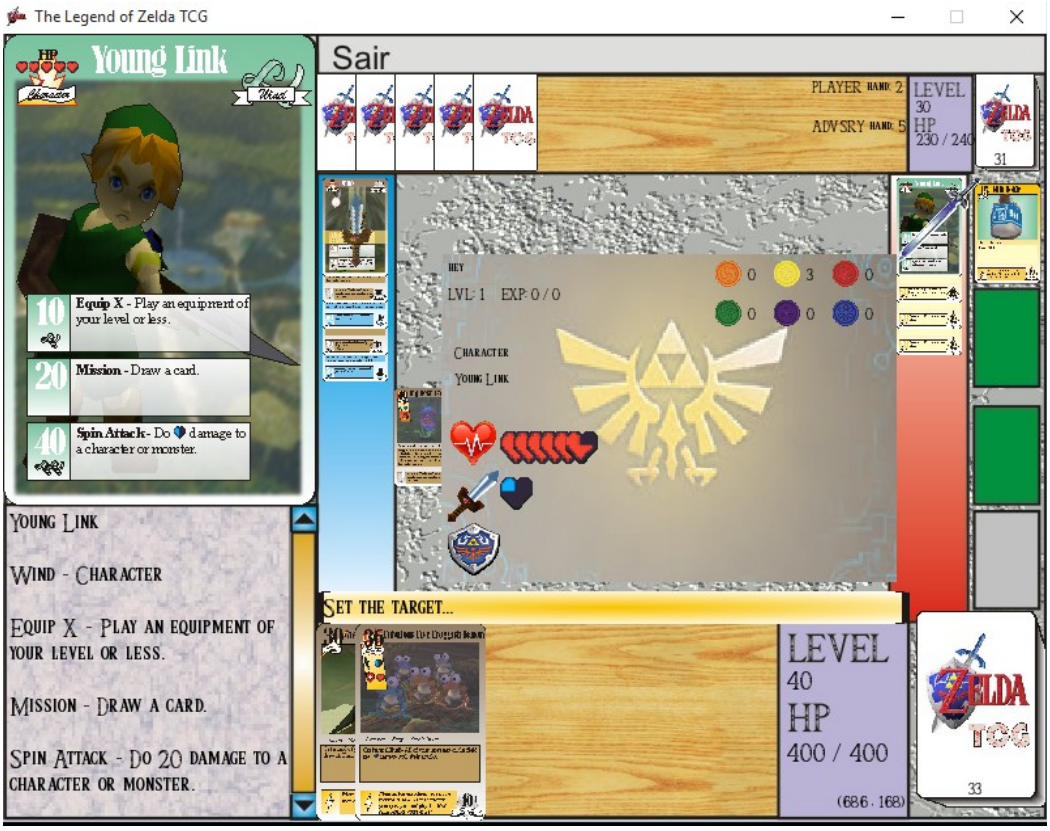

Figura 8: Interface do jogo.

A seção esquerda da tela está relacionada a informações pertinentes ao último *card* cujo cursor foi passado sobre. Certos *cards* podem conter textos muito longos (o que pode tornar a leitura inapropriada); sendo assim, o setor inferior à imagem da carta trata de mostrar, na íntegra, todo o texto presente, utilizando um mecanismo de barra de rolagem para auxiliar a leitura.

A seção inferior contém informações pertinente ao estado do jogo do jogador (o ator desta rodada, a sua contra-parte é o oponente): mão, campo de batalha, HUD (pontos de vida, nível do seu *character*), *deck*, *graveyard* (em cinza, acima do *deck*), *field* (em verde).

A seção superior é a contra-parte: pertençe ao adversário do ator desta rodada, ao outro jogador. É possível analisar todas as cartas que o adversário possui em campo e no *graveyard*.

Observe que há uma carta presente no campo de batalha: é possível notar que ela pertença ao jogador, uma vez que ela está situada mais próxima à parte inferior da interface (que por sua vez representa o jogador).

Repare a caixa que está aparecendo na Figura 5.3.1. Ela está dando diversas informações adicionais sobre o *character* do adversário:

- nome e nível do *profile* (somente cartas *character*);
- tipo e nome da carta;
- pontos de vida atuais (vermelho), pontos de dano (azul), bloqueio (vermelho).

O jogador está num caso de decisão de ataque:

- carta oponente possui 5 (corações inteiros) * 40 (HP / coração) + 30 = 230 HP
- dano aplicado = 10
- bloqueio = 0
- HP após ataque = 230 – (10 – 0) = 220

Pode parecer pouco o dano infligido por essa ação, porém vale ressaltar que existem diversos *buffs* neste TCG, capazes de aumentar consideravelmente poderes de ataque e de defesa de ambos *characters* e *mobs*, tornando o jogo mais estratégico e desafiador.

6.1.4 Mecânicas do Jogo

6.1.4.1 Turnos e rodadas

Caracteriza-se como turno todo o conjunto de ações relevantes para a alteração do estado do jogo realizado por ambos jogadores no decorrer de duas rodadas (rodada do jogador seguida da do adversário, ou vice-versa).

Caracteriza-se como rodada todo o momento onde o foco do jogo é passado para um dos jogadores, este que detêm o título de “ator” da rodada. Note que o outro jogador ainda poderá atuar, mesmo que fora da rodada dele, caso isso seja permitido por alguma dinâmica existente na partida.

6.1.4.2 Cancelamento de Ação

A maioria das ações podem ser renunciadas pelo jogador ao pressionar o botão direito do mouse. Haverão momentos em que renunciar uma ação será a melhor estratégia que o jogador poderá tomar.

6.1.4.3 Fases da rodada

As fases estão listadas na ordem que devem ser realizadas durante uma rodada. São elas:

6.1.4.3.1 Nivelamento (Level Up phase)

O ator escolhe uma carta em sua mão para consumir e aumentar em 10 o nível de seu Character. A carta consumida vai para baixo da pilha de cartas consumidas, ou seja, ela vai pra baixo do card Character.

Se o card consumido possui uma habilidade de efeito instantâneo, ela toma ação ainda nesta fase. Habilidades de efeito instantâneo são usáveis somente no momento do consumo da carta.

6.1.4.3.2 Ações do Personagem (Character Action phase)

Desde a primeira habilidade do personagem e seguindo ordem de consumo das cartas, o jogador efetua as habilidades adquiridas pelo personagem cujo mínimo de requerimentos exigidos é satisfeito. Para a maioria das habilidades o jogador pode escolher ativar ou não, contudo existem habilidades que não permitem este tipo de liberdade. Exemplo: comprar carta (Draw Card).

6.1.4.3.3 Ações dos Mobs (Mob phase)

Todos os Mobs do ator da rodada têm seus efeitos ativados, ativação esta que pode ser limitada se a carta descrever alguma limitação. Observe que efeitos de Aura dos Mobs são computados enquanto tais Mobs estiverem presentes em campo e, portanto, não caracteriza ativação de efeito.

6.1.4.3.4 Ataques dos Mobs (Mob Attack phase)

Salvo por situações descritas durante o *gameplay* da partida, todo Mob do ator da rodada tem direito a uma instância de ataque contra um Character ou Mob oponente. O ator escolhe se resolve atacar com o Mob ou não.

6.1.4.4 Finalização da partida

A partida se encerra quando algum dos jogadores atinge 0 (zero) de HP, considerando-se assim derrota para este, e vitória para o outro. Se for o caso de ambos os jogadores atingirem o zero ao mesmo tempo, o jogo é considerado um empate. A desistência de algum jogador também leva à finalização da partida, garantindo vitória para aquele que permaneceu na partida.

6.1.4.5 Mecânicas de Carta

Muitas das cartas possuem efeitos únicos; mesmo assim, é possível discernir certos padrões nos efeitos das mesmas. Os mais proeminentes caracterizam um comportamento natural a todas as cartas que compartilham destes padrões e, portanto, podem ser definidas como segue:

6.1.4.5.1 Aura

Basicamente o efeito descrito na carta tem um caráter abrangente para ela e outras cartas enquanto ela estiver presente em campo. Geralmente é um bônus/ônus nos valores de HP ou de ataque dos personagens e mobs, mas pode também ser garantia de certos privilégios a outras cartas no campo.

6.1.4.5.2 Stun

Aplicável somente em mobs. Se um mob está paralisado, ele se torna inapto a atacar o adversário (observe que ataques vindo de certas habilidades ainda podem ser realizados).

6.1.4.5.3 Silence

Aplicável somente em mobs. Se um mob está silenciado, ele se torna inapto em ativar suas habilidades, se ele possuir alguma. Também se aplica sobre efeitos de Aura.

6.1.4.5.4 Damage Over Turn (DOT)

Se um personagem ou mob está envenenado, no início da rodada do adversário o alvo irá sofrer dano direto de valor pré-determinado. A cada turno o contador é decrementado uma unidade, ou seja, geralmente DOT tem efeito finito.

6.1.4.5.5 Draw Card

Algumas cartas possuem efeitos que sobrescrevem a ação de comprar uma carta, contudo geralmente o efeito final desta ação é adquirir uma nova carta para a mão vinda do deck.

6.1.4.5.6 Prevent Damage

Anula todo o dano de uma instância de ataque sobre uma carta-alvo, na prática o atacante inflige zero de dano no alvo.

6.1.4.5.7 Reveal Card

Torna a carta a ser revelada pública, ou seja, ambos os jogadores tomam conhecimento da carta revelada.

6.1.4.5.8 Peek Card

Revela a carta para si e retorna a carta para a posição que ela estava. Geralmente, a carta revelada é a do topo do deck do autor da ação.

6.1.4.5.9 Withdraw

É a ação de retirar temporariamente uma carta do campo de batalha. Por retirar, interpreta-se que esta carta não pode receber dano proveniente de nenhum ataque de um Character ou um Mob, e nem pode ser destruído diretamente, somente por instâncias de dano que o alvo já tenha contraído.

7 DOCUMENTAÇÃO DO FRAMEWORK

Detalhes referentes à API a ser utilizada pelo usuário está em anexo, junto com outros documentos de referência do projeto. (Definição do texto detalhado ainda está em fase de desenvolvimento).

Lista de docs:

- *pointcuts* – definição dos pontos onde o código do usuário pode ser ativado.
- *lua_wrapper_documented* – definição das funções usadas no *framework*.

Para a inclusão de novos cards no sistema, referencie-se aos seguintes tópicos:

- *enum.lua* – o usuário pode definir novos valores ao sistema o quanto for preciso, podendo então ser usados em seus descritores de cards. Este arquivo deve ser mantido sempre em concordância com a linguagem Lua. IMPORTANTE: alteração de valores ou nomes que foram pré-definidos no arquivo pode impactar de forma inesperada no sistema.

- *cards/metadata.txt* – ao final do arquivo, insira uma nova linha com somente o nome do arquivo da nova carta, sem extensão (“.txt”) e sem caminho de diretório (“/cards/...”).

- DESCRIÇÃO DO CARD: extensão LUA, mesmo nome descrito no *metadata* e dentro de “cards/info”. Tem que estar no formato descrito pelo parser do sistema. Observar exemplos de cards.

1. **Exemplo de descritor de card: Character**

```
ZTCG_CARD
{
   "NAME" "Avatar"
   "IMAGE" "base_card.png"
   "TYPE" "CHAR"
   "ELEMENT" "WIND"
   "RARITY" "7"
   "INFO" "random info"
   "COST" "777"

   TYPE_CHAR
   {
      LVL_ACTION
      {
         "LEVEL" "10"
         "ATTRB" "0"
         "TEXT" "Mission - Draw Card"
      }

      LVL_ACTION
      {
         "LEVEL" "100"
         "ATTRB" "3"
         "TEXT" "OP Attack - Do 100 damage."
      }

      LVL_ACTION
      {
         "LEVEL" "50"
         "ATTRB" "1"
         "TEXT" "Attack - Do 10 damage to all."
```

```
        }

        "HP" "20"
    }

    function onReceiveAttack(player, gp)
        levelUp(player,gp);
    end

    function onDrawCard(player,gp)
        recoverTopCard(player,gp);
    end

}
```

8 PERFIL ATUAL

No estado atual do desenvolvimento deste TCG virtual, é possível jogar (somente *off-line*) uma partida completa entre dois *profiles* instalados na máquina. Um conjunto de 200 cards já foram implementados e testados, obtendo sempre resultados esperados.

Além do sistema das partidas, o jogo já conta com interfaces de menus, em Inglês, que permitem ao usuário gerenciar seus *profiles* (movimentar cartas entre os *decks* principais e de reserva), compra e venda de cartas para o sistema, criar novos *profiles* e deletar *profiles* já instalados.

O *framework* implementado está atualmente em fase de testes para os seguintes contextos: API, *pointcuts*. A tabela de enums (que será usada extensivamente pelo usuário) está funcionando perfeitamente; além do parser de cards customizados, que consegue ler arquivos descritores de *cards* de forma flexível.

9 CONTEXTUALIZAÇÃO E TRABALHOS RELACIONADOS

Este jogo de cartas possui, em suas mecânicas, influências de regras de vários outros jogos de cartas TCG relacionados. Vale a pena ressaltar o TCG cuja infuência foi bem mais significativa: MapleStory® iTCG®, da Nexon® em conjunto com a Wizards of the Coast®. O manual se encontra disponível *on-line* como relatado em [1].

Outra influência é no *design* da interface deste jogo: baseado no jogo eletrônico [2], própria da Konami®.

10 RESULTADOS E DISCUSSÃO

O programa desenvolvido e o código-fonte do projeto podem ser encontrados no seguinte link: “https://github.com/ronancpl/ztcg_hs”.

Para compilar e rodar o programa, é necessário que Allegro 5 e Lua 5.3 estejam devidamente instalados em sua máquina. Refira-se a [3] para a instalação do Allegro. Para a instalação do Lua, refira-se à homepage do mesmo.

Os resultados encontrados são satisfatórios: os menus do jogo rodam sem problemas, os 200 cards originais estão funcionando corretamente, o programa consegue processar todos os dados de cards customizados, passando dados dos arquivos para a memória principal sem problemas. O que está faltando agora é devidamente testar e verificar existência de *bugs* no *framework* elaborado.

11 CONCLUSÃO E TRABALHOS FUTUROS

Uma vez que a partida do desenvolvimento deste *software* iniciou-se antes desta temporada (2015-1), no início do POC I o sistema já estava todo montado e funcionando perfeitamente. Para o POC I pude aprofundar na elaboração individual das mecânicas das cartas e do sistema interno do jogo (mecânicas *in-game*).

Durante o POC II, minha meta foi elaborar um *framework* para o jogo TCG desenvolvido anteriormente, o que ocorreu sem maiores problemas, uma vez que todo o sistema foi desenvolvido pensando na possibilidade de poder portá-lo de um jogo fechado para um sistema que permitisse cards customizados, sem muito esforço.

Como trabalho futuro, pretendo consertar *bugs* que podem ter surgido ao longo do desenvolvimento do POC II, além de implementar um esquema de banco de dados que gerencie todos os *profiles* especiais (que são controláveis pelo usuário após autenticados *login* e senha).

12 REFERÊNCIAS

[1] "MapleStory iTCG", publicado e desenvolvido pela Nexon e Wizards of the Coast, PC, 2007.
Manual do jogo disponível em <http://ccggamez.com/files/mpls/maplestory_itcg_rules.pdf>. Acesso em 15 dez. 2015.

[2] "**Yu-Gi-Oh! Power of Chaos: Yugi the Destiny**", publicado e desenvolvido pela Konami, PC, 28 nov. 2003

[3] LIESENBERG, Biz. "Ambiente: Code::Blocks + Allegro 5". Disponível em <https://sites.google.com/a/liesenberg.biz/cjogos/home/software/ambiente-code-blocks-allegro-5>. Acesso em 15 dez. 2015.