Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

Gabriel Merêncio dos Santos

Hugo Mitsumori

Leon Henrique Pires de Brum

Lucas Hideki Sakurai

Shadow Struggles 2013

- Manual Técnico -

Volume único

São Paulo, 2013

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

Gabriel Merêncio dos Santos

Hugo Mitsumori

Leon Henrique Pires de Brum

Lucas Hideki Sakurai

Shadow Struggles 2013

- Manual Técnico -

Volume único

São Paulo, 2013

Projeto apresentado ao curso de

Ensino Médio integrado ao Técnico em

Programação e Desenvolvimento de Sistemas do Instituto Federal de São Paulo, Coordenadoria da Área de Informática, como requisito parcial da disciplina de Prática e Desenvolvimento de Sistemas
Professores Orientadores:

Ivan F. Martinez e Renato Fernandez

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO

Índice de Imagens

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO

Sumário

Glossário

Deck: Conjunto de cartas do jogador.

NPC: Personagem do jogo com a qual é possível interagir.

Perfil: Usuário com o qual um jogador grava seu progresso e acessa o jogo.

Renderizar: Mostrar na tela.

RPG: Estilo de jogo no qual o jogador faz o papel de um personagem, devendo completar os objetivos do jogo com o mesmo, e evoluir suas habilidades.

Settings: Configurações.

Sprite: Imagens do personagem ou ilustração de uma carta em campo

SS: Shadow Struggles.

Tile: Uma unidade de área utilizada no modo batalha do jogo.

Tower Defense: Estilo de jogo no qual se deve defender uma base, ou “torre”.

Visual Novel: Jogo focado no enredo, no qual o jogador acompanha uma história por meio de textos ou de outros recursos.

Introdução

O seguinte manual técnico tem como objetivo fornecer todo o conhecimento técnico e criativo utilizado para o desenvolvimento do projeto Shadow Struggles. Porém, antes do aspecto técnico, é importante que os desenvolvedores conheçam a idéia por trás do jogo.

O projeto Shadow Struggles surgiu da vontade de criar um jogo que unisse aspectos de RPG, Tower Defense e Visual Novel. Permitindo uma jogabilidade divertida e emocionante, com a possibilidade de diferentes estratégias e um progresso não linear ao longo do jogo. Os diferentes caminhos da história escolhidos pelo jogador levam para diferentes locais e inimigos, resultando em diferentes enredos.

O jogador também tem a possibilidade de iniciar batalhas rápidas com seu deck, como em um jogo casual. Assim, pretende-se abranger o maior público possível.

O estilo do jogo também deixa brechas para a implementação de benefícios por dinheiro real, seja pela aquisição de itens exclusivos, melhorias em atributos, visuais diferentes para o personagem, entre muitas outras idéias.

Em resumo, o projeto foi resultado de um grande brainstorm de criatividade e experiência em jogos. Aproveitando-se das tendências atuais e incrementando com diversas características de outros estilos já concebidos, o jogo tem grande potencial no mundo dos jogos, tanto no aspecto móvel e casual quanto no aspecto gamer.

Descrição Técnica Geral

O projeto Shadow Struggles foi desenvolvido em linguagem Java 7, utilizando uma biblioteca especializada no desenvolvimento de jogos, chamada libGdx. A biblioteca também permite a geração de um arquivo executável para plataforma Android. Assim, todo código pode ser executado tanto em computadores, quanto em tablets e smartphones que utilizam o sistema operacional Android, sem necessidade de modificações no código fonte.

É importante manter a biblioteca do libGdx atualizada. Por se tratar de uma biblioteca em constante desenvolvimento, regularmente são feitas atualizações nas classes e novos métodos são criados.
http://libgdx.badlogicgames.com/index.html

Segue o endereço da documentação oficial do libGdx:
http://libgdx.badlogicgames.com/nightlies/docs/api/

Por conta da conversão para aplicação Android, é necessário a utilização exclusiva das classes nativas do libGdx. Classes como ArrayList, Math, entre outras, precisam ser substituídas pelas classes correspondentes do libGdx.

O jogo foi estruturado em MVC (Model Viewer Controller) .

Os dados do jogo, tanto voláteis quanto não voláteis são armazenados em arquivos no formato JSON. O libGdx possui classes para leitura e gravação em formato JSON.

Paralelamente ao projeto principal, foi desenvolvido um outro projeto para editar as informações dos elementos do jogo e gerenciar os recursos utilizados. O projeto (denominado Tools) se mostrou extremamente necessário por conta da grande variedade planejada de itens, cartas, cenas, diálogos, etc. O projeto Tools fornece uma maneira prática e simplificada de adicionar e editar os registros nos arquivos JSON.

Estrutura e Fluxo do Jogo

- Ao iniciar o jogo pela primeira vez, o usuário cria um novo perfil, com atributos e itens padrão.
- Ao iniciar o modo campanha, é iniciado o primeiro capítulo da história.
- Após diversos diálogos, o jogador é direcionado para um tutorial, onde irá aprender a mecânica básica da batalha.
- Após a introdução, o usuário inicia a exploração do mundo, de acordo com o enredo.
- No mapa, é possível interagir com NPC's, encontrar itens, visitar a loja para compra de itens.
- Ao explorar uma área de batalha, a cada passo, há uma pequena chance de iniciar uma batalha automaticamente.
- Após cada batalha, o personagem pode receber itens, e sempre irá receber uma determinada quantidade de experiência de acordo com a dificuldade da batalha.
- Ao atingir a quantidade necessária de experiência, o personagem evolui de nível, ganhando pontos de atributos, que podem ser atribuídos em diferentes qualidades que irão afetar seu desempenho durante as próximas batalhas, como quantidade de vida, poder máximo do deck, possibilidade de saque duplo, entre outras.
- Conforme o usuário for explorando o ambiente, serão ativados diferentes eventos que fazem parte do enredo. Tais eventos podem ser batalhas obrigatórias, diálogos, cenas, entre outros.
- Eventualmente, durante as cenas, o usuário será obrigado a fazer escolhas, que irão determinar diferentes possibilidades de enredo e de progresso.
- Assim, o usuário segue o enredo e vai evoluindo seu personagem através de batalhas e missões.
- Apesar da evolução por níveis, o principal método de aprimoramento é a obtenção de novas cartas.

- Conforme o avanço no enredo, novas cartas são liberadas para a obtenção, seja comprando, ou encontrando aleatoriamente no mapa.
- As cartas liberadas vão ficando cada vez mais poderosas e variadas, permitindo a criação de diversas estratégias de batalha e diferentes decks.

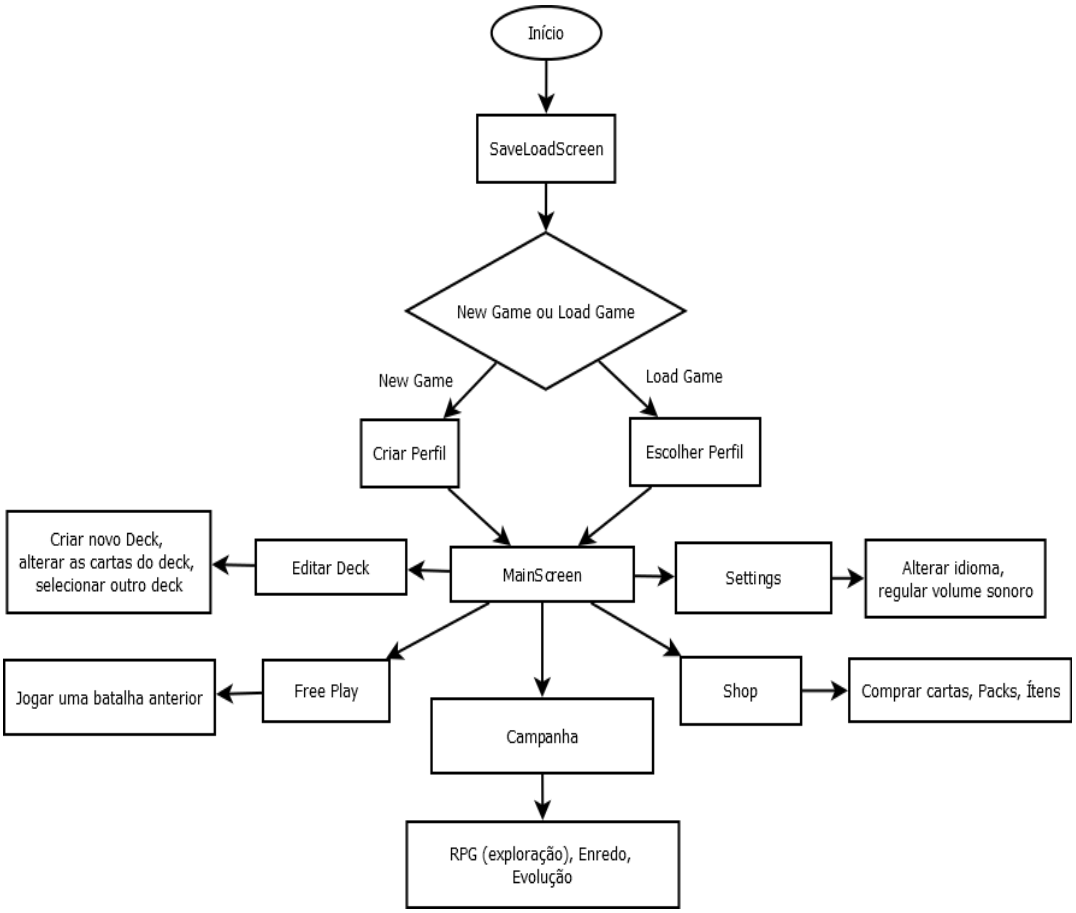

Imagem 1 – Estrutura de ações

Navegação

O LibGdx possui um mecanismo bem facilitado de criação e gerenciamento de telas. A maneira mais simples requer apenas a criação de classes que estendem Game e Screen (ambas nativas do LibGdx)

A criação das telas consiste basicamente na adição de componentes (Actors) na Stage da Screen. O libGdx possui uma boa variedade de Actors, como textBox, Label, Image, TextButton, ComboBox, entre outros.

1 - Ao criar uma classe que estende Screen, são herdados os métodos que compõe o mecanismo da tela: render, resize, dispose, etc . O método render() é chamado automaticamente múltiplas vezes por segundo, compondo a thread principal do jogo.

2 – A classe Screen possui um atributo do tipo Stage, que é onde todos os componentes devem ser adicionados. Ex: stage.addActor(textButton)

3 – Após adicionar todos os componentes no Stage, é preciso definir sua renderização no método render, utilizando os métodos stage.act() e stage.draw().

-BaseScreen

Para facilitar a criação de telas, criamos uma classe abstrata BaseScreen, que contém a configuração padrão de todas as telas do ShadowStruggles, como movimentação da câmera, Skins dos Actors, resize, acesso aos recursos gráficos.

-Skin

Para instanciar um Actor, é necessário passar como argumento uma Skin, que define a aparência visual do componente. O libGdx possui uma Skin padrão para os componentes, mas o ShadowStruggles utiliza uma Skin própria, que se encontra na

pasta **data/images/skin.** As imagens da Skin são atribuídas no arquivo **data/skin.json.**

Para mais informações sobre definições de Skin por JSON, acesse: http://code.google.com/p/libgdx/wiki/Skin

- Texture Atlas

Para melhorar o desempenho do jogo e o gerenciamento de memória, as imagens necessárias são unidas em um TextureAtlas e carregadas na memória RAM apenas quando forem utilizadas.

OBS: O LibGdx só aceita Imagens em resolução de potência de 2. Ex: 64x64, 1024x512, 2048x2048

Armazenamento de Dados

A grande parte das informações que são mostradas ao usuário é armazenada em arquivos no formato Json. Na raiz do projeto, existe uma pasta data, onde ficam contidos todos os dados e recursos (gráficos e sonoros) que serão utilizados.

Existem os dados voláteis e os não-voláteis. Os voláteis são alterados ao longo do jogo, como o progresso do jogo, os atributos do personagem, etc.

Os não-voláteis são as informações e atributos fixos, como diálogos de NPCs, descrição de itens, texto dos menus, entre outros.

Todos os dados voláteis são alterados de acordo com as ações do usuário. Os arquivos que contêm tais dados são lidos e sobre-escritos ao decorrer do progresso do jogo.

Os dados fixos são previamente inseridos no jogo pelo desenvolvedor através do Editor de Recursos (projeto denominado Tools), que gerencia recursos gráficos e sonoros, e permite a criação, edição e exclusão dos arquivos Json. Ao termino da edição, os arquivos são agrupados em uma pasta e compactados no formato ZIP, de

tal modo que estarão prontos para serem exportados (manualmente) para o projeto principal do jogo.

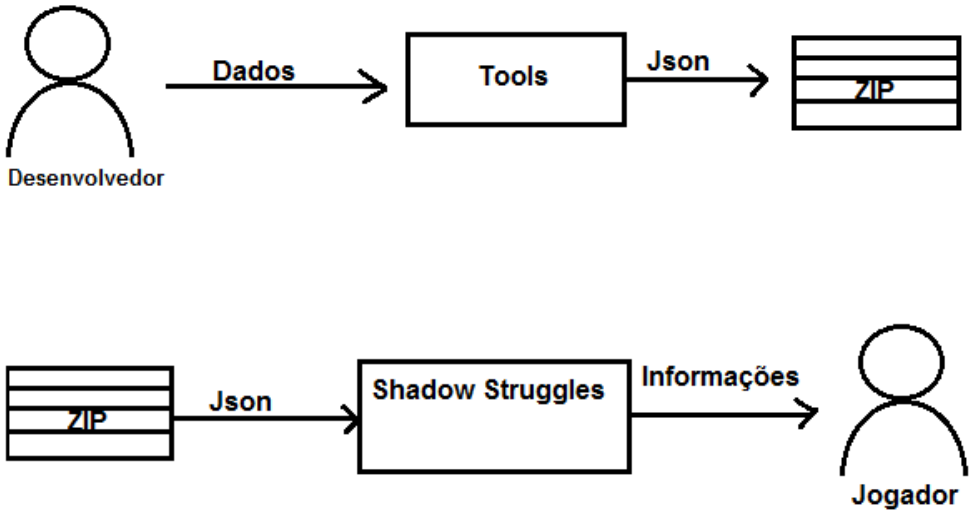

Imagem 2 – Gravação e leitura

Descrição Técnica do Projeto Tools

O projeto Tools foi desenvolvido na linguagem Java 7, utilizando a interface Swing, e estruturas análogas às do LibGdx. A estrutura do projeto em si, é bem parecida com a parte de gerenciamento de dados do projeto principal. O modelo é praticamente o mesmo, e muitas classes do projeto principal, como DataManager, FileMap, também estão presentes no projeto Tools, cumprindo as mesmas funções, mas com algumas adaptações para o objetivo do Tools.

A arquitetura utilizada também foi o MVC, com as funcionalidades e entidades na camada lógica, e a interface de manipulação na camada visual.

O projeto Tools utiliza a mesma estrutura de diretórios do projeto principal, centralizada na pasta ***data,*** de modo que o pacote de arquivos gerado pelo Tools pode ser utilizado normalmente no projeto principal.

Ao criar um novo pacote ZIP, o programa já cria automaticamente a estrutura de pastas e arquivos JSON que serão utilizados pelo projeto. Para cada idioma selecionado, é criada a mesma estrutura de pastas e arquivos dentro da pasta nomeada com o código do idioma.

Apesar de as estruturas e os dados armazenados serem os mesmos utilizados pelo LibGdx, as classes são um pouco diferentes, já que o LibGdx possui uma biblioteca própria para funcionar em todas as plataformas. Para resolver essa diferença de classes análogas no arquivo JSON, foi criada uma classe FileMap que se encarrega de se referenciar aos objetos gravados através de tags, ao invés da referência completa da classe. Assim, são eliminados os problemas de compatibilidade entre as classes nativas do JDK e as classes do LibGdx.

Diagramas de Fluxo de Dados (DFD's)

DFD0

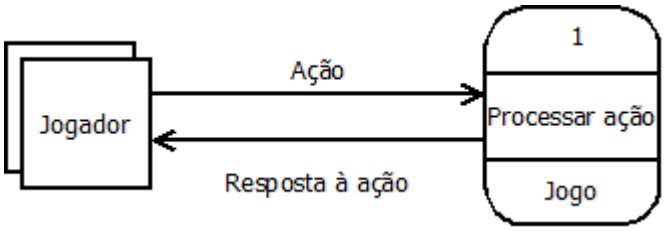

Imagem 3 – DFD0

DFD1

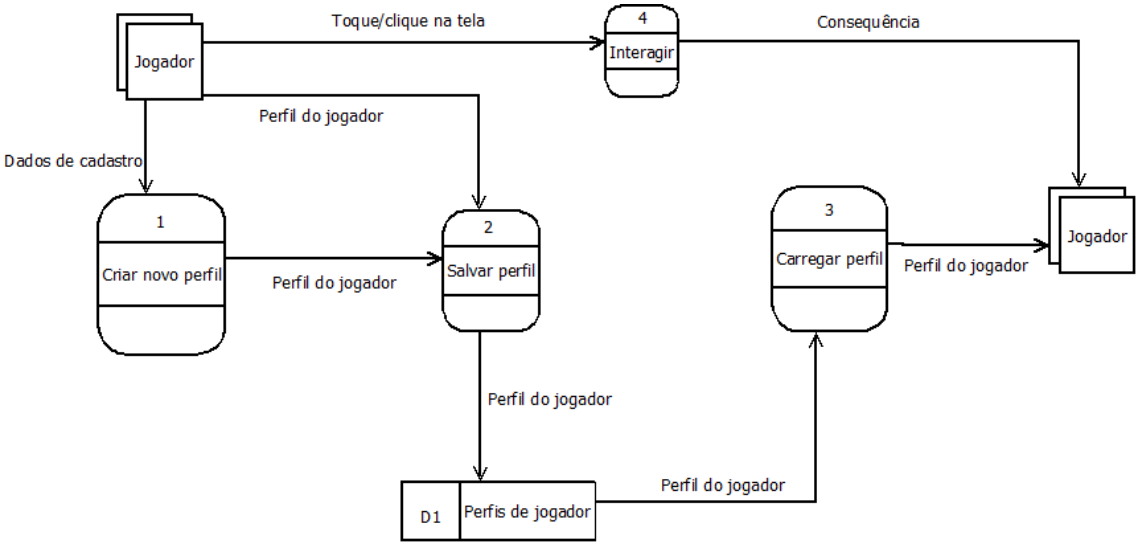

Imagem 4 – DFD1

DFD 2

- Tela Inicial

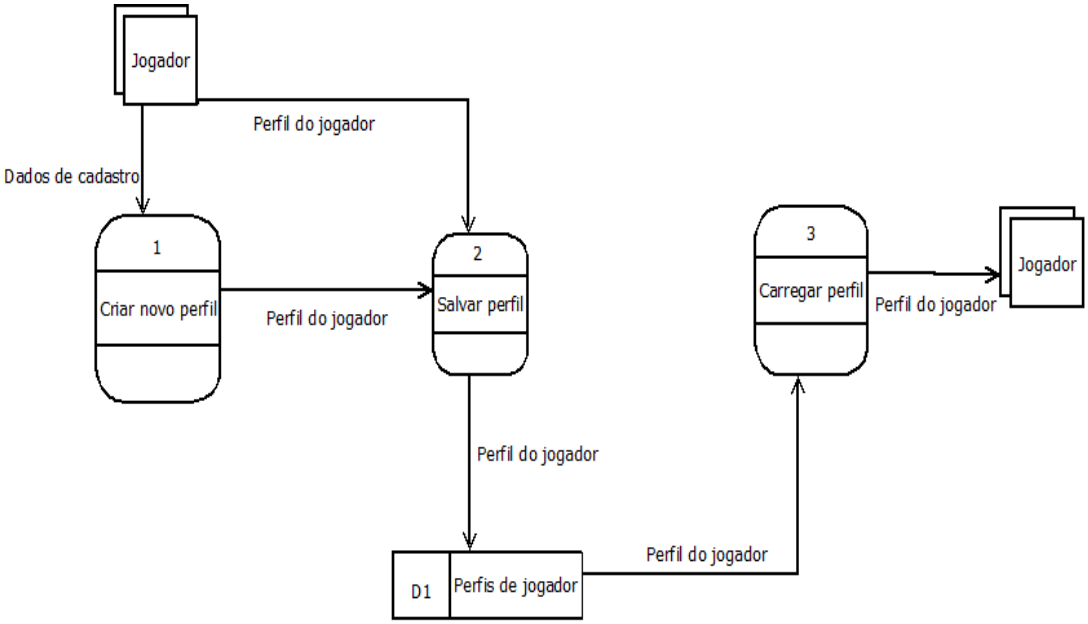

Imagem 5 – DFD2 Tela Inicial

- Menu principal:

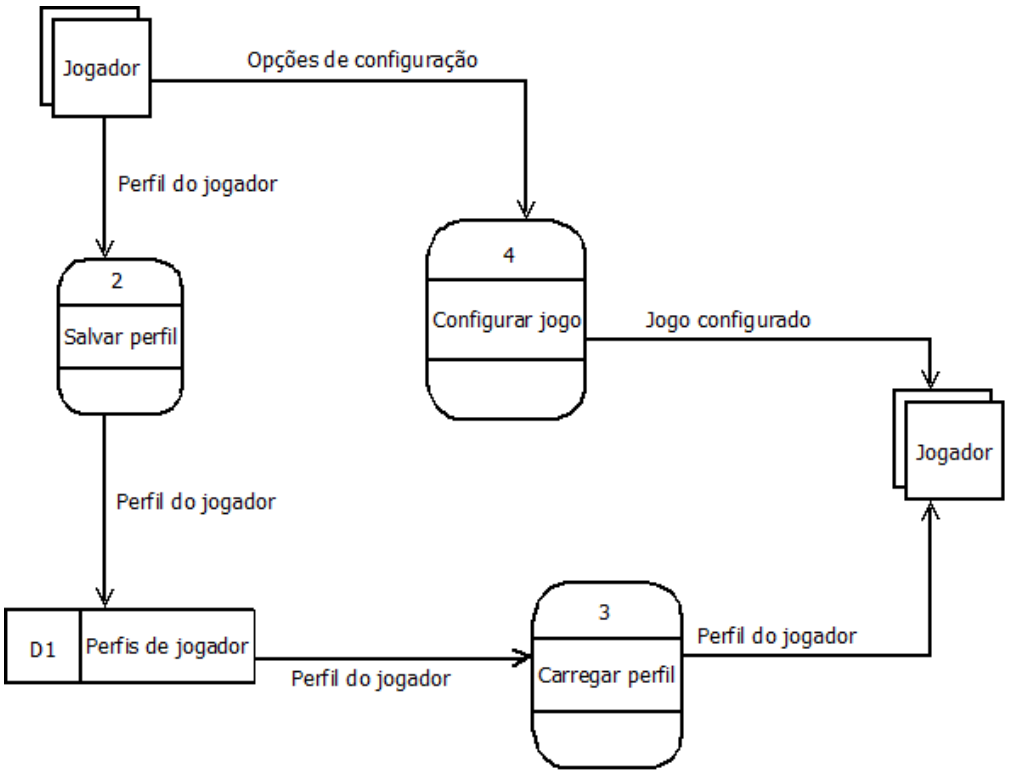

Imagem 6 – DFD2 Menu Principal

- Edição de baralho:

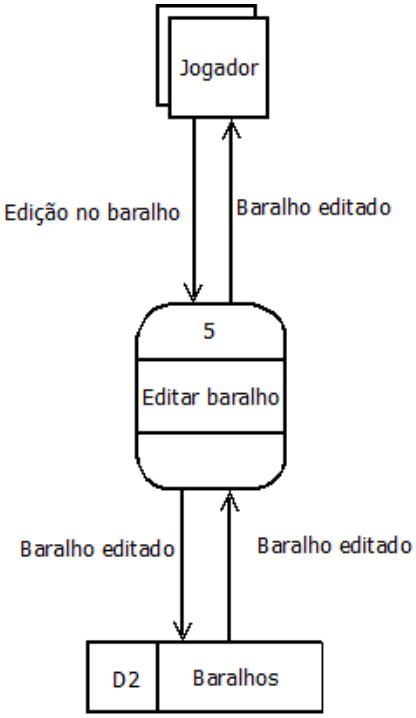

Imagem 7 – DFD2 Edição de Baralho

- Loja:

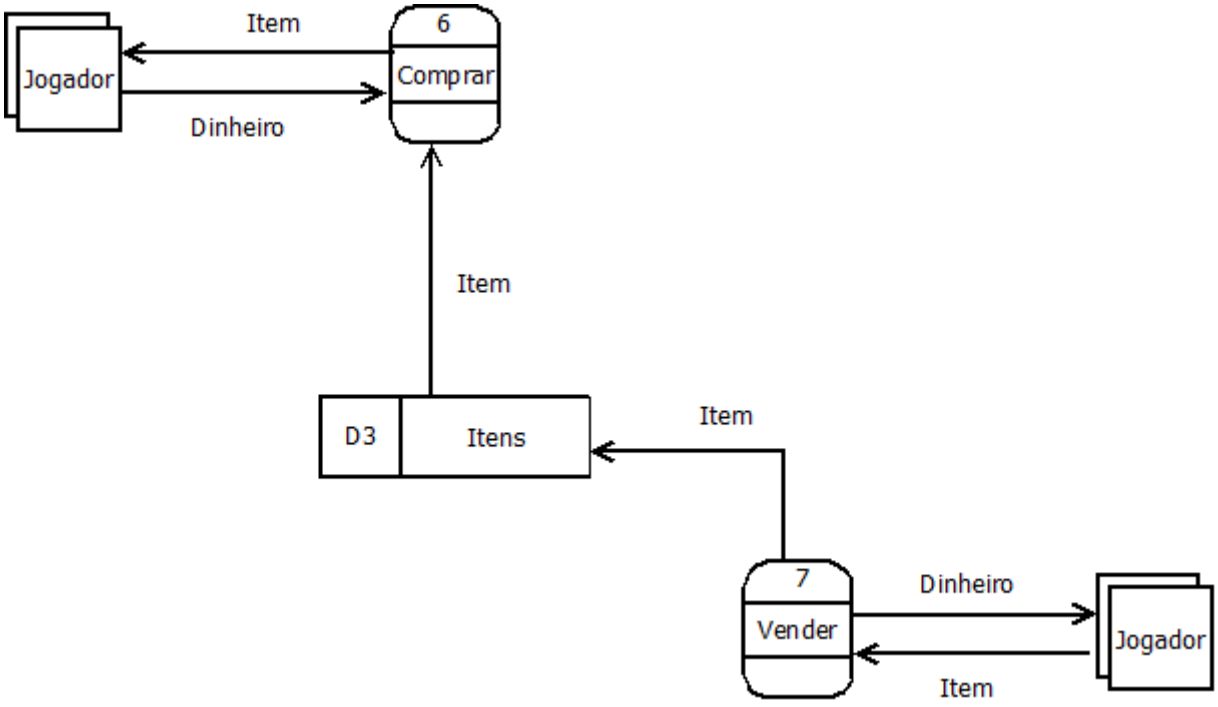

Imagem 8 – DFD2 Loja

- Modo campanha:

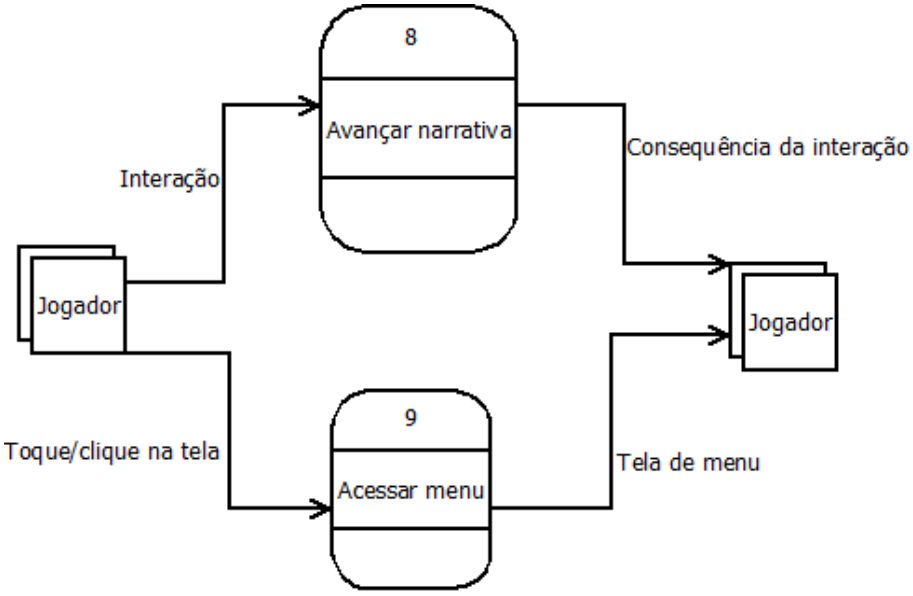

Imagem 9 – DFD2 Modo Campanha

- Modo livre:

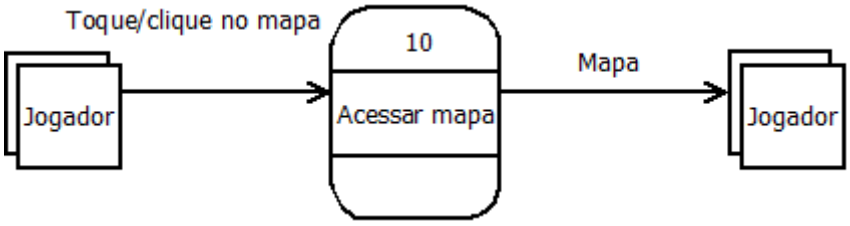

Imagem 10 – DFD2 Modo Livre

- RPG:

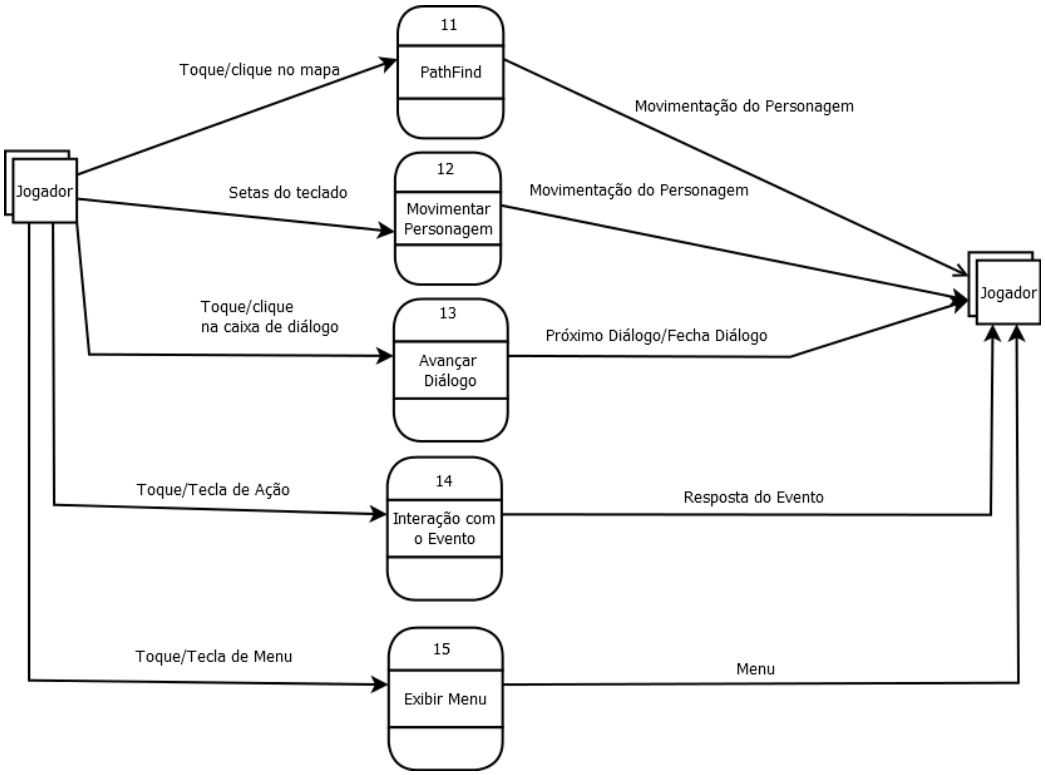

Imagem 11 – DFD2 Exploração

- Batalha:

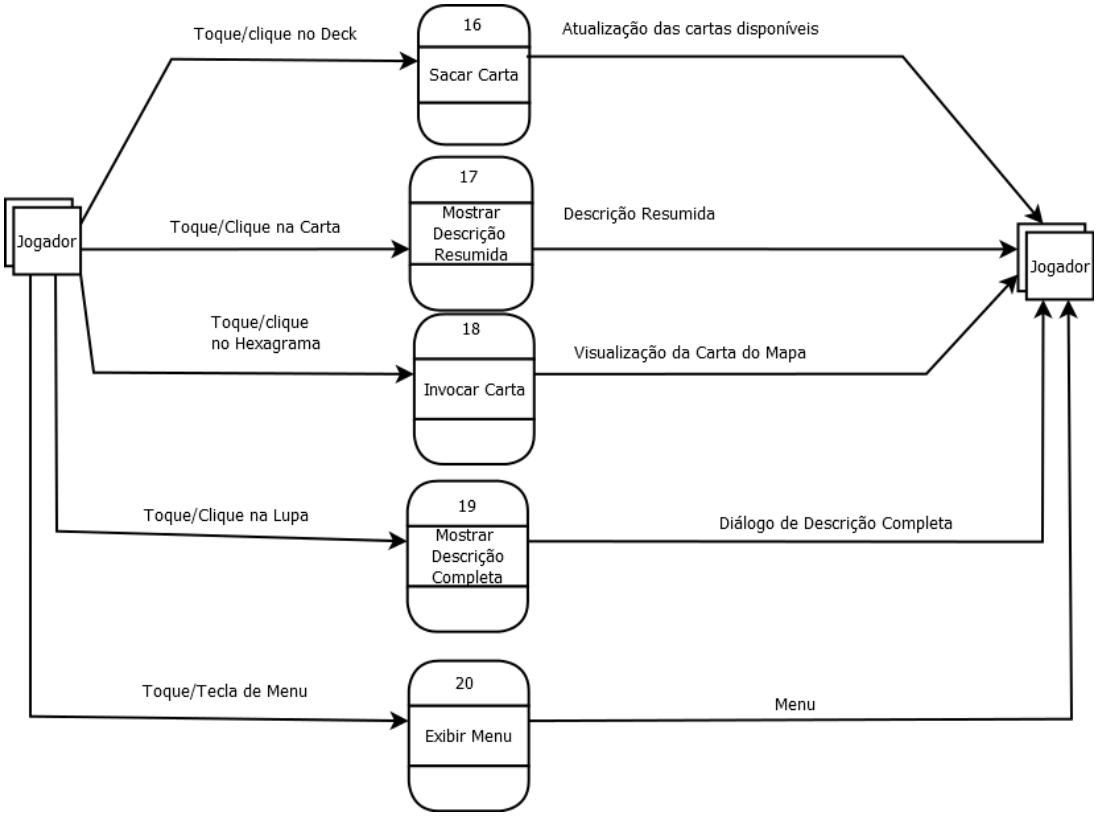

Imagem 12 – DFD2 Batalha

DFD 3

Tela inicial:

Semelhante ao DFD nível 2.

Menu principal:

Semelhante ao DFD nível 2

- Edição de baralho:

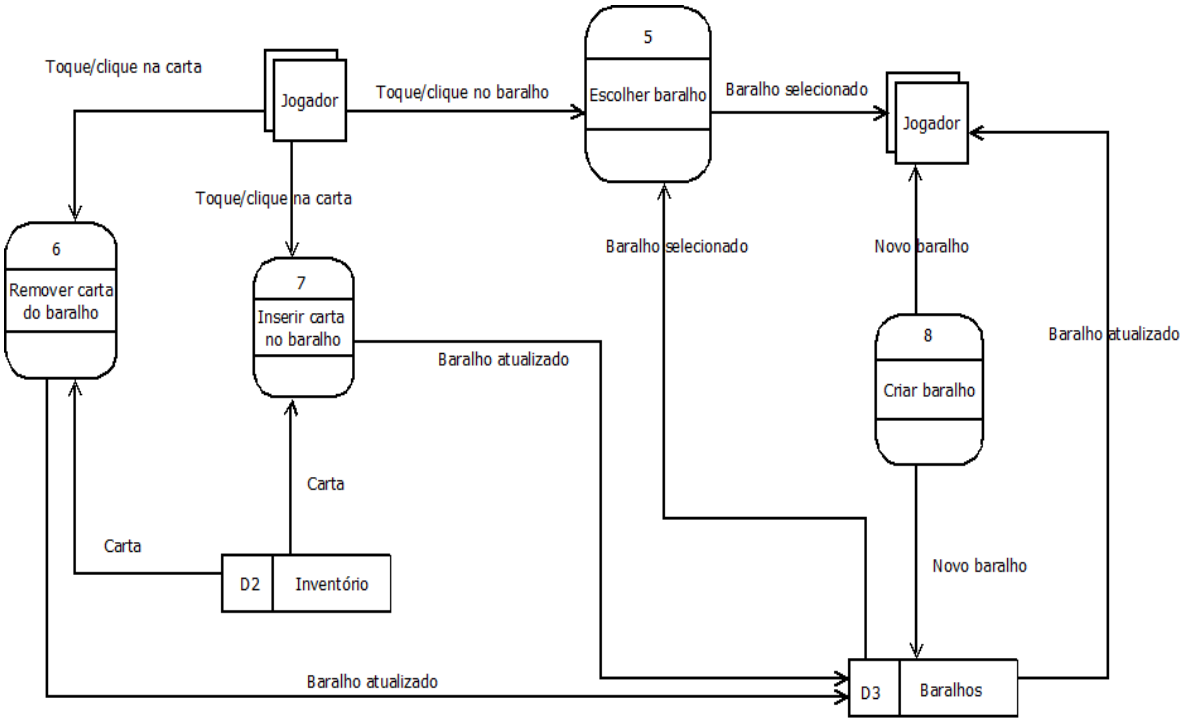

Imagem 13 – DFD3 Edição de Baralho

- Loja:

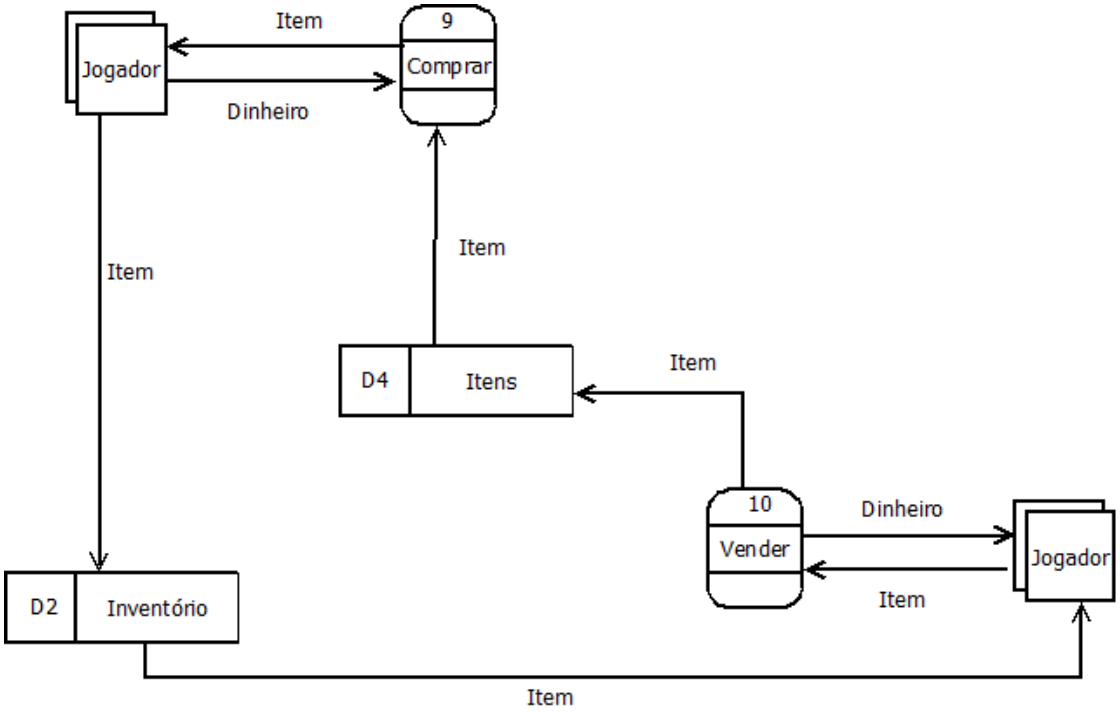

Imagem 14 – DFD3 Loja

- Modo campanha:

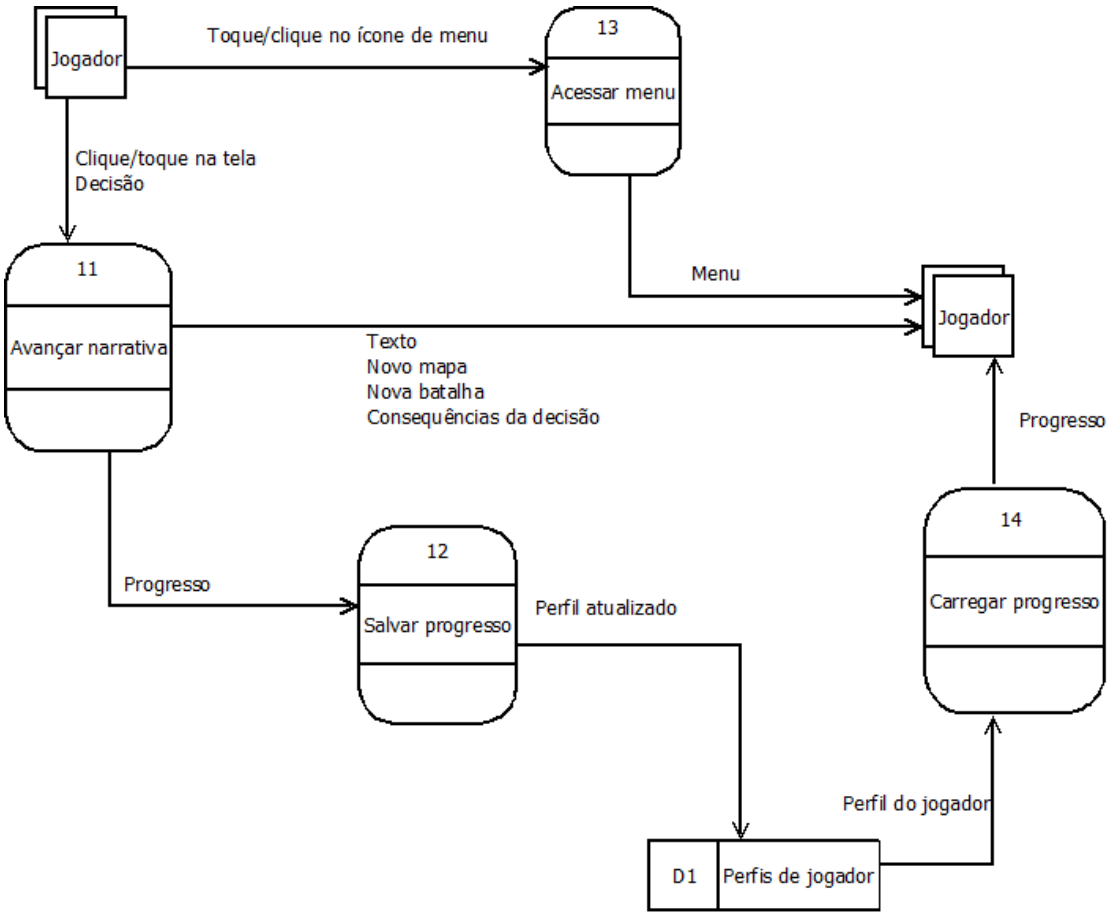

Imagem 15 – DFD3 Modo Campanha

- Modo livre:

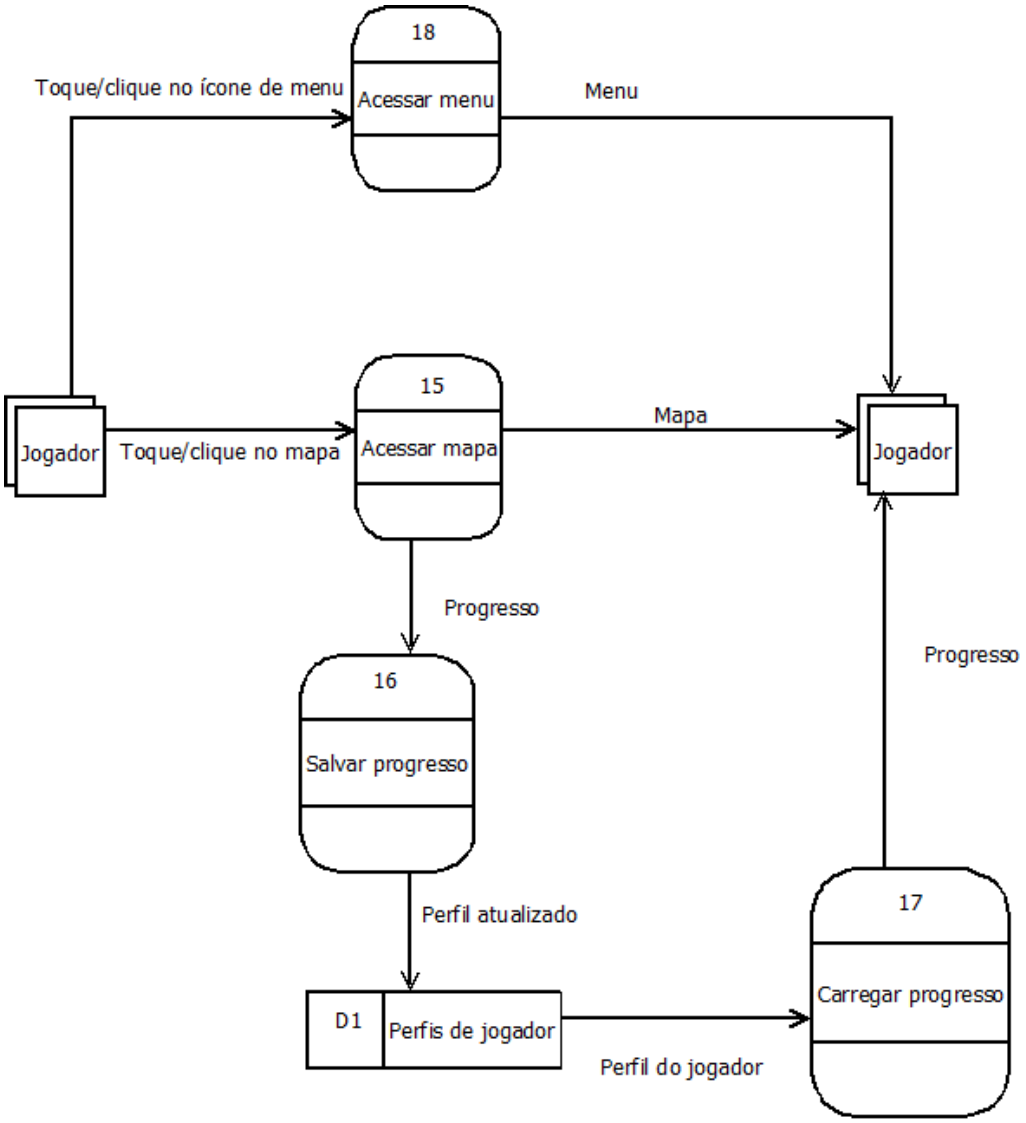

Imagem 16 – DFD3 Modo Livre

- RPG:

Ativar Evento

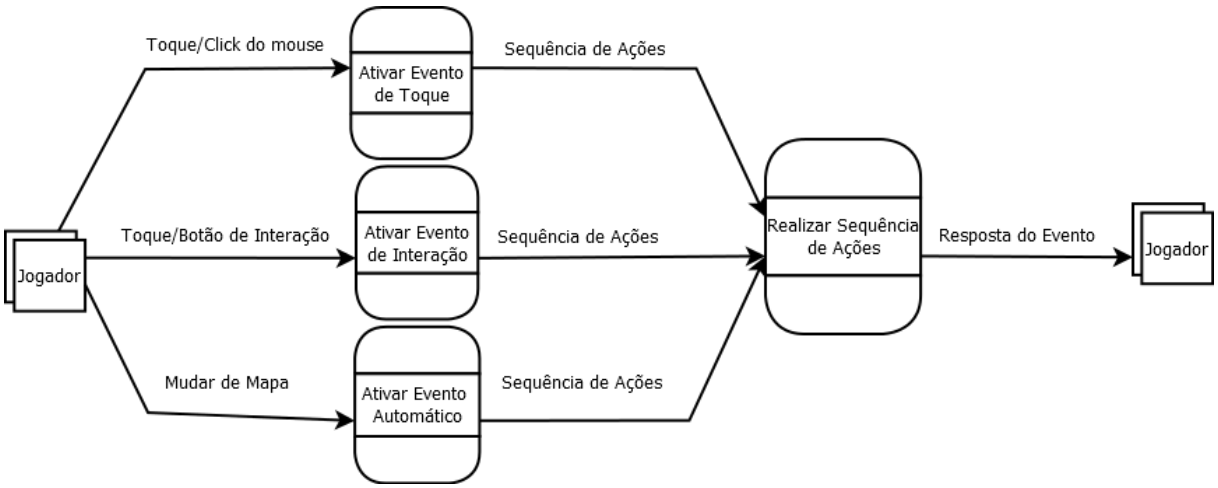

Imagem 17 – DFD3 RPG Ativar Evento

Avançar Diálogo

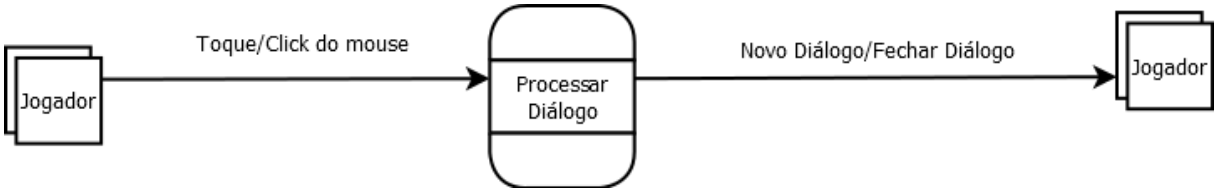

Imagem 18 – DFD3 RPG Ativar Diálogo

Movimentar Personagem

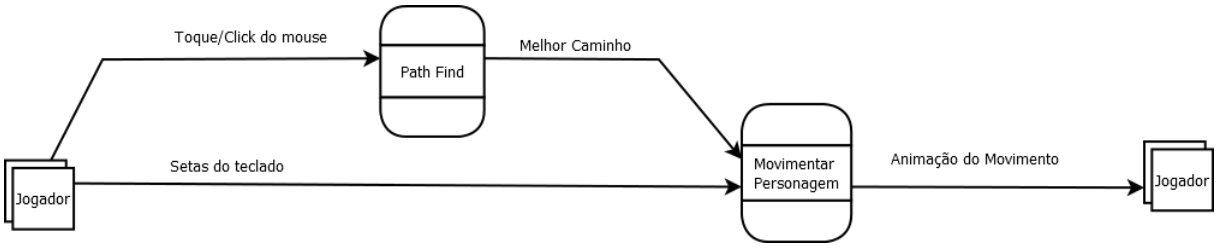

Imagem 19 – DFD3 RPG Movimentar Personagem

Acessar Menu do RPG

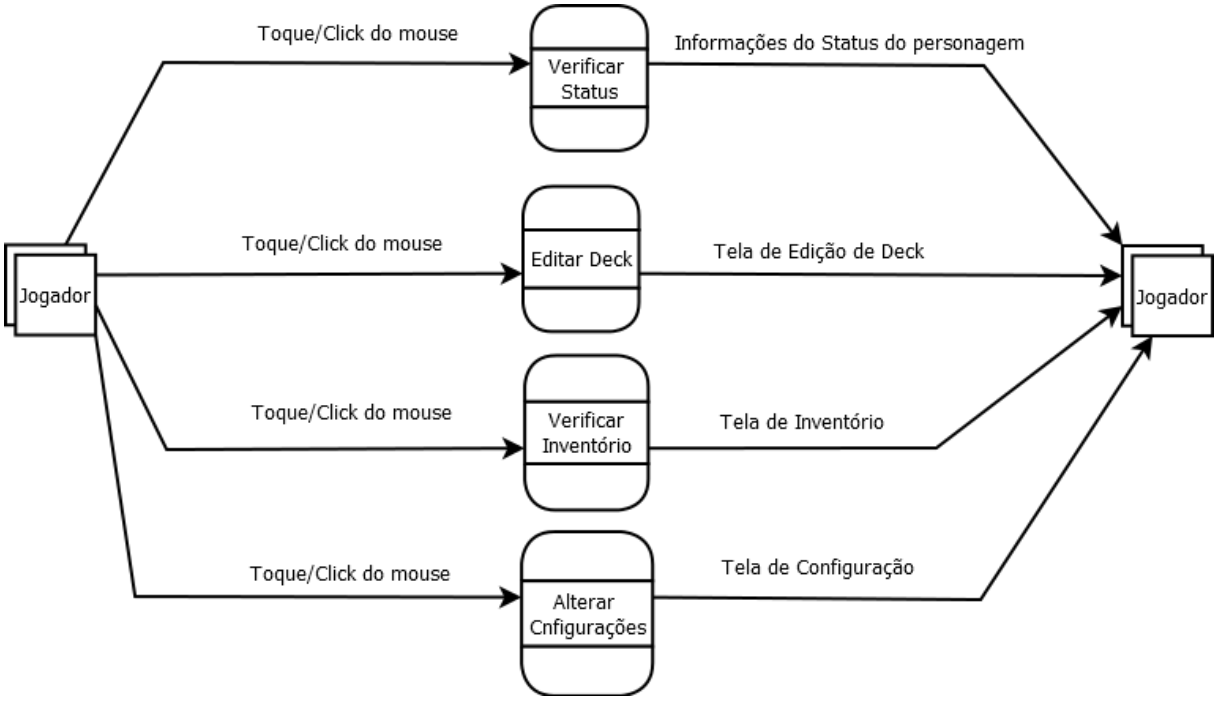

Imagem 20 – DFD3 RPG Acessar Menu

- Batalha:
Semelhante ao DFD Nível 2.

Casos de Uso do Shadow Struggles 2013

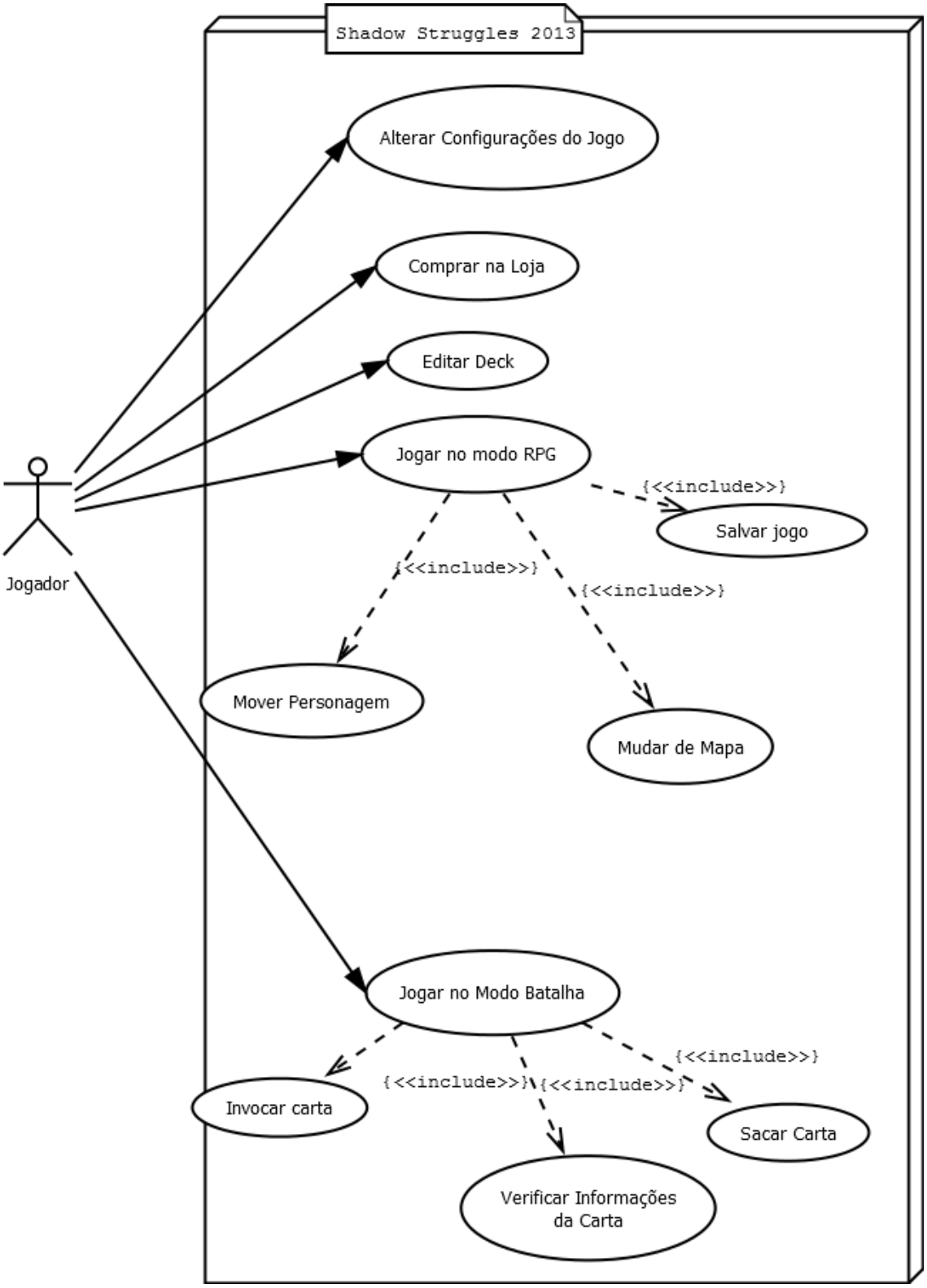

Imagem 21 – Casos de Uso

Navegação Geral

Descrição dos casos de uso – Navegação

Tela de carregamento

Esse caso tem como objetivo inicializar o jogo, carregando os dados do usuário (se existirem) e os elementos da navegação.

Ator primário: Usuário

Ator secundário: Shadow Struggles 2013

Fluxo de ações:

P0: O caso de uso é iniciado quando o ator primário inicia a aplicação.

P1: Ator secundário inicializa o áudio e mapas da batalha.

P2: O caso de uso é encerrado.

Tela inicial

Esse caso tem como objetivo direcionar o usuário em relação a continuação de um perfil de jogo já criado ou a criação de um novo.

Ator primário: Usuário

Ator secundário: Shadow Struggles 2013

Fluxo de ações:

P0: O caso de uso é iniciado após o término da tela de carregamento.

P1: O ator secundário exibe na tela o menu com as opções de “Continue” e “New Profile”.

P2: Após a seleção do ator primário o ator secundário verifica qual opção selecionada e realiza a devida ação

P2.A1: Caso o ator primário selecionar a opção de continuar, o ator secundário irá disponibilizar os perfis já criados para este selecionar.

P2.A2: Caso o ator primário selecione a opção de criar um novo perfil, o ator secundário cria um novo perfil no arquivo profiles.json.

P3: O caso de uso é encerrado.

Menu Principal

Esse caso de uso tem como objetivo exibir a interface do menu principal onde o usuário pode dirigir-se para as diferentes modalidades do jogo.

Ator primário: Usuário

Ator secundário: Shadow Struggles 2013

Fluxo de ações:

P0: O caso de uso é iniciado após o ator primário selecionar a ação a ser tomada na tela inicial.

P1: O ator secundário exibe na tela o menu com as opções de direção da navegação para o ator primário selecionar qual área ele deseja acessar.

P2: Após a seleção do ator primário o ator secundário verifica qual opção selecionada e realiza a devida ação

P2.A1: Caso o ator primário selecione a opção de modo campanha, o ator secundário irá iniciar a tela que leva o jogador ao jogo principal.

P2.A2: Caso o ator primário selecione a opção de modo Free Play, o ator secundário irá exibir a tela do modo Free Play.

P2.A3: Caso o ator primário selecione a opção do Shop, o ator secundário irá exibir a tela da loja do jogo.

P2.A4: Caso o ator primário selecione a opção de Edit Deck, o ator secundário irá exibir a tela da edição de decks.

P2.A2: Caso o ator primário selecione a opção de Setting, o ator secundário irá exibir a tela de configurações.

P3: O caso de uso é encerrado.

Free Play

Esse caso de uso tem como objetivo exibir a interface onde estarão dispostos as batalhas que o usuário poderá jogar de maneira independente do jogo principal.

Ator primário: Usuário

Ator secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: O caso de uso é iniciado após o ator primário selecionar a ação a ser tomada no menu principal.

P1: O ator secundário exibe na tela as batalhas disponíveis para serem jogadas separadas por capítulos nos quais estes estão inseridos.

P1.A1: Caso o ator primário selecione uma batalha, o ator secundário irá exibir a tela da batalha selecionada

P1.A2: Caso o ator primário selecione o botão de volta, o ator secundário irá exibir o menu principal.

P2: O caso de uso é encerrado.

Shop

Esse caso de uso tem como objetivo exibir a interface da loja do jogo, onde o usuário poderá realizar as compras e vendas de cartas a serem utilizadas na batalha.

Ator primário: Usuário

Ator secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: O caso de uso é iniciado após o ator primário selecionar a ação a ser tomada no menu principal.

P1: O ator secundário irá exibir as cartas a serem vendidas ou compradas pelo ator primário divididas em seções de cartas individuais, pacotes, decks e venda.

P1.A1: Caso o ator primário opte por comprar a carta, o ator secundário irá identificar qual carta foi selecionada, procurará o arquivo correspondente a esta, realizará um débito do valor correspondente de compra e irá adicionar seus dados ao perfil do ator primário.

P1.A2: Caso o ator primário opte por vender a carta, o ator secundário irá identificar qual carta foi selecionada, procurará o arquivo correspondente a esta, realizará um reembolso do valor correspondente de venda e removerá seus dados ao perfil do ator primário.

P1.A3: Caso o ator primário selecione o botão de volta, o ator secundário irá exibir o menu principal.

P2: O caso de uso é encerrado.

Edit Deck

Esse caso de uso tem como objetivo exibir a interface de edição dos decks onde o usuário poderá adicionar e remover cartas do deck e criar novos decks.

Ator primário: Usuário

Ator secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: O caso de uso é iniciado após o ator primário selecionar a ação a ser tomada no menu principal.

P1: O ator secundário irá exibir as cartas disponíveis para ator primário adicionar ao deck, as cartas já adicionadas no deck e a opção de criação de um novo deck.

P1.A1: Caso o ator primário retire uma carta do deck, o ator secundário removerá o dado da carta do deck alterado

P1.A2: Caso o ator primário adicione uma carta ao deck, o ator secundário adicionará o dado da carta do deck alterado

P1.A3: Caso o ator primário selecione a criação de um novo deck, o ator secundário irá inserir os dados do novo deck no arquivo json dos decks do ator primário.

P1.A4: Caso o ator primário selecione o botão de volta, o ator secundário irá exibir o menu principal.

P2: O caso de uso é encerrado.

Settings

Esse caso de uso tem como objetivo exibir a interface de configurações do jogo, como ajuste do volume e linguagem do jogo.

Ator primário: Usuário

Ator secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: O caso de uso é iniciado após o ator primário selecionar a ação a ser tomada no menu principal.

P1: O ator secundário irá exibir as opções de configurações com a configuração atual.

P1.A1: Caso o ator primário selecione a mudança de linguagem, o ator secundário irá exibir as linguagens disponíveis a serem utilizadas.

P2: O ator secundário irá efetuar as alterações feitas pelo primário

P2.A1: Caso o ator primário selecione o botão de salvar as alterações, o ator secundário irá identificar se o volume está habilitado, o nível do mesmo e a linguagem selecionada. Após identificar as configurações o ator secundário irá efetuar as alterações no som e utilizará os dados de acordo com a linguagem selecionada

P2.A2: Caso o ator primário selecione o botão de descartar as alterações, o ator secundário irá retornar ao menu principal sem realizar as alterações.

P3: O caso de uso é encerrado.

RPG

Jogar no modo RPG
Este caso tem como objetivo permitir que o usuário jogue no modo RPG do aplicativo.

Ator Primário: Usuário

Ator Secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: O caso de uso é iniciado quando o ator primário inicia o aplicativo

P1: Após o carregamento do jogo:

P1:A1: O Ator primário inicia um novo jogo clicando no botão

P1:A2: O Ator primário seleciona um perfil já criado, para continuar um jogo já iniciado, clicando no botão continuar, e escolhendo um perfil

P2: O ator primário entra então no modo campanha, clicando no botão campanha, para acessar o modo RPG de jogo

P3: O ator primário então explora o mapa, podendo acionar eventos como salvar o jogo ou entrar em batalhas.

P4: O caso de uso é encerrado

Movimentação do Personagem
Neste caso de uso é descrita a movimentação do personagem pelo jogador.

Ator Primário: Jogador

Ator Secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: O modo RPG do jogo é iniciado pelo ator primário

P1: O Ator secundário exibe a tela do modo RPG

P2: O Ator primário aciona a movimentação do personagem

P2: A1: Pelas setas do teclado

P2: A1: P0: O ator secundário verifica se o destino do movimento está desobstruído

P2: A1: P1: A1: Se o caminho está livre, o ator secundário movimenta o personagem

P2: A1: P1: A2: Se o caminho está obstruído, o ator secundário ignora o comando do ator primário.

P2: A2: Por meio do clique com o mouse

P2: A2: P0: O ator secundário verifica se é possível chegar no destino

P2: A2: P0: A1: Não é possível alcançar o destino

P2: A2: P0: A1: P0: O ator secundário calcula a rota ao local mais próximo possível e move o personagem para lá

P2: A2: P0: A2: O destino é alcançável

P2: A2: P0: A2: P0: O ator secundário calcula a rota até o local

P2: A2: P0: A2: P1: O ator secundário move o personagem até o local por meio da rota calculada

P3: O caso de uso é encerrado

Save game
Este caso tem como objetivo permitir que um usuário salve o jogo, podendo fechar o jogo e retornar a jogar sem perder o progresso obtido.

Ator primário: Usuário
Ator secundário: Shadow Struggles 2013

Fluxo de Ações:
P0: O caso de uso é iniciado quando o ator primário que estiver jogando no modo RPG ativar um save point clicando nele ou passando sobre ele.
P1: O ator secundário gravará o progresso do ator primário no arquivo de dados do jogador.
P2: O caso de uso é encerrado.

Teleport event
Este caso tem como objetivo transportar um jogador no modo rpg de um lugar para outro.

Ator primário: Usuário
Ator secundário: Shadow Struggles 2013

Fluxo de Ações:
P0: O caso de uso é iniciado quando o ator primário ativa um evento de teletransporte pisando em um bloco ativador ou por outro meio
P1: O ator secundário então deve transportar o usuário para o local previamente determinado para o evento ativado
P2: O caso de uso é encerrado

Batalha

Início de batalha

Esse caso de uso tem como objetivo inicializar o ambiente de batalha e preparar os elementos que deverão ser controlados pelo jogador

Ator Primário: Usuário

Ator Secundário: Shadow Struggles 2013

Fluxo de ações:

P0:O caso de uso é iniciado quando o ator primário inicia uma batalha ou quando é ativado um evento de batalha.

P1: o ator secundário exibe na tela a interface do usuário e o ambiente da batalha, inicializando a vida, energia e o tempo da partida com os valores padrões

P2:O ator secundário lê o arquivo do deck padrão do usuário e embaralha as cartas para dispô-las na interface do usuário.

P3:Após todas as renderizações, o ator secundário começa o ciclo de verificação de vida e aumento de tempo e energia.

P3.A1: Caso a vida da base do ator primário seja igual a zero, é exibida a tela com opções de Recomeçar a Partida ou sair do jogo.

P3.A2: Caso a vida da base inimiga seja igual a zero, é exibida a tela de vitória com os prêmios obtidos.

P3.A3: Caso a vida da base do ator primário e da base inimiga seja diferente de zero, as atualizações da batalha são realizadas continuamente.

P4.: O ator secundário fica disponível para alguma ação do ator primário (Menu, Sacar Carta do Deck, Invocar Carta, Verificar Informações da Carta).

P5: O loop da batalha é feito através do método render() da classe BattleScreen. O método render() é invocado automaticamente pelo programa cerca de 30 vezes por secundo. Assim são feitas as atualizações lógicas e visuais.

P6: O ator secundário aumenta a energia de ambos os jogadores em uma unidade por segundo.

P7: Para cada carta em campo, é realizada sua respectiva ação, podendo ser de movimento, ataque ou alteração de atributos.

P8: A vida atual é calculada de acordo com os ataques direcionados à base.

P9: A interação do Ator Primário no campo é detectada e a câmera é atualizada para a posição escolhida.

P10: Os recursos visuais são atualizados de acordo com seus respectivos representantes lógicos.

Sacar Carta do Deck (Draw Card)

Esse caso de uso tem como objetivo permitir ao usuário a obtenção de cartas do deck ao longo da batalha.

Ator Primário: Usuário

Ator Secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: Uma vez que o ator primário escolhe a opção de Sacar uma Carta do Deck, o ator secundário verifica se o tempo de espera já terminou (existe um intervalo de 10 segundos entre cada saque).

P0.A1: Caso ainda existe tempo de espera restante, nada é feito. O usuário deve aguardar o tempo de espera terminar.

P0.A2: Se o tempo de espera tiver terminado, é verificado se a pilha de cartas representada pelo Deck contém cartas (?tamanho=0)

P0.A2.A1:Caso a pilha do Deck contenha cartas (tamanho>0) , o ator secundário retira a carta do topo da pilha e adiciona a carta para a Lista de Cartas da “mão” na interface do ator primário.

P0.A2.A2:Caso a pilha do Deck não contenha mais cartas (tamanho==0) é definido a derrota do ator primário, e é exibida a tela com opções de Recomeçar a Partida ou sair do jogo.

Invocar Carta (Summon Card)

Esse caso de uso tem como objetivo permitir ao usuário a invocação de uma das cartas disponíveis na “mão”, para o campo de batalha.

Ator Primário: Usuário

Ator Secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: Uma vez que o ator primário clica em uma das cartas da “mão” (lista de cartas disponíveis para invocação), são exibidas imediatamente as informações básicas da carta.

P1: É exibido um conjunto de hexagramas para indicar os locais do campo onde a carta pode ser invocada.

P2: Quando o Ator Primário escolhe o “Tile”(local do campo) onde a carta será invocada, o Ator Secundário verifica se há energia necessária (cada carta tem um gasto energético diferente).

P2.A1: Caso a energia atual do Ator Primário seja inferior à energia necessária para a invocação da carta, nada é feito. O Ator Primário deve aguardar o aumento da energia ou escolher outra carta.

P2.A2: Caso a energia atual do Ator Primário seja igual ou maior à energia necessária, o Ator Secundário verifica se os requisitos para a invocação da carta foram atingidos (algumas cartas dependem de outras cartas para serem invocadas).

P2.A2.A1: Caso um ou mais requisitos não forem cumpridos, nada é feito, e o jogador deve escolher outra carta para ser invocada.

P2.A2.A2: Caso todos os requisitos de invocação forem cumpridos, a carta é retirada da “mão” do usuário, e é instanciado um objeto do tipo Card (Fighter, Trap ou Effect) que representa tanto lógica quanto visualmente a carta escolhida.

P2.A2.A2.P1: O objeto instanciado é colocado no campo no Tile escolhido, e suas ações são realizadas no loop geral da batalha.

Verificar Informações da Carta

Esse caso de uso tem como objetivo permitir ao usuário a verificação de todas as informações de determinada carta durante a batalha, para saber seus efeitos, requisitos, e poder.

Ator Primário: Usuário

Ator Secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: Ao clicar em uma das cartas da “mão”, o Ator Primário pode clicar na lupa para verificar as informações sobre a carta selecionada.

P1: Ao clicar no ícone de lupa, é exibida uma janela sobre a tela da batalha, com a imagem ampliada da carta, o nome, a descrição, e todos os atributos da carta.

P2: Ao clicar novamente no ícone da lupa, a janela com as informações desaparece e o Ator Primário volta á batalha.

Menu de Pausa (Pause Menu)

Esse caso de uso tem como objetivo permitir ao usuário pausar o jogo e exibir o menu de pausa com diversas opções de navegação.

Ator Primário: Usuário

Ator Secundário: Shadow Struggles 2013

Fluxo de Ações:

P0: Ao clicar no ícone de Pausar o jogo no canto superior esquerdo da tela de batalha, é exibido um menu com as seguintes opções: Voltar ao Jogo, Verificar Informações das Cartas, Alterar Configurações e Sair da batalha.

Estrutura dos pacotes Java

Pacote br.edu.ifsp.pds.shadowstruggles

Contém as classes básicas para a inicialização do programa. A classe principal ShadowStruggles e a classe que inicia a aplicação da plataforma Desktop se ecnontram nesse pacote. Também possui a classe Controller geral da aplicação.

Pacote br.edu.ifsp.pds.shadowstruggles.data

Representa a camada de acesso de dados. Contém as classes de entrada e saída de dados, gerenciamento de recursos, e constantes globais da aplicação. A principal classe do pacote é a classe DataManager, que gerencia a leitura e escrita dos arquivos JSON.

Pacote br.edu.ifsp.pds.shadowstruggles.data.dao

Contém as classes DAO utilizadas para obtenção dos objetos gerados pela classe DataManager do pacote data.

Pacote br.edu.ifsp.pds.shadowstruggles.games

Pacote que armazena os diferentes tipos de batalha.

Pacote br.edu.ifsp.pds.shadowstruggles.model.*

Representa a camada lógica do jogo, responsável pelas principais funcionalidades. Contém todas as classes dos elementos lógicos do jogo, como Cartas, Itens, Cenas, e a própria mecânica de funcionamento do jogo.

Pacote br.edu.ifsp.pds.shadowstruggles.object2d

Contêm as representações visuais dos elementos da batalha, como Cartas, Botões, Imagens em geral.

Pacote br.edu.ifsp.pds.shadowstruggles.object2d.rpg

Contêm as representações visuais dos elementos do RPG, como Itens, NPC’s, o Personagem, entre outros.

Pacote br.edu.ifsp.pds.shadowstruggles.rpg

Contêm algumas classes auxiliares exclusivas na mecânica do modo RPG.

Pacote br.edu.ifsp.pds.shadowstruggles.screens

Contêm todas as telas de interface gráfica de navegação, como a Tela de Início, a Tela de Salvar o jogo, a Tela de Edição de Deck, O Shopping, entre outras.

Pacote br.edu.ifsp.pds.shadowstruggles.screens.rpg

Contém as telas que são utilizadas exclusivamente no RPG

Pacote br.edu.ifsp.pds.shadowstruggles.screens.utils

Contém classes auxiliares para a interface gráfica

Pacote br.edu.ifsp.pds.shadowstruggles.scripts

Contém classes que representam as ações das cartas na batalha

Descrição das Principais Classes

Classe br.edu.ifsp.pds.shadowstruggles.DesktopStarter.java

É o ponto de entrada para a execução do jogo na plataforma Desktop.

Extende a classe **LwjglApplication** (nativa do LibGdx), da API LWJGL(Lightweight Java Game Library), contida no LibGdx.

Basicamente cria uma instância do jogo executável pelo Desktop (através do ApplicationListener) e inicializa o sistema de Logging.

Nessa classe também é definido o tamanho da janela ao iniciar o jogo no Desktop e é definido o tipo de execução do jogo (DEBUG, TESTS, RELEASE)

Classe br.edu.ifsp.pds.shadowstruggles.Shadow Struggles.java

Representa a estrutura da aplicação, responsável pela troca de telas e outras operações gerais relativas ao jogo.

Extende a classe **Game**(nativa do LibGdx), especializada no gerenciamento de *Threads* de Games no LibGdx

É inicializada pelo método ***create()***.

- O **método create()** é responsável pela inicialização da classe gerenciadora de sons SoundManager, pela classe gerenciadora de recursos AssetManager e pela classe controladora geral **Controller**.
    - Em seguida, é verificado se o jogo está em modalidade de TESTS. Nesse caso, são instanciadas as classes dos testes unitários.

    - Caso esteja em modo DEBUG ou RELEASE, é instanciada a tela inicial do jogo **SaveLoadScreen** e a tela LoadingScreen (serve apenas para mostrar o progresso dos recursos sendo carregados na memória RAM).
    - A classe Game se encarrega de chamar o método **create()** automaticamente quando é instanciada.

- O método **initLoading()** inicializa a classe Loader (responsável por carregar os recursos) e as Arrays de recursos que serão utilizados ao longo do jogo.
- O método **setScreenWithTransition(Screen screen)** é responsável pelas trocas de telas com efeito de transição (FadeOut e FadeIn)

Classe br.edu.ifsp.pds.shadowstruggles.Controller.java

Estabelece a ligação entre os elementos visuais e lógicos. A camada visual funciona como um *Listener* para a Controller, chamando um de seus métodos quando acontece a ativação de um evento (geralmente uma ação do usuário).

Os métodos da Controller podem ser divididos em 3 categorias:

- Métodos de Ativação de Eventos:

- **mapClicked(float x, float y):** ativado quando o usuário clica no mapa. Sua principal função é detectar se determinado clique no Mapa representa a invocação de uma carta ou a ativação de um efeito em uma das cartas do campo. Também verifica se o usuário possui energia necessária para fazer a invocação.
- **hexagramClicked(int Lane, int tile):** ativado quando o jogador vai invocar uma carta em determinado hexagrama do campo. Desconta a energia de

invocação, chama o método de invocação **playCard**, e retira os hexagramas do campo.

- **handCardClicked(Card handCard, boolean isSelected):** ativado quando o jogador clica em uma das cartas da “mão”, disponíveis para invocação. Faz com que a carta clicada seja selecionada (começa a piscar), mostra as informações resumidas da carta, e exibe os hexagramas no campo (representam os locais onde a carta pode ser invocada)
- **deckClicked():** ativado quando o jogador clica no Deck. Verifica se o tempo de espera para saque já terminou. Se sim, retira uma carta do topo do Deck e a coloca na “mão” do jogador.
- **menuButtonClicked(ShadowStruggles game):** ativado quando o jogador clica no botão de pause. Exibe a tela de pausa com o menu de pausa.
- **playerLifeChanged(int amount):** verifica se a vida do jogador pode ser alterada pelo novo valor. Se sim, soma/subtai o valor da vida o usuário e atualiza o indicador de vida da tela.
- **tileChanged(Fighter card):** ativado quando um lutador em campo está no limite máximo de um tile. De acordo com a direção do movimento, retira o lutador do Tile atual e o coloca no Tile posterior.

- Métodos de Ação

- **playCard(Card handCard, int Lane, int tile):** torna visível um marcador na posição da carta invocada, verifica o tipo da carta (Fighter, Effect, Trap), e cria a representação visual da carta.Por fim, chama o método **rearrangeCards.**
- **addCardToMap(Card handCard, Image cardImage, int lane, int tile):** adiciona a imagem da carta na lista de imagens, remove a imagem da carta da “mão” do usuário, coloca o *Sprite* da carta na posição escolhida do campo, e ativa o efeito da carta caso seja do tipo Effect.
- **rearrangeCards()** : rearranja as cartas na “mão” do jogador, após uma invocação.

- Métodos de Cálculo:

- **verifyValueChange(int newValue, int maxValue):** verifica se o novo valor pode ser atribuído a determinada variável (se não supera o valor máximo estabelecido).
- **yToLane(float y):** retorna a Lane correspondente a uma determinada posição y da tela. A Viewer trabalha apenas com posições x,y. E a Model trabalha apenas com Tiles e Lanes. Para que ocorra a interação entre as camadas, é necessária tal conversão.
- **xToTile(float x):** retorna o Tile correspondente a uma determinada posição x da tela.

Classe br.edu.ifsp.pds.shadowstruggles.data.DataManager.java

É a principal classe para trabalhar com operações de entrada e saída, incluindo i18n. Utiliza o formato JSON para armazenamento de dados.
Principais métodos:

- **changeLanguage(String language):** altera o idioma padrão do jogo, utilizado em menus, textos, nomes.
- **retrieve():** percorre todos os arquivos do diretório padrão eadiciona todos os objetos em um Mapa de Objetos
- **writeProfile(Profile profile):** método utilizado para a criação de um novo perfil do usuário quando é iniciado um novo *save.*
- **save():** armazena em arquivo JSON todas as alterações realizadas pelo jogador durante o jogos

Classe br.edu.ifsp.pds.shadowstruggles.screens.BattleScreen.java
Representa a camada visual da batalha. Principal elemento de interação entre o jogador e o jogo. Além de mostrar todos os elementos gráficos da batalha, também cumpre a missão de *Listener* das ações do jogador.
Principais métodos:

- **initComponents():** método utilizado para inicializar e preparar todos os componentes gráficos persistentes que serão renderizados na interface de batalha.
- **render():** compõe a *Thread* principal do jogo. É chamado automaticamente múltiplas vezes por segundo para renderizar continuamente todos os elementos gráficos. É responsável também por manter a atividade constante do inimigo.
- **update():** ativado pelo método render(), é utilizado para atualizar as informações do jogo, como tempo decorrido, energia atual, controle de câmera, e as ações das cartas.