

**Centro Universitário Senac**

Luis Henrique Menezes Mauruto

Nilson Calazans Dias Filho

Willians Schallenberger Schneider

IR Taktiks – Jogo tático para mesas multitoque

**São Paulo**

**2008**

**Luis Henrique Menezes Mauruto**

**Nilson Calazans Dias Filho**

**Willians Schallenberger Schneider**

**IR Taktiks – Jogo tático para mesas multitoque**

**Projeto de Conclusão de Curso**

**Apresentado ao Centro Universitário**

**Senac - Campus Senac.**

**Orientador: Fabio R. Miranda**

**São Paulo**

**2008**

“ESPAÇO PARA”

Folhe de Aprovação (obrigatória);

Dedicatória (opcional);

Agradecimentos (opcional);

Epígrafe (opcional);

SEGUNDO:<http://www.monografia.net/abnt/index.htm>

**RESUMO**

Este trabalho propõe o desenvolvimento do jogo “IRTaktiks – Jogo tático para mesas multitoque”, como parte dos experimentos para a conclusão do curso, projeto este desenvolvido no “Laboratório de TCC” do Centro Universitário Senac. O jogo consiste no uso de uma ecrã interativa, multitoque, que utiliza reflexão total interna frustrada da luz para que com dispositivos de baixo custo se possa mapear a interação, através de toque, com bibliotecas de visão computacional de livre acesso, transformadas em eventos do IRTaktiks. Jogo de estratégia com elementos de RPG desenvolvido sobre arcabouço XNA, permitirá um controle natural dos personagens que compõe os exércitos distintos e suas ações, com toques e gestos de comando como estudo dessa forma de interação com superfícies.

Palavras-chave: interação,multi-toque, visão computacional, jogo, reflexão total interna frustrada da luz

**ABSTRACT**

Este trabalho propõe o desenvolvimento do jogo “IRTaktiks – Jogo tático para mesas multitoque”, como parte dos experimentos para a conclusão do curso, projeto este desenvolvido no “Laboratório de TCC” do Centro Universitário Senac. O jogo consiste no uso de uma ecrã/tela interativa, multitoque, que utiliza reflexão total interna frustrada da luz para que com dispositivos de baixo custo possa mapear a interação, seja toque ou fiduciais, com bibliotecas de visão computacional de livre acesso.

Palavras-chave: multi-toque, visão computacional, jogo, reflexão total interna frustrada da luz

Conteúdo

[Introdução 9](#_Toc199691155)

[Interação Multitoque 10](#_Toc199691156)

[História 11](#_Toc199691157)

[Microsoft Surface 11](#_Toc199691158)

[Especificação 12](#_Toc199691159)

[Interação 12](#_Toc199691160)

[ReacTable 12](#_Toc199691161)

[Especificação 13](#_Toc199691162)

[Interação 13](#_Toc199691163)

[iPhone 13](#_Toc199691164)

[Especificação 14](#_Toc199691165)

[Interação 14](#_Toc199691166)

[Jogos e a interatividade 14](#_Toc199691167)

[Controles Wireless com dispositivo de posicionamento 15](#_Toc199691168)

[Realidade Virtual 16](#_Toc199691169)

[Realidade Aumentada 16](#_Toc199691170)

[Jogos de Estratégia 16](#_Toc199691171)

[Go 17](#_Toc199691172)

[Age of Empires 17](#_Toc199691173)

[Jogo de RPG 17](#_Toc199691174)

[Dungeons & Dragons 18](#_Toc199691175)

[Jogo RPG Eletrônico 19](#_Toc199691176)

[RPG Tático x RPG Ação 22](#_Toc199691177)

[Final Fantasy Tactis 22](#_Toc199691178)

[Objetivos 22](#_Toc199691179)

[Proposta 23](#_Toc199691180)

[Tecnologias 24](#_Toc199691181)

[Framework XNA 24](#_Toc199691182)

[Plataform 25](#_Toc199691183)

[Core Framework 25](#_Toc199691184)

[Extended Framework 25](#_Toc199691185)

[Games 25](#_Toc199691186)

[ReacTIVision 26](#_Toc199691187)

[TouchLib 26](#_Toc199691188)

[OSCPack 27](#_Toc199691189)

[Projeto 28](#_Toc199691190)

[Arquitetura 29](#_Toc199691191)

[Mesa 30](#_Toc199691192)

[Jogo 30](#_Toc199691193)

[Projeto 31](#_Toc199691194)

[Protótipo 31](#_Toc199691195)

[Versão Final 33](#_Toc199691196)

[Módulo Listener 35](#_Toc199691197)

[Módulo Input 37](#_Toc199691198)

[Módulo Resource 38](#_Toc199691199)

[Módulo Drawable 41](#_Toc199691200)

[Submódulo Camera 41](#_Toc199691201)

[Submódulo Sprite 42](#_Toc199691202)

[Submódulo Map 43](#_Toc199691203)

[Submódulo Area 46](#_Toc199691204)

[Submódulo Animation 48](#_Toc199691205)

[Submódulo Damage 50](#_Toc199691206)

[Módulo Game 51](#_Toc199691207)

[Módulo Logic 52](#_Toc199691208)

[Módulo Screen 54](#_Toc199691209)

[Módulo Debug 55](#_Toc199691210)

[Módulo Menu 56](#_Toc199691211)

[Módulo Interaction 59](#_Toc199691212)

[Submódulo Mover 59](#_Toc199691213)

[Submódulo Aim 61](#_Toc199691214)

[Módulo Action 63](#_Toc199691215)

[Referencias 64](#_Toc199691216)

[Bibliográfica 64](#_Toc199691217)

[Sites 64](#_Toc199691218)

[Imagens 64](#_Toc199691219)

[Gráficos 64](#_Toc199691220)

# Introdução

Os meios de interação entre usuários e dispositivos são objeto de constantes avanços e busca por funcionalidade e diversificação. A cada dia surgem novas idéias e modelos de interação assim tornando o mundo virtual cada dia mais real. Idéias e propostas de formas de interação que surgiram através de filmes ou seriados, como o *Holodeck* presente no seriado *Star-Trek* ou então o monitor de operações do filme *Minority Report,* hoje encontram-se em vias de estar presentes no cotidiano das pessoas através de interfaces em celulares, restaurantes, hotéis, cassinos, exibições artísticas, na indústria, em pesquisas, em museus e em entretenimento.

Os jogos têm um papel essencial nessa evolução, tanto hardware voltado para o publico civil, pois alavancam as vendas e desenvolvimento de novos dispositivos como processadores, placas de vídeo, memórias, e claro, dispositivos de interação, quanto financeiro, arrecadando bilhões de dólares todos os anos (Não achei do mundo inteiro, mas segundo a NDP, 18bilhões nos estados unidos só em venda, e cerca de 6 bilhões com jogos online (assinaturas e venda de intens – 1 bilhão de world of warcraft).

## Interação Multitoque

É uma técnica de interação homem-computador com utilização de dispositivos periféricos. O multitoque consiste no reconhecimento de múltiplos toques simultâneos em uma superfície, (pode ser uma tela, ou uma mesa com projeção, por exemplo) e sua interpretação por software. Esse reconhecimento pode ser de posição, pressão ou ângulo, dependendo do dispositivo de captura, de cada toque independentemente, permitindo diversos dedos, mãos (ou pessoas dependendo do tamanho do dispositivo) interajam provendo forma rica e intuitiva de interação, como por exemplo o monitor de operações do filme *Minority Report*.

Esta tecnologia tornou-se muito popular com ajuda do Youtube, em 2006, com vídeos do evento *Technology Entertainment Design Conference*, em Monterey na Califórnia, em que o pesquisador do Instituto de Ciências Matemáticas Courant (divisão de pesquisas e treinamento avançado de ciência da computação e matemática da Universidade de Nova Iorque), Jeffenson Y. Han, demonstra seu trabalho de pesquisa de interação multi-toque, utilizando uma superfície com display gráfico interativa, que permite duas mãos, reconhecimento multi-ponto e aceita múltiplos usuários, apresentando uma implementação elegante de varias técnicas e aplicações nessa superfície com projeção traseira. Essa difusão acarretou no interesse de diversas vertentes de pesquisa sobre esse modo interativo, e populando a web com diversos tutoriais de como montar este tipo de interface, e diversos weblogs com a experiência desses pesquisadores.

Algumas implementações multitoque são limitadas a somente dois dedos, por isso não é preciso reconhecer a interação de mais de uma pessoa para caracterizar aplicações deste tipo.

### História

Multitoque teve seu início em 1982, com os tablets mutitoque feitos na universidade de Toronto e com as telas multitoque dos laboratórios Bell. Nos anos 90 a universidade de Delaware desenvolveu um sofisticado sistema de reconhecimento de gestos e escrita, base para o mouse pad iGesture e teclados TouchStream comercializados pela FingerWorks em 2001. Estes teclados são reconhecidos pela sua ergonomia: aponte e arraste com dois ou mais dedos sobre as teclas (que eram impressas numa superfície macia), eliminando totalmente a necessidade de um dispositivo apontador (mouse). O primeiro dispositivo multitoque com display visual integrado comercializado foi o *Lemur Input Device*, um controlador multimídia profissional da companhia francesa JazzMutant lançado em 2005. Em julho de 2007, a Apple registrou cerca de 300 patentes com seu produto iPhone. E, meses depois, a Microsoft anuncia seu futuro produto, Surface, como grande inovação para garantir patentes.

### Microsoft Surface

Produto em fase de desenvolvimento, o Microsoft Surface Computer permite manipular arquivos e objetos virtuais através do toque, por exemplo imagens, vídeos, aplicativos e utilitários. Possui também a capacidade de reconhecer objetos colocados sob sua superfície, como palmtops, celulares e semelhantes e até mesmo cartões de crédito, sendo ativada, a partir do reconhecimento visual, um interface de interação na própria mesa com o objeto utilizando tecnologia Wi-Fi ou Bluetooth.

#### Especificação

O Surface possui uma tela de 30 polegadas disposta como uma mesa, 22 polegadas de altura (56cm), 21 polegadas de profundidade (53cm), e 42 polegadas de largura (42cm). Sua superfície é de acrílico e o interior de aço com pintura eletrostática. A plataforma é um Windows Vista customizado, e possui conectividade Ethernet 10/100, wireless 802.11 b/g e Bluetooth 2.0.

O Microsoft Surface já vem preparado para reconhecer vários aparelhos com tecnologia bluetooth 2.0 e wireless 802.11 b/g, basta colocar o aparelho sobre a tela que o sistema reconhecerá e irá mostrar as opções de interação com o [periférico](http://pt.wikipedia.org/wiki/Perif%C3%A9rico).

A visão computacional é criada por led infravermelho direcionado a superfície e cinco câmeras de resolução 1280 x 960 permitem detectar o toque.

#### Interação

A interface possui quatro componentes principais: interação direta, contato multitoque, multi usuário e reconhecimento de objetos. Permitindo diversas pessoas interajam simultaneamente com as aplicações de forma simples e intuitiva, com a capacidade de reconhecer a presença de dispositivos na superfície, e que possa trocar informações com esses a partir de uma de suas opções de conectividade.

### ReacTable

Instrumento musical colaborativo, desenvolvido pela Universidade Pompeu Fabra, situada em Barcelona, permite o reconhecimento de objetos e com a possibilidade de interação multi usuário, a *Reactable* sintetiza sons gerados através de fontes, filtros e osciladores. Cada objeto, podendo este ser um cubo, cone, cilindro, entre outros, é classificado por um software a partir de imagens de fiduciais situados no fundo dos mesmos e capturados por uma câmera. Assim cada objeto é classificado como um dos geradores e filtros obtendo-se como resultado um som único vindo da interação destes objetos. Este instrumento utiliza como base o software de detecção de marcadores fiduciais *ReacTIVision* para o reconhecimento dos objetos sobre a mesa.

#### Especificação

É uma mesa redonda e translúcida, utilizada numa sala escura, e aparenta como um display iluminado por trás. Por baixo da mesa fica câmera utilizada para a visão computacional. E um projetor embaixo da mesa conectado ao computador.

A parte tangível são peças transparentes com fiduciais impressos em suas laterais, imagens preto e branco compostas por círculos e pontos em varias formas, para o uso do software reacTIVision.

#### Interação

Os diversos tipos de tangíveis representam diferentes módulos de um sintetizador analógico, alguns desses interferem com o funcionamento do outro. Quando posicionada uma peça sobre a mesa, em alguns casos aparece um controle do tamanho de onda que pode ser mudado com o toque do dedo.

### iPhone

O iPhone é um smartphone desenvolvido pela Apple Inc. com funções de iPod, câmera digital e Internet. A interação é feita através de uma tela sensível ao toque. A partir desse aparelho a Apple registrou mais de 300 patentes.

Explicar que o iPhone é um marco em divulgar as interfaces multi-roque para o mercado de massa, e também por familiarizar a idéia e aumentar as expectativas dos usuários de encontrarem mais apliacações amparadas em paradigmas parecidos.

#### Especificação

Processador de 620 MHz ARM 1176[2], underclocked to 412mhz , 128 MB DRAM, amazenamento flash de 8 ou 16 GB, conectividade: Headphone, USB, FireWire,Wi-Fi (802.11b/g), Bluetooth 2.0+EDR e uma tela muntitoque sensível ao calor equivalente a pele humana.

#### Interação

A tela widescreen responde a três sensores: sensor de proximidade que desliga o monitor e a tela quando o iPhone esta no rosto, prevenindo gasto de bateria, e que o rosto ou orelhas interajam com o dispositivo, um sensor de luz ambiente que ajusta o brilho do display, e um acelerômetro de 3 eixos, para orientação do telefone e mudar a tela de acordo com sua posição, e os toques são reconhecidos por mapeamento do campo elétrico em uma película que fica entre uma tela protetora e o display LCD.

## Jogos e a interatividade

Desde os primeiros jogos que foram tornados públicos já se estudavam novas formas de interatividade com o jogador, para melhor o envolverem na história e ambiente do jogo para melhor desfrutar desta diversão, como por exemplo volantes para jogos de corrida, pistola para westerns e policiais, e manches para aeronaves. Seja a utilização de celofane nos *arcades* de space invaders para o surgimento do primeiro jogo “multicolorido”, destacando assim os alienígenas, ou para a própria disposição das telas e dispositivos controladores como pode ser vista em diversos simuladores de corrida e de vôo em áreas de lazer de shoppings, a interatividade sempre foi um dos focos de pesquisa da indústria do game.

### Controles Wireless com dispositivo de posicionamento

Em algum momento vocês precisam demarcar o fim do capítulo da introdução e chamar oficialmente de início dos fundamentos teóricos. Quando chegar este momento, precisam de uma figura para visão geral do projeto e um diagrama de blocos que permita entender sua arquitetura

Pessoal, acho que a chance de vocês é falar de interação multi-toque baseada em gestos, e esquecer um pouco os diversos gadgets de interação de jogos que foram inventados.

* Sugiro procurar uma lista de boas aplicações multitoque, independente de ser para o iPhone, Surface ou telas experimentais.
* Estamos atrás do que são boas idéias em termos de interação multitoque, exemplos de aplicação mesmo e descrição de como se interage com elas
* Estamos também atrás de saber quais são os gestos comuns que costumam compor um vocabulário de interação multitoque (coisas como “aproxime dois dedos para diminuir imagens”).

### Realidade Virtual

#### Realidade Aumentada

### Jogos de Estratégia

Do grego *stratègós,* algo como liderança(ago) do exercito(stratos), tinha o significado de liderança do exército. Por isso grande parte, dos jogos de estratégia, envolvem embates militares e distribuição de recursos, entre nações, povos, condados, planetas ou qualquer que seja a história que justifique a divergência.

Existem duas características básicas para os jogos de estratégia, se ele é tempo real ou turnos, e se são abstratos ou simuladores.

Na dimensão temporal, jogo de tempo real não existe divisão de tempo ou de permissão para jogadores terminarem sua jogada, ambos podem agir assim que tomadas as decisões sem necessidade de esperar sua vez, como ocorre no jogo em turno, que cada jogador deve esperar seu momento de agir, enquanto espera as ações de seu/seus adversário(s). Em alguns jogos virtuais podemos encontrar ambas características em mini-jogos dentro do contexto principal, sendo que para algumas ações são todas em tempo real, quando os jogadores entram em combate muda para um ambiente de turno.

Em termos de realidade, trata-se da pureza de estratégia do jogo em relação a ambiência, que enquanto alguns jogos tentam reproduzir fielmente as guerras napoleônicas, ou as conquistas do império romano, enquanto em outros, tudo que é acessório a estratégia é retirado, como por exemplo o go e damas.

#### Go

#### Age of Empires

### Jogo de RPG

RPG é a sigla de roleplaying game, pode ser traduzido como jogo de interpretação, ou representação de personagens, é um tipo de jogo em que os participantes assumem papeis de personagens e criam a história colaborativamente, e seu progresso segue a partir da improvisação dos jogadores dentro de um sistema de regras predeterminado, determinando o progresso do jogo.

Os RPGs são tipicamente mais colaborativos e sociais do que competitivos. Um jogo típico une os seus participantes em um único time que se aventura como um grupo. Um RPG raramente tem ganhadores ou perdedores. Isso o torna fundamentalmente diferente de outros jogos de tabuleiro, jogos de cartas, esportes, ou qualquer outro tipo de jogo. Por esses motivos a NASA, em missões espaciais longas, utiliza o RPG como forma de entretenimento nas missões, evitando o conflito entre os tripulantes, como no caso das tripulações russas quanto ao jogo de xadrez, e para reduzir o tédio, estresse e depressão que essas viagens podem causar [ FOLHA].

O RPG é um jogo pouco convencional quando comparamos com os jogos habituais. Em um teatro, os atores recebem seu guião (ou "script"), o conjunto de suas ações, gestos e falas, com tudo o que suas personagens devem saber. Você interpreta uma personagem de ficção, seguindo o enredo definido em um roteiro. Num jogo de estratégia, por outro lado, você está seguindo um conjunto de regras onde, para vencer, você precisa vencer desafios impostos por seus adversários - cada partida é única, já que é impossível prever seus movimentos durante o jogo. No RPG, esses dois universos se unem.

Como em um jogo de estratégia, há regras que o definem, e guiam aquilo que o seu personagem pode ou não pode fazer. A esse conjunto de regras chama-se sistema. Como no teatro, cada personagens tem uma história, e deve ser interpretado assim como os atores fazem. Mas, diferente de um jogo de estratégia, você não luta contra um adversário específico, mas vive aventuras em um mundo imaginário. Diferentemente do teatro, você não segue um roteiro, mas age pelo seu personagem com liberdade de ação.

Como romances ou filmes, RPGs agradam porque eles alimentam a imaginação, sem, no entanto, limitar o comportamento do jogador a um enredo específico. Este enredo pode ser orientado a um tipo comercial especifico de livro de RPG como no caso do Dungens & Dragons, ou o jogadores podem desenvolver o seu próprio enredo e sistema como era feito como antigamente, em que boa parte era baseada em livros ou filmes, como no caso do famoso Senhor dos Anéis de J. R. R. Tolkien, que influenciou a criação de diversos RPGs.

#### Dungeons & Dragons

Dungeons & Dragons (abreviado como D&D ou DnD) foi o primeiro RPG comercial publicado em 1974 nos EUA , isto é, é um livro que retrata um universo, neste caso de fantasia medieval , e seu sistema, sobre os quais os jogadores podem interpretar. É um jogo bem estruturado, em que cada jogador controla um único personagem, e cada personagem tem uma área de atuação (bardo, monge, gerreiro, entre outros dentre o universo D&D), atuando em grupo e a partida geralmente dura variais sessões, que são chamadas de aventura e esse conjunto de sessões é chamada campanha.

Os resultados das escolhas do grupo e a historia básica do jogo é determinada pelo Mestre de acordo com as regras do jogo e com a interpretação do mestre dessas regras. O Mestre escolhe e descreve os vários Personagens do Mestre (PDMs), que o os personagens encontram, o cenário em que essas relações ocorrem, e is resultados desses encontros baseados nas escolhas e ações dos jogadores. As extensas regras do jogo, que cobrem áreas diversas como interações sociais, uso de magia, combate e o efeito do ambiente nos personagens, ajudam o mestre em suas decisões. O Mestre do Jogo pode escolher quais regras publicadas ele vai usar e até mesmo criar regras novas, se achar necessário.

### Jogo RPG Eletrônico

Os primeiros jogos de RPG para computador surgiram no início dos anos 70 baseados na série *Dungeons & Dragons*. Ganharam popularidade durante a década de 80 e hoje é um dos genros de jogos mais populares de todo o planeta.

Um RPG (*Role Playing Game*) é um tipo de jogos onde o jogador incorpora um personagem fictício e através dele participa da criação de uma determinada história. O jogador determina que ações este personagem deva fazer e estas são impostas a um sistema de regras. Agindo de acordo com estas regras, o personagem evolui e faz a sua própria história. Segundo um estudo de *Gwendolyn Kestrel*, Ph.D. em educação, *RPGs* podem ampliar a capacidade de leitura, interpretação, e raciocínio lógico de seus jogadores, além de estimular o pensamento criativo .

Os primeiros *RPGs* foram criados para computadores eram jogados através de comandos, estilo mainframe, onde cada comando determinava a ação que o personagem deveria executar. A interface era em *ASCII* e as ações que poderiam ser feitas eram limitadas a movimentos e ataques à monstros imaginários.

Com o início dos videogames em formato console durante a década de 80, o genro passou a se popularizar e ganhar jogos mais inteligentes e visualmente mais interessantes. O personagem agora é visível e os objetos com os quais ele interage também. No final da década, séries, hoje em dia populares, começaram a ser lançadas, como por exemplo, *Final Fantasy* e *Might and Magic*.

  
Final Fantasy – Square - 1987

Já na década de 90, os jogos de *RPG* deixaram ser *2D* e passaram a ser *3D*. Cada vez com histórias mais envolventes e ações mais complexas, ganharam mais adeptos ao gênero. Grande exemplo desta popularização são os jogos *Final Fantasy VII* e *Final Fantasy VIII*, sendo que o primeiro ganhou uma continuação em filme.

  
Final Fantasy VII – Squaresoft - 1997

Juntamente com a popularização da internet, um novo gênero de *RPGs* foi criado: o MMORPG (*Massively Multiplayer Online Roling Playing Game*). Nesse gênero vários jogadores conectados pela internet interagem seus personagens uns com os outros dentro de um mesmo mundo virtual. Devido à grande possibilidade de interação com jogadores de qualquer parte do mundo, esse gênero de jogo é sucesso hoje em dia. Exemplos mais recentes são os jogos: *Ultima Onlin*e, *Ragnarok Online*, *World of Warcraft*, *Lineage*, entre outros.

  
World of Warcraft – Blizzard - 2004

#### RPG Tático x RPG Ação

Os RPGs Eletrônicos são divididos em vários gêneros, cada um tentando compensar mais uma característica de RPG, seja tentando chegar mais próximo do RPG original, ou valorizando uma característica para um determinado publico alvo. entre elas se destacam o RPG Tático, como Final Fantasy TacTics e Ogre Battle, e RPG Ação a exemplo Diablo e Fable.

RPGs de ação o combate é mais rápido e ágil, normalmente basta enviar o personagem contra o inimigo que a batalha se desencadeia. Já os Táticos o combate é estratégico, com posicionamento de personagens, normalmente envolve turnos de combate, e administração de recursos.

#### Final Fantasy Tactis

## Objetivos

Objetivo desenvolver um jogo semelhante ao Final Fantasy Tactics incorporando e explorando novos recursos a mesa multi-toque desenvolvida por alunos do Centro Universitário Senac

A construção do protótipo utilizará a mesa multitoque com o intuito de acrescentar novos recursos ao jogo, melhorando a jogabilidade e dinamismo do jogo.

O jogo permitirá múltiplos jogadores simultâneos podendo controlar um ou mais personagens por posicionamento de fiduciais ou opções no menu de interação multitoque.

O uso de fiduciais corresponderá ao posicionamento e movimentação do personagens na mesa, sendo verificada pela regra de jogo as posições possíveis de acordo com terreno e características do personagem, terrenos íngremes terrão acesso restrito ao condicionamento físico do personagem, assim como uma movimentação mais lenta em comparação com planícies.

A quantidade de personagens será definida no inicio do jogo, em que cada jogador decide qual dos 2 exércitos possível vai fazer parte, e deve colocar em seu território disponível o numero de personagens que deseja controlar.

Havia algumas figuras no paper de vocês para tentar explicar o jogo, poderiam entrar aqui

O toque dos jogadores na superfície interativa, para a utilização dos menus de controle de personagem, servirá como ordem de movimentação, em que aparecerá o campo de movimentação possível para reposicionamento do personagem com fiducial; ataque ou magia, em que um sub-menu de opções de áreas de ataque a um ou mais adversários variam de personagem; defesa , para aumentar a probabilidade de defender um ataque do adversário durante aquele turno; item, para uso de itens de recuperação de vida.

## Proposta

O jogo desenvolvido é um *RPG* tático semelhante ao conhecido *Final Fantasy Tactics*, onde o jogador controla vários personagens com características diferentes, cujo objetivo é derrotar o inimigo através de ataques, magias e itens, utilizando táticas, como por exemplo, se beneficiar de uma determinada posição no campo de batalha para obter vantagens sobre o inimigo.

Transpondo a idéia do jogo para uma mesa multi-toque, decidimos utilizar objetos físicos para representar os personagens no campo de batalha e o toque seria utilizado para interagir com as ações que os personagens deveriam executar. Como para a detecção de objetos utilizaríamos fiduciais, e devido a problemas em sua detecção, o controle dos personagens também ficou a cargo do toque.

Com isso, o jogo ganhou portabilidade, uma vez que não é necessário utilizar uma mesa para jogá-lo. Qualquer superfície multi-toque em qualquer posição é suficiente para interagir com o jogo.

O jogo deve ser jogado por dois jogadores, sendo que cada jogador terá várias unidades de combate. Cada uma possui diversos atributos que quando configurados tornam-na única e diferente das demais em vários aspectos. Além de atributos, as unidades possuem classes que lhe dão características, vantagens, desvantagens e ações diferentes ampliando as possibilidades de estratégia de cada um dos times. O objetivo é, utilizando as características de cada unidade de combate e suas respectivas ações, derrotar todas as unidades do jogador adversário.

# Tecnologias

## Framework XNA

O Framework XNA é um conjunto de classes e bibliotecas que auxiliam no desenvolvimento de jogos.

O XNA permite somente programação em C#, uma linguagem da plataforma .NET bastante utilizada ultimamente devido à grande facilidade em integrar sistemas.

O XNA trabalha em camadas assim tornando a aplicação mais estável, de fácil atualização e ou criação de novas versões.

Entre as camadas encontram-se:

• Plataform

• Core Framework

• Extended Framework

• Games

### Plataform

Esta camada tem a responsabilidade no tratamento da programação mais próxima do hardware. Contem módulos que possibilitam o trabalho com teclado, mouse, placas de vídeos, placas de som entre outros.

• Direct3D – Gráficos 3D.

• XACT – Áudio e criação de áudio.

• XINPUT – Suporte a dispositivos de entrada (Teclado, Mouse..)

• XContent – Suporte a conteúdo de imagens, vídeos e texturas.

### Core Framework

Responsável pelas funcionalidades nativas do XNA que as outras camadas usam, como gráficos, áudio, entrada de dados, matemática e armazenamento.

### Extended Framework

Responsável pelo gerenciamento do jogo como parte de cálculos, sons e carregamento de texturas.

### Games

Camada de desenvolvimento dos jogos

Camada de desenvolvimento onde disponibiliza jogos prontos, códigos e conteúdos feito por outras pessoas.

## ReacTIVision

Utiliza-se no projeto a ReacTIVision por ser um ótimo, e open source, arcabouço para visão computacional voltado para reconhecimento de marcadores fiduciais fixados em objetos físicos.

Desenvolvido pelo Music Technology Group da universidade Pompeu Fabra de Barcelona como parte do projeto Reactable [FIGURA 2], um instrumento musical eletrônico com interface tangível que emite sons como resultados das interações de objetos fiduciais, representados na projeção como simples componentes musicais, sendo estes fontes, filtros e osciladores.

O ReacTIVision reconhece os objetos fiduciais tangíveis a superfície de interação em tempo real de captura da webcam, iniciando o processo convertendo a imagem fonte para branco e preto, então, segmenta em uma arvore de regiões brancas e pretas que será percorrida por um algoritmo que transformará essa região em um símbolo fiducial, e finalmente comparará com uma biblioteca, retornando um identificador para aquele elemento caso encontrado.

Mensagens OSC utilizando o protocolo TUIO codificado com a presença de fiduciais, localizações, orientações e identificação são transmitidas para aplicações clientes via UDP.

## TouchLib

A tecnologia que será utilizada para obter a interação multi-toque baseia-se no princípio da reflexão total interna frustrada da luz, onde os raios de luz infravermelhos se propagam pelo interior do acrílico.

Quando há um toque do dedo, a luz encontra um índice de refração maior, sendo espalhada. Desse modo, com o auxílio de uma câmera para descobrir onde o toque ocorreu, e com uso da biblioteca TouchLib, implementada sobre OpenCV, para reconhecer esses sinais infra-vermelhos como toque e enviará eventos como dedo pressionado, dedo movido, e dedo retirado por protocolo TUIO em pacotes OSC via UDP para a aplicação cliente.

Embora a ReacTIVision também reconheça toques optamos pelo TouchLib para nos desvencilharmos das etapas de segmentação dos fiduciais que são feitas para fazê-lo no ReacTIVision;

## OSCPack

OscPack é uma biblioteca feita em C++ voltada para criação e leitura de pacotes OSC para diversas aplicações e servidores, portável para varias plataformas e muito robusta. Facilitando a construção, envio, recibo e leitura de pacotes OSC para nosso arquitetura.

O OSC é um tipo de pacote criado pela biblioteca OSCPack voltado para a comunicação entre instrumentos musicais, computadores e equipamentos multimídia.

A primeira versão publica da Oscpack foi em auxílio da ReactTable que necessitava de uma biblioteca para manipulações de pacotes OSC. Portanto, sua escolha neste projeto deve-se à utilização do ReacTIVision e do TouchLib, que utilizam, ambos, o protocolo TUIO devidamente codificados em pacotes OSC para comunicação com a aplicação cliente via UDP. Desta forma aproveitamos a estrutura dos aplicativos com poucas mudanças, assim facilitando a implementação do projeto

# Projeto

Nos jogos de RPG o controle dos personagens por turno se torna cansativo devido à execução sempre da mesma alternância de controle, assim também limitando as estratégias do jogador.

Com a integração de novos recursos o jogo deixa de ser repetitivo passando a ser mais dinâmico, ágil e pelo fato dos turnos estarem associados a cada personagem a gama de estratégias aumenta consideravelmente.

Algumas vantagens em se utilizar uma mesa deste tipo é que a visão do jogo é natural de cima para baixo, deixando a visualização das unidades mais fácil, facilitando também na hora de selecionar ou de efetuar alguma ação com algum personagem, evitando passar por menus e interfaces semelhantes. Uma mesa desta é onde o contexto se trata de estratégia. Assemelha-se ao xadrez onde este é um jogo de pura estratégia.

Uma outra vantagem em utilizar este tipo de hardware é que o jogo mesmo em um computador comum seria forçado a ter turno pois os jogadores seriam forçados a utilizar o mouse um de cada vez ou utilizar dois computadores para jogar. E sendo no mesmo meio de ação fica muito mais fácil.

Novos recursos estes como eliminação dos turnos, possibilitando que vários jogadores controlem vários personagens simultaneamente, tornando o jogo mais ágil; campo de ação radial, trazendo maior realidade e precisão em cada movimento realizado pelo jogador; inclusão de times, permitindo que o jogador tenha mais de um inimigo para destruir, dificultando o jogo e formação de alianças.

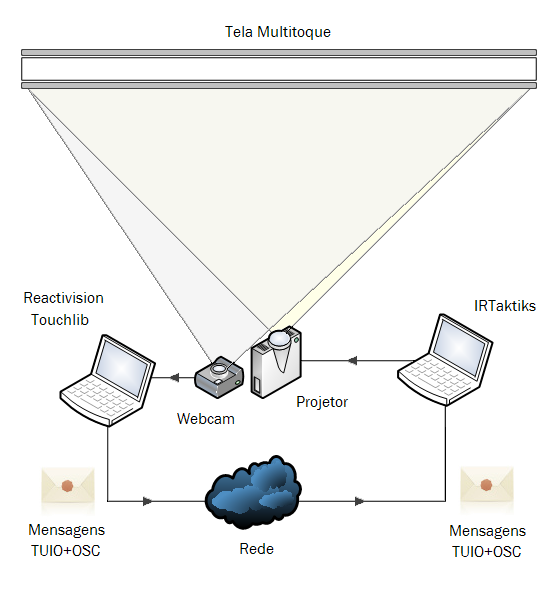
## Arquitetura

O aperfeiçoamento da mesa se dará através da otimização do circuito elétrico, de modo a operar em sua máxima capacidade, inclusão de uma superfície para projeção, utilização de uma webcam com melhor resolução de captura de imagens e inclusão de softwares para reconhecimentos de toques e fiduciais.

Para o reconhecimento de toques o software *Touchlib* foi utilizado, e para o reconhecimento de fiduciais, o software *Reactivicion* foi utilizado. Ambos os softwares utilizam o protocolo *TUIO* juntamente com o protocolo *OSC* para se comunicar com softwares externos.

O desenvolvimento do jogo se dará através do desenvolvimento de um protótipo inicial e uma versão final, sendo que está será o jogo com todas as funcionalidades codificadas. As funcionalidades do jogo bem como a sua implementação serão descritas neste capítulo.

Dessa forma, a arquitetura geral do sistema é mostrada a seguir. As ações que os usuários executarem sobre a mesa, como o posicionamento de um fiducial ou o toque de um ou mais dedos sobre sua superfície será reconhecida pelos softwares através da análise das imagens enviadas por uma *webcam*. O software processa as informações e envia uma mensagem *TUIO* para cada objeto ou dedo sobre a mesa, contendo as informações como posição, ângulo de movimentação, velocidades calculadas entre outras informações. Estas mensagens *TUIO* são empacotadas em envelopes *OSC* e enviadas via *UDP* para as aplicações clientes, no caso deste projeto, o jogo. O jogo lê estas informações e atualiza o estado do jogo, projetando-o através de um projetor sobre a mesa. Com isso o usuário tem a impressão de estar manipulando diretamente os objetos do jogo.

  
Arquitetura geral do sistema

## Mesa

## Jogo

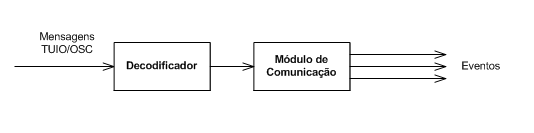
A primeira etapa do desenvolvimento do jogo foi escolher qual seria o ambiente de desenvolvimento. A escolha deveria ser baseada nas funcionalidades de comunicação das bibliotecas utilizadas para o reconhecimento dos toques e objetos sobre a mesa.

Como utilizamos o software *Touchlib* para o reconhecimento de toques e o *Reactivision* para o reconhecimento de fiduciais, e pelo fato de ambos utilizarem a mesma arquitetura de comunicação, bem como a mesma biblioteca, a *oscpack*, o jogo poderia ser desenvolvido em praticamente qualquer ambiente. Dessa forma a escolha foi baseada apenas em qual ambiente a produtividade seria maior e qual teria mais recursos. Dentre *frameworks* existentes, escolhemos o *Microsoft® XNA 2.0*, devido à enorme variedade de recursos disponíveis, documentação, desempenho e ganho de produtividade, uma vez que a linguagem de programação adotada seria *C#.*

### Projeto

O projeto do jogo, como dito anteriormente, consistiu no desenvolvimento de um protótipo seguido da versão final. Antes do desenvolvimento do protótipo, um módulo de comunicação entre o jogo e a mesa foi projetado e desenvolvido. Dessa forma, futuros problemas de integração seriam eliminados, uma vez que a construção do jogo levaria este módulo de comunicação em consideração.

Foi decidido que este módulo utilizaria eventos para representar as interações dos usuários com a mesa. Com isso, o projeto do jogo foi simplificado e modularizado. O serviço que lê as mensagens *TUIO*, e dispara os eventos, é executado em uma *thread* apartada, aumentando o desempenho do sistema de comunicação.

  
Visão geral do módulo de comunicação

#### Protótipo

O protótipo foi desenvolvido com o intuito de validar as tecnologias empregadas na concepção do projeto. Teve como foco de desenvolvimento a decodificação de mensagens *TUIO*, geração de eventos respectivos à ação realizada na mesa, e criação de um simples jogo utilizando o *Microsoft ® XNA*. Desta forma, após o desenvolvimento do protótipo, estaríamos seguros quanto à escolha dos softwares escolhidos para o reconhecimento dos toques e fiduciais, bem como da tecnologia empregada para o desenvolvimento do jogo.

Foi de extrema importância, pois possibilitou uma visão geral do problema que seria desenvolver um jogo com inúmeras regras. Com isso, sua arquitetura deveria ser bem planejada, ou então teríamos um jogo de fraco desempenho e de grande tempo de resposta.

Os testes realizados em cima do protótipo se mostraram bastante satisfatórios, uma vez que o tempo de resposta de uma ação foi praticamente instantâneo, que se trata de um requisito muito importante a este tipo de sistemas, já que a sensação de estar manipulando o objeto diretamente tem que ser sentida pelos usuários de qualquer aplicação em superfícies multi-toque.

  
Protótipo

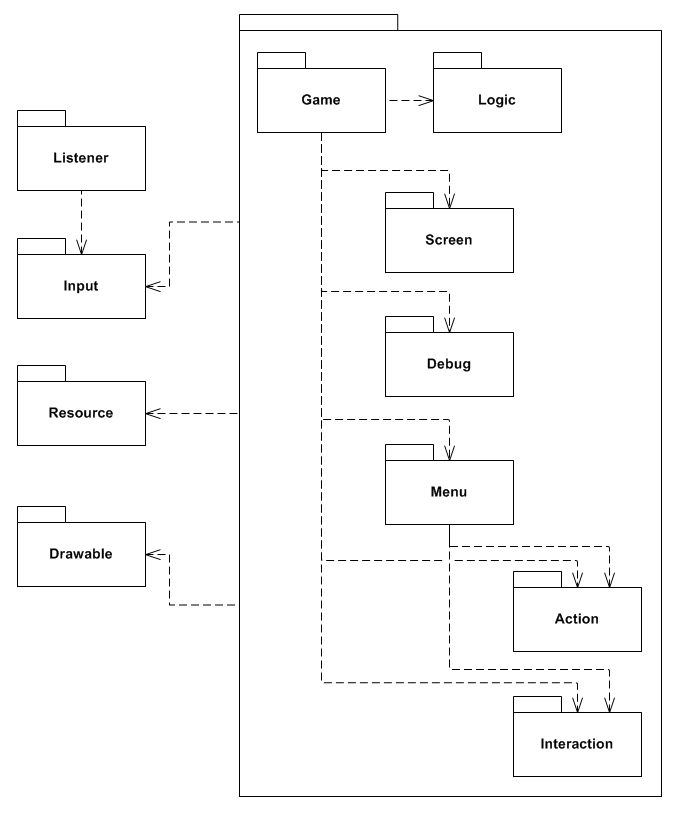
A arquitetura do protótipo não será explicada, pois grande parte foi reaproveitada no desenvolvimento do segundo protótipo, sendo mais apropriado o comentário apenas da arquitetura final.

#### Versão Final

O desenvolvimento da versão final teve como foco principal sua arquitetura. Foi trabalhada de modo a deixar o jogo o mais rápido possível, sem comprometer a qualidade e os requisitos propostos. A arquitetura foi dividida em diversos módulos, a fim de facilitar a implementação e extensão de funcionalidades, uma vez que com padrões definidos, a adição de novas funcionalidades é bastante fácil e ágil.

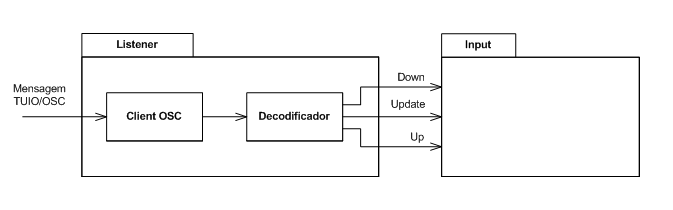
  
Versão Final

A arquitetura interna no *XNA* é centralizada na classe *Game*, que provê métodos para atualização e desenho de objetos, além de possuir uma lista de *GameComponents* e *Services*, que são atualizados e desenhados automaticamente pela classe *Game*. Internamente, o *XNA* cria uma *thread* para cada componente e serviço, não havendo, portanto, uma ordem prevista de execução. A vantagem desta arquitetura é a velocidade na execução, uma vez que várias *threads* executando paralelamente se beneficiam dos processadores *multi-core*, bastante comuns hoje em dia.

  
Arquitetura da versão final

##### Módulo Listener

Construído utilizando as bibliotecas do *oscpack*, é responsável por obter as mensagens *TUIO* enviadas pela mesa, decodificá-las e transformá-las em entradas para o jogo através da comunicação com o módulo Input. Baseia-se em uma arquitetura cliente-servidor, exercendo a função de cliente.

Visão do módulo Listener

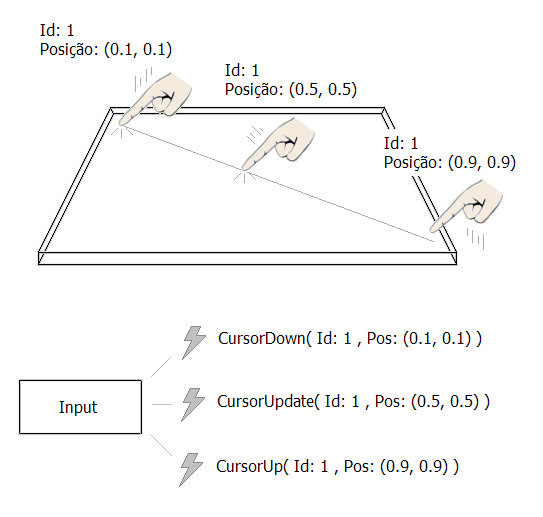
As mensagens *TUIO* possuem informações sobre cada um dos toques e objetos que estão sobre a mesa. Estas mensagens tratam toques sobre a mesa como cursores e objetos como objetos. Cada cursor ou objeto possui um identificador, servido de base para o reconhecimento de ações mais complexas como funcionalidades *drag-and-drop*, ou simplesmente arrastar e soltar. Além de identificadores, cada cursor e objeto possuem três tipos de mensagens diferentes: *Down*, *Update* e *Up*.

As mensagens *Down* são enviadas quando o objeto ou o cursor são criados, ou seja, quando o objeto é colocado sobre a mesa ou quando o dedo encosta sua superfície. As mensagens *Update* são enviadas para informar que o cursor ou o objeto estão ativos, em outras palavras, servem para informar que o objeto continua sobre a mesa, parado ou em movimento, ou ainda para informar que o mesmo dedo encontra-se sobre a mesa, também parado ou em movimento. Já as mensagens to tipo *Up* são enviadas quando o objeto ou o cursor são removidos, ou seja, quando removidos da superfície da mesa. Com estes três tipos de mensagens é possível rastrear qualquer tipo de movimento sobre a mesa, seja ele usando objetos, toques, ou até mesmo uma combinação de ambos.

##### Módulo Input

É responsável por gerenciar as entradas de ações por todo o jogo. Para isso, utiliza uma arquitetura que distribui eventos comuns a todos os componentes do jogo. Dessa forma, não existem problemas de integração, uma vez que a comunicação entre mesa e jogo é centralizada. Após a definição dos eventos e dos dados que estes enviam a quem os trata, bastou apenas utilizar este módulo para obter as informações sobre toques e objetos sobre a mesa.

Inicialmente foram projetados seis eventos. Três representariam as ações possíveis com objetos sobre a mesa e três, os toques. Com o descarte do uso de fiduciais e conseqüentemente de objetos, o modelo final dispõe apenas de três eventos: *CursorDown*, *CursorUpdate* e *CursorUp*, todos estes enviando em seus argumentos o identificador do cursor, ou dedo, e a posição em que se encontra.

  
Exemplo de eventos do módulo Input

O identificador trata-se de um número inteiro gerado automaticamente pelo software *Touchlib* enviado dentro da mensagem *TUIO*. A posição também é enviada nesta mensagem, porém escalonada em valores entre zero e um, permitindo fácil conversão. Com estes dois parâmetros, é possível rastrear seqüências de movimentos, como arraste e rotacionamento. O movimento de arrastar é dado como um evento *CursorDown*, seguido de inúmeros eventos *CursorUpdate*, encerrando com um evento *CursorUp*.

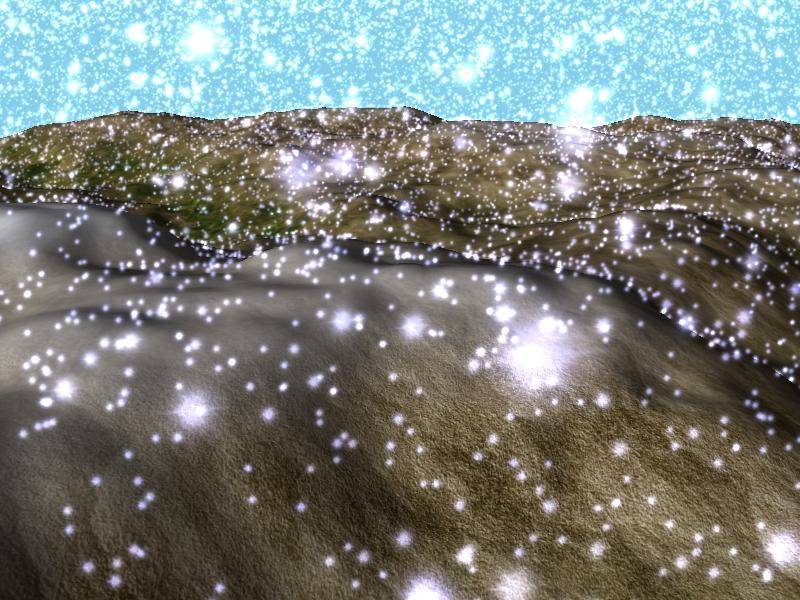
##### Módulo Resource

É o módulo responsável por gerenciar os recursos utilizados pelo jogo, como texturas, fontes, mapas, imagens e efeitos. Baseia-se em gerenciadores, que são responsáveis por carregar os recursos a partir de arquivos e transformá-los em objetos manuseáveis dentro do *framework* *XNA*. Existem três tipos de recursos utilizados pelo jogo: texturas, efeitos e fontes.

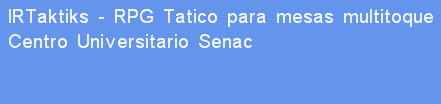
Texturas são imagens, em diversos formatos e codificações. São utilizadas no desenho das estruturas dos menus, imagens de fundo, personagens do jogo entre outros. São usadas também na geração do mapa e na criação de partículas.

Efeitos, por sua vez, são códigos em *HLSL* (*High Level Shader Language*), utilizados para aplicar efeitos específicos, píxel a píxel, durante a renderização de uma cena. A aplicação destes efeitos acontece dentro da própria placa de vídeo, não consumindo assim tempo de *CPU* da máquina que executa o jogo, mas por sua vez, requer uma placa de vídeo que possua suporte.

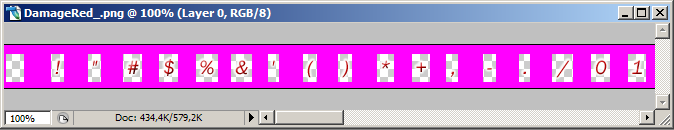
Por utilizar na renderização do mapa mesclagem de texturas, a fim de obter um mapa mais realista, instruções *HLSL* mais complexas foram utilizadas. Com isso, a placa de vídeo necessária deve ser capaz de compilar estas instruções utilizando *PixelShader* *3.0* e *VertexShader* *3.0*. Placas de vídeo com suporte nativo ao *DirectX 9.0c*, possuem esta característica.

  
Exemplo de utilização de efeitos *hlsl*

Já as fontes, são utilizadas na escrita de textos no jogo. Existem dois tipos de fontes utilizadas no desenvolvimento do jogo. A primeira é baseada em uma estrutura XML, onde as propriedades como espaçamento vertical, espaçamento horizontal, tamanho, cor e fonte são informados. Este tipo de fonte é a que consome menos memória, porém não é possível aplicar nenhum efeito.

  
Exemplo de uso de fonte XML

Quando se deseja que o texto escrito possua efeitos, como sombreamento, brilho, chanfros, contornos e texturas; é necessário o uso de fontes-textura. A fonte-textura trata-se de uma imagem que possui a seqüência os caracteres ASCII, já com os efeitos desejados, do número 32 aos 127. Cada caractere deve estar dentro de uma área de fundo totalmente transparente e entre elas deve existir em todas as direções, no mínimo, um píxel na cor magenta (R: 255; G: 0; B: 255; A: 255), para definir a separação dos caracteres, pelo *ContentProcessor* do framework *XNA*.

  
Exemplo de fonte-textura.

Com isso, é possível desenvolver os caracteres em qualquer software gráfico, como por exemplo, o *Adobe Photoshop*; e em seguida aplicar as regras acima de forma a utilizar a fonte no jogo. Como a aplicação das regras é algo trabalhoso e repetitivo, foi desenvolvido um programa que aplica as regras tendo como fonte o arquivo com os caracteres sobre um fundo transparente, facilitando o desenvolvimento de texturas para utilização no jogo.

  
Exemplo de uso de fonte-textura

Os gerenciadores de recursos trabalham com cache. O carregamento dos recursos é feito apenas quando a primeira solicitação acontece, e a partir disso, qualquer uso ao recurso será imediato, pois o mesmo estará já estará alocado em memória, sendo desalocado apenas ao fim do jogo.

##### Módulo Drawable

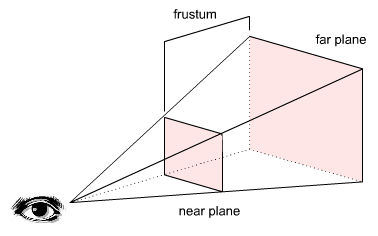
É o módulo responsável por gerenciar todas as operações de desenho do jogo, desde textos e texturas a mapas e animações. É subdividido em seis submódulos, cada um responsável pelo gerenciamento de uma entidade: *Camera*, *Sprite*, *Map*, *Area*, *Animation* e *Damage*.

Parte desse gerenciamento é feita utilizando uma lista de tarefas. Como a execução da atualização dos componentes não segue uma ordem específica, utilizando o mesmo gerenciador, é possível ordenar os componentes, através de uma prioridade, quando necessário.

###### Submódulo Camera

Representa a câmera do jogo, provendo matrizes de visão e projeção, ângulo de visão e posição do observador dentro da cena. Cada objeto 3D, como o mapa, áreas e animações, necessitam de uma matriz de projeção e visão para serem projetadas na tela de forma correta. A matriz de projeção define a área visível da cena, ou *frustum*; enquanto a matriz de visão representa as transformações que devem ser aplicadas para os objetos serem projetados corretamente na tela.

A área visível é representada de maneira semelhante a uma pirâmide, cortada por dois planos. O plano mais próximo da posição do observador é chamado de *near plane*, enquanto o mais afastado de *far plane*. Qualquer objeto que após a aplicação de suas transformações encontra-se fora desta “pirâmide” é automaticamente removido da renderização da cena.

  
Representação da área visível da cena

A altura e largura do *viewport* da tela de jogo coincidem com as do *near plane*. Devido às especificidades de cada objeto 3D, decidiu-se por utilizar matrizes de projeção e visão individuais.

###### Submódulo Sprite

Gerencia o desenho de *sprites*, ou seja, textos e imagens, utilizando prioridades para ordenar os componentes que serão desenhados. Quando um objeto necessita se desenhar na tela, ele transfere esta responsabilidade ao gerenciador, que armazena o que deve ser desenhado, e a cada atualização do jogo, a lista é ordenada e os itens armazenados são desenhados.

  
Exemplo de sobreposição de *sprites*

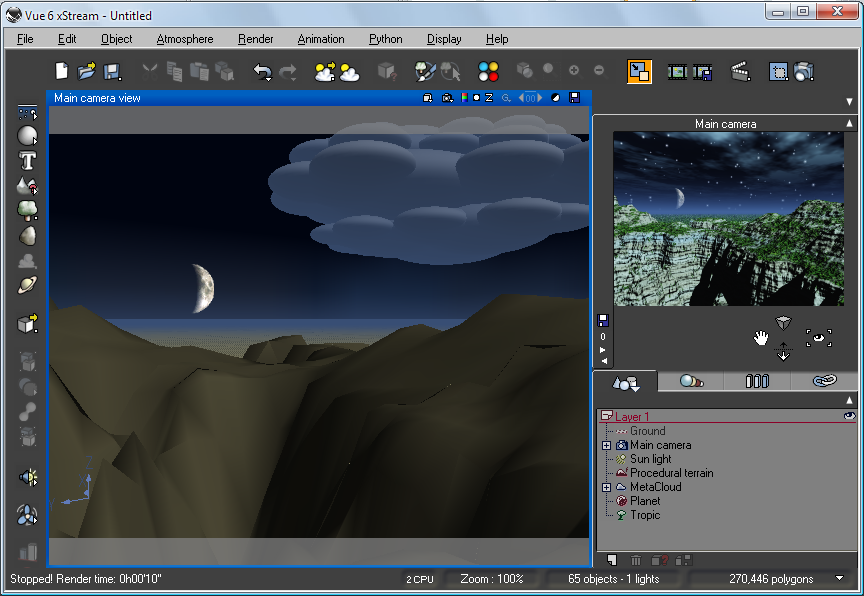
###### Submódulo Map

Responsável pela criação e desenho do mapa sobre o qual o jogo acontece. O mapa, diferentemente dos *sprites*, trata-se de um objeto 3D, fazendo uso do submódulo *Camera*.

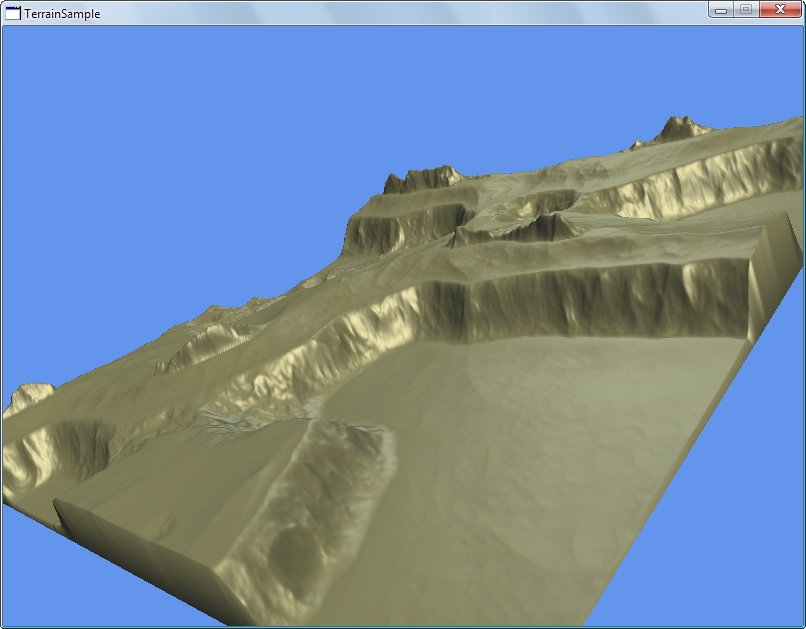
Não há gerenciadores que controlam o seu desenho, pois não existe a possibilidade de existirem dois mapas sendo desenhados ao mesmo tempo. O mapa é sempre o primeiro item a ser desenhado, uma vez que todos os personagens devem estar sobre ele.

Inicialmente, optou-se por utilizar um arquivo de geometria para representar o mapa com suas texturas. Para sua criação, utilizou-se geradores automáticos de terrenos, como o *Terragen 2*, *L3DT*, *Nem's Mega 3D Terrain Generator* e o *Vue xStream 6*. Estes programas exportam arquivos em diversos formatos, com texturas embutidas, que são facilmente reconhecidos por diversos softwares de modelamento, como o *Autodesk 3ds Max* ou *Softimage XSI*.

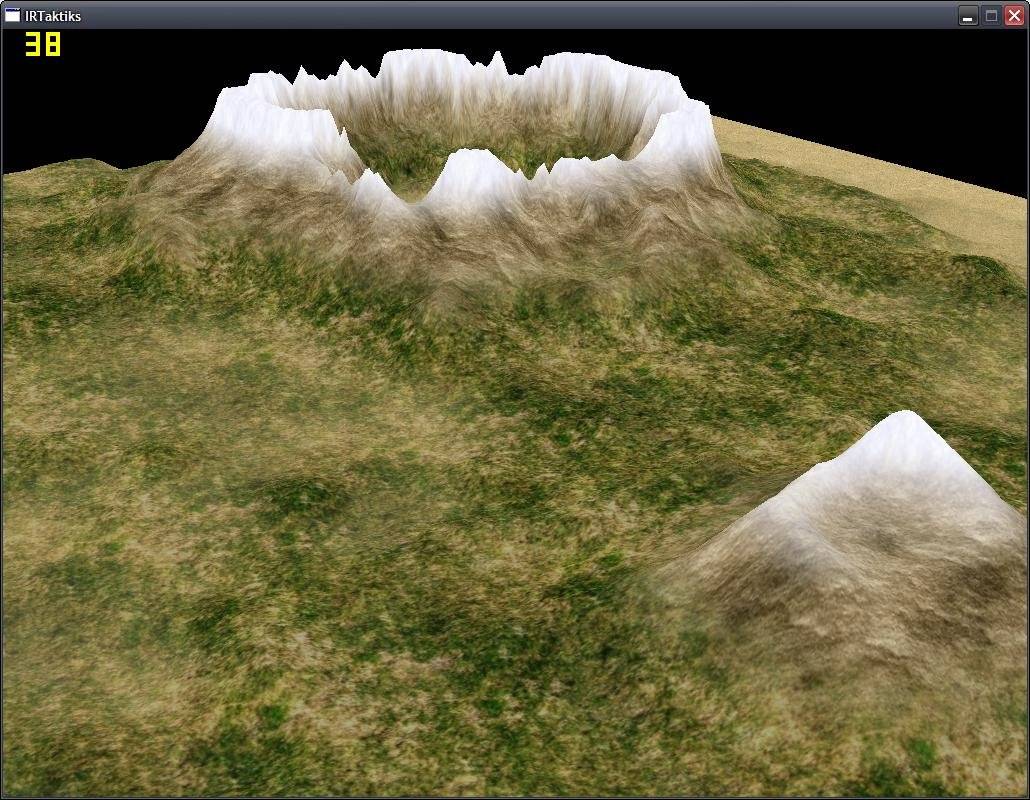
O framework *XNA*, por trabalhar intimamente com o *Microsoft DirectX*, apenas é capaz de reconhecer arquivos de geometria de formato *X*. Para a geração destes arquivos foi utilizado o software *Autodesk 3ds Max*, que possui um *plugin* otimizado para a integração com o *XNA*. O software escolhido para gerar os terrenos foi o *Vue xStream 6*, pois além de exportar a cena para o formato próprio do *Autodesk 3ds Max*, possui alta qualidade de geração, diversas texturas e algoritmos de geração.

  
Software Vue xStream 6

Após a criação do mapa, este foi importado para dentro do jogo. Apesar de o arquivo exportado conter as texturas e estas estarem referenciadas o XNA não conseguia interpretá-las. Tentativas de aplicação da textura por código dentro no jogo também não surtiram efeitos, e outra abordagem para a geração do mapa teve que ser pensada. Outro fator que nos levou a tomar esta decisão foi a quantidade de memória consumida (aproximadamente 50Mb) e o tamanho do arquivo de geometria (aproximadamente 40Mb).

  
Mapa utilizando arquivo de geometria (40Mb)

A segunda alternativa para a geração do terreno, seria a utilização de um arquivo de imagem, monocromático chamado de *heightmap*, ou mapa de altura. Este arquivo também pode ser gerado através do *Vue xStream 6*. O tamanho do arquivo determina o tamanho do terreno que é gerado, enquanto cada píxel determina um vértice do terreno. A altura de cada um destes vértices é dada pela intensidade da cor branca do píxel em questão. O jogo lê o arquivo, criando os vértices para cada píxel lido e um efeito *hlsl* aplica uma textura sobre cada um destes vértices. Para que o terreno tivesse bastante realismo, foi utilizado um efeito que mescla quatro texturas, baseando-se na altura do vértice. Para que este efeito rode corretamente, a placa de vídeo deve possuir suporte nativo ao *DirectX 9.0c*.

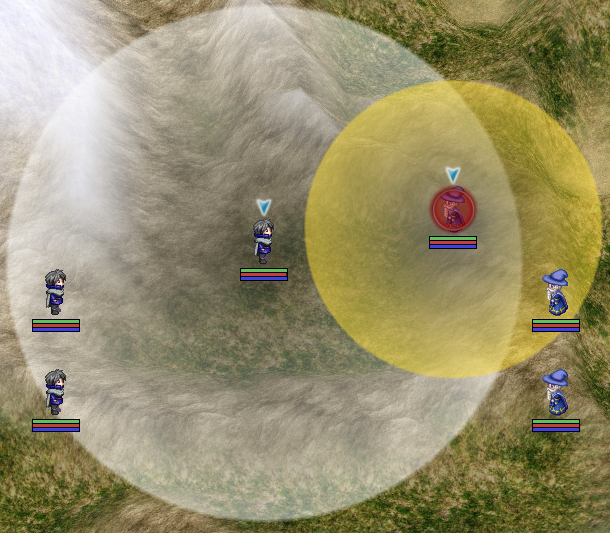
  
Mapa usando *heightmap* e efeito *hlsl* de mesclagem

###### Submódulo Area

Responsável pela criação e representação de uma área circular sobre o mapa. É utilizado para impor limites nas ações dos jogadores. Assim como o mapa, também se trata de um objeto 3D, porém com uma das dimensões igual a 1.

Apesar de uma área circular não ser 3D, isso se deu ao fato do framework *XNA* não desenhar nenhuma primitiva simples, retas, elipses e retângulos; além de triângulos, dada uma lista de pontos no espaço.

Da mesma maneira que o submódulo *sprite*, há um gerenciador para controlar o desenho das áreas, uma vez que pode existir mais de uma área sendo desenhada ao mesmo tempo. Este gerenciador também trabalha com uma fila de prioridades. Por decisão, todas as áreas são desenhadas após o desenho do mapa e antes de qualquer *sprite*, para que os personagens, textos e menus não sejam sobrepostos pelas áreas.

  
Exemplo de utilização de áreas

Da mesma maneira que o mapa, também utiliza um efeito *hlsl* para definir uma cor para a área, e aplicar uma suavização no serrilhados do desenho de sua geometria. O preenchimento da cor acontece de maneira interpolada das bordas para o centro, aplicando um efeito de degradê. Já a suavização foi feita utilizando um algoritmo de *anti-aliasing* com o canal *alpha* da cor da área.

  
Comparação entre áreas com (esquerda) e sem (direita) suavização

###### Submódulo Animation

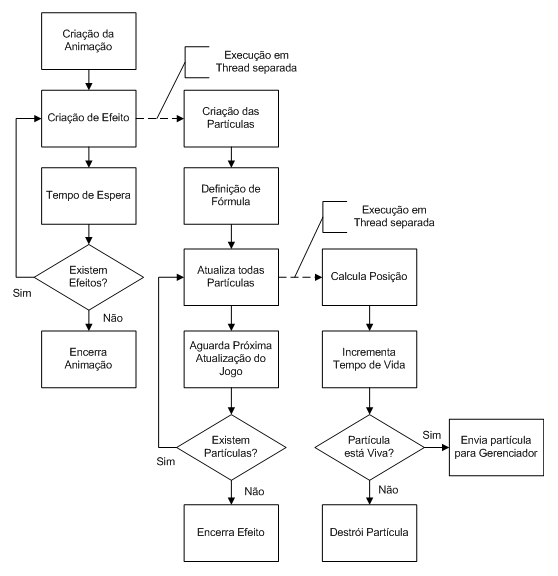
É o responsável por executar animações em uma determinada posição no espaço. As animações são um ou mais efeitos de partículas executados ao longo do tempo. Efeitos de partículas são diversos pontos (partículas) que de movimentam no espaço, baseadas em fórmulas matemáticas que regem seu movimento.

A cada ação executada no jogo, uma animação é invocada, de modo a responder ao usuário um *feedback* visual sobre a sua interação. Foram codificadas cerca de vinte animações diferentes, de modo que cada ação possua sua respectiva animação, permitindo a associação ação-animação por parte do jogador.

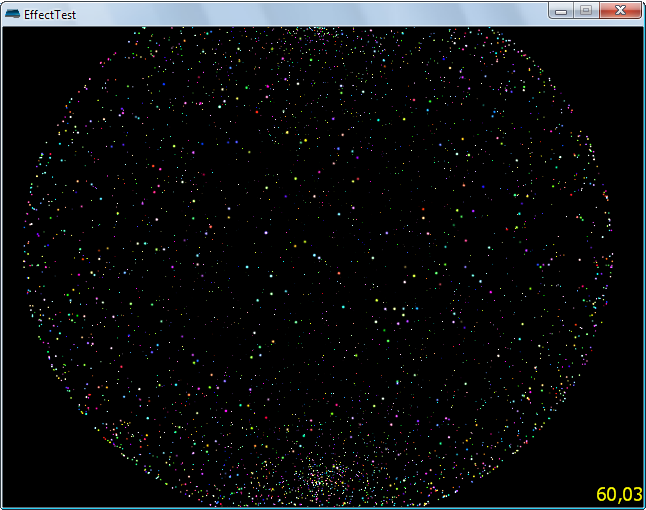
De modo a não comprometer a performance do jogo, cada animação é executada em uma *thread* separada. Com isso, é possível iniciar efeitos em tempos diferentes, uma vez que enquanto a animação espera antes de iniciar um novo efeito, o jogo continua sua execução normalmente.

Como cada animação pode ser composta por um ou mais efeitos de partículas diferentes, o controle de cada um destes efeitos exigiu o desenvolvimento de um gerenciador de partículas. Este gerenciador é responsável por desenhar todas as partículas existentes no jogo em um determinado momento. A animação somente invoca os efeitos na ordem correta e no período de tempo calculado, enquanto o próprio efeito se encarrega de solicitar o desenho de suas partículas ao gerenciador.

Cada uma das partículas de um determinado efeito possui suas características próprias, como posição, cor, tempo de vida atual e tempo total de vida. Quando o tempo de vida de uma partícula atinge o tempo total de vida, ela automaticamente se destrói, não sobrecarregando o jogo, calculando posições de partículas de efeitos que não existem.

  
Fluxo de execução de uma animação

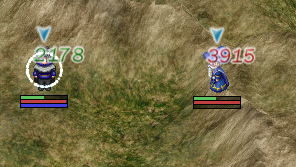
O gerenciador de partículas utiliza ainda, um efeito *hlsl* para determinar a cor de cada um das partículas e sua escala. De acordo com o tempo de vida da partícula, a cor desta vai de tornando transparente, de modo a fazer uma transição suave durante o fim da animação. A escala é baseada na distância da partícula em relação ao observador da câmera. Quanto mais próxima a partícula estiver do observador, maior ela será, enquanto mais distante do observador, menor ela será; tornando o efeito o mais realístico possível.

  
Exemplo de efeito de partículas

###### Submódulo Damage

Responsável por exibir informações sobre os efeitos das ações entre as unidades. Exibe informações como: quantidade de dano sofrido, quantidade de mana utilizada, quantidade de pontos de vida recuperados e atributos que tiveram seu valor aumentado.

A exibição destas informações ajuda na visualização do andamento do jogo e da quantização por parte do jogador, da força que sua unidade possui. Todas as informações benéficas à unidade são informadas na cor verde, enquanto as maléficas, na cor vermelha.

  
Exemplo de exibição de informações

##### Módulo Game

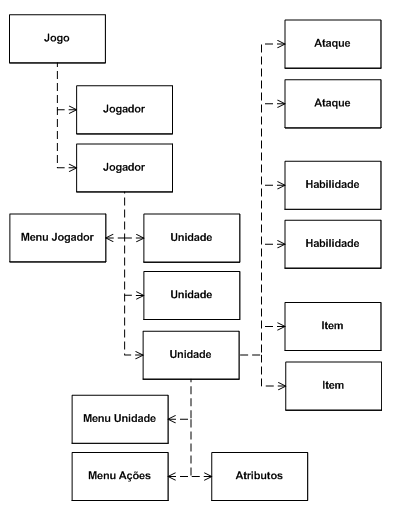
É o módulo central do jogo. Contém as classes que representam o jogo (*IRTGame*), os jogadores (*Player*) e as unidades (*Unit*), que são os personagens que o jogador controla. A classe que representa o jogo herda da classe *Game*, do *framework XNA*, possuindo funcionalidades de atualização e desenho. As classes que representam os jogadores e as unidades herdam da classe *GameComponent*. Com isso, ambas podem ser componentes da classe *IRTGame*.

Cada jogador possui nome, e uma lista de unidades que este irá controlar no jogo. Uma unidade possui diversas características, como nome, pontos de vida, pontos de mana, tempo de espera após a execução de uma ação, classe, atributos e ações. Os atributos foram separados em outro módulo, devido ao fato de serem utilizados em todos os cálculos do jogo. As ações, por sua vez, se subdividem em três tipos: Ataques, Habilidades e Itens. Cada unidade possui uma lista de cada um destes tipos de ações.

As ações, assim como os atributos, também foram implementados em outro módulo, responsável pela criação das ações, e pelo gerenciamento dos acontecimentos que a execução da ação provoca entre as unidades.

Ambos, jogadores e unidades, possuem menus, que informam o jogador real o estado atual de suas unidades, e propiciam a interação com o jogo, através da execução de ações. Todos os menus, assim como os atributos e ações, também foram implementados em um modulo separado, com seus respectivos gerenciadores.

Cada unidade possui indicadores, que auxiliam o jogador a ter uma visão geral sobre o jogo de maneira rápida e eficiente. Existem quatro indicadores, no total, para cada unidade dentro do jogo. Três delas se encarregam de mostrar, em forma de barras, a quantidade de pontos de vida, mana e tempo de espera após uma ação; enquanto a quarta se encarrega de indicar se a unidade está selecionada ou não. Quando o jogador seleciona uma unidade, ela pode executar ações; desde que o tempo de espera tenha de passado.

  
Estrutura organizacional do jogo

##### Módulo Logic

Responsável pela representação lógica das unidades, como orientação, atributos e classes.

A orientação indica em qual direção a unidade está olhando. Pode possuir os valores: *Norte*, *Sul*, *Leste* ou *Oeste*.

Os atributos determinam as características da unidade, e são divididos em dois tipos: fixos e calculados. Os atributos fixos são formados por valores entre 1 e 99 e não dependem de nenhum outro fator para determinar seu valor. Os atributos calculados, por sua vez, são regidos por fórmulas baseadas nos atributos fixos e sofrem influências da classe a qual a unidade pertence. Uma classe que beneficia um determinado atributo terá neste um valor maior, do que uma classe que não o bonifica; para os mesmos valores de atributos fixos.

Os atributos fixos de uma unidade são:

* Nível
* Força
* Agilidade
* Vitalidade
* Inteligência
* Destreza

Os atributos calculados são:

* Pontos de vida total
  + Quantidade máxima de pontos de vida de uma unidade.
* Pontos de mana total
  + Quantidade máxima de pontos de mana de uma unidade.
* Ataque
  + Utilizado no cálculo do dano de ataques físicos.
* Defesa
  + Utilizado no cálculo da defesa contra ataques físicos.
* Ataque mágico
  + Utilizado no cálculo do dano de ataques mágicos.
* Defesa mágica
  + Utilizado no cálculo da defesa contra ataques mágicos.
* Taxa de desvio
  + Porcentagem de desviar de um ataque físico.
* Taxa de acerto
  + Porcentagem de acertar um ataque físico.
* Alcance de Ataque
  + Tamanho da área onde um ataque pode ser desferido.
* Alcance de Habilidade
  + Tamanho da área onde uma habilidade pode ser solta.
* Alcance de Movimento
  + Tamanho da área para onde a unidade pode se mover.
* Tempo de espera
  + Tempo de espera após a execução de uma ação.

As classes definem as habilidades que a unidade poderá utilizar e influenciam nos valores dos atributos calculados. Cada atributo calculado possui fatores que, de acordo com a classe, podem aumentar ou diminuir o seu valor calculado. Com isso, unidades que possuam os mesmos valores de atributos físicos podem ter atributos calculados diferentes, aumentando a diversidade de estratégias que o jogador pode formar. Há seis classes disponíveis no jogo:

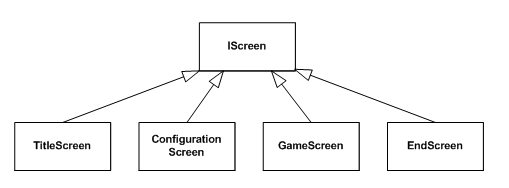
* Cavaleiro
  + Possui habilidades de ataques físicos
  + Bônus em ataques físicos e pontos de vida.
* Paladino
  + Possui habilidades de cura e sacrifício.
  + Bônus em pontos de vida, defesa física e mágica.
* Bruxo
  + Possui habilidades de ataques mágicos.
  + Bônus em ataques mágicos, destreza e pontos de mana.
* Sacerdote
  + Possui habilidades de proteção e cura.
  + Bônus em pontos de mana, defesa física e mágica e ataques mágicos.
* Assassino
  + Possui habilidades de ataque físico e críticos.
  + Bônus em ataques físicos e tempo de espera.
* Monge
  + Possui habilidades de ataque físico e mágico mesclados.
  + Bônus em ataques físicos e mágicos.

##### Módulo Screen

Módulo responsável por gerenciar os estados do jogo. Decidiu-se utilizar telas para representar cada um dos estados possíveis do jogo: *Título*, *Configuração*, *Jogo* e *Fim*. A qualquer momento, o jogo pode solicitar a mudança de estado. Quando isso ocorre, todos os componentes que pertencem ao estado anterior são removidos, ou seja, é como se o jogo recomeçasse. Dessa forma, ao mudar de estado, nenhum componente desnecessário é mantido, consumindo tempo de processamento e memória.

A tela de título é o estado inicial do jogo. Exibe o nome do jogo e as informações iniciais. Quando o jogo se inicia, o jogo muda seu estado para a tela de configuração, onde os jogadores e as unidades são criados. Após a configuração de todas as unidades, o jogo muda seu estado para a tela de jogo, onde o jogo, de fato, acontece. Quando um jogador vence a partida, o estado do jogo é novamente alterado, dessa vez para a tela de fim, onde o resultado do jogo é exibido.

Cada uma destas telas, é uma implementação da classe *IScreen*. Esta classe é responsável por controlar os componentes que o jogo possui em sua própria tela, e eliminar os componentes de outra tela. A utilização desta arquitetura fornece bastante agilidade ao jogo, pois não é necessário se preocupar com a reciclagem de componentes mortos, durante a execução do jogo.

  
Generalização de telas

##### Módulo Debug

Responsável por exibir um indicador de performance e auxílio à detecção de toques.

A cada toque que acontece na mesa, uma animação de curta duração é executada, de forma a verificar a qualidade da calibração do software *TouchLib*. Em uma boa calibração, a animação acontece exatamente sob o dedo fez o toque.

O indicador de performance é exibido na parte inferior da tela. Este indicador calcula a taxa de *frames* por segundo (*FPS*). Um valor aceitável deve possuir média maior que 35, uma vez que abaixo deste valor, o jogo apresenta travamentos.

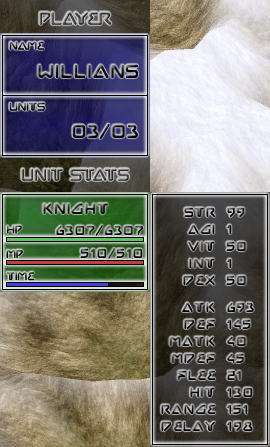
##### Módulo Menu

É o módulo que cria, desenha e gerencia todos os menus do jogo. Cada jogador possui um menu, enquanto cada unidade possui dois menus. O do jogador exibe seu nome e a quantidade de unidades vivas que ele possui. O primeiro menu da unidade é responsável por exibir seu estado atual, enquanto o segundo se encarrega de exibir as ações que podem ser executadas.

O menu que exibe o estado da unidade é dividido em duas partes. A primeira exibe as informações mais importantes de maneira mais detalhada que os indicadores. Exibe o nome da unidade e a quantidade de pontos de vida, mana e do tempo de espera após a execução de uma ação; no formato de barras e números. Dessa forma, uma medição menos precisa pode ser obtida através dos indicadores, enquanto uma mais precisa, através deste menu.

A cor deste menu se altera conforme o estado dos pontos de vida da unidade. Quando ela possui mais de 50% dos pontos de vida totais, este menu é verde, indicando que tudo está bem. Quando o total de pontos está entre 10% e 50%; a cor se altera para amarelo, indicando atenção. Quando o total de pontos está abaixo dos 10%, ele se altera para vermelho, indicando que a unidade está prestes a morrer.

Ao tocar sobre este menu, a segunda parte é expandida. Esta exibe os atributos fixos e calculados da unidade atual. Em conjunto com a primeira parte, permite que o jogador tenha um “raio-x” sobre a unidade, tendo acesso às principais informações que a unidade possui.

  
Menu do jogador e da unidade

O menu responsável por exibir as possíveis ações de uma unidade é controlado por um gerenciador, por ser mais complexo e possuir a necessidade de se comunicar diretamente com o módulo *Action*, responsável por controlar a execução das ações. Este menu apenas está ativo quando o tempo total de espera da unidade transcorreu completamente.

É composto por uma lista de itens que possuem uma lista de subitens, representados por instâncias da classe *ActionMenu* e *CommandMenu*, respectivamente. O menu possui quatro itens: *Mover*, *Atacar*, *Habilidades* e *Itens*. Os subitens estão ligados às ações da unidade que possui o menu, permitindo sua construção de forma dinâmica.

Durante o projeto do menu, decidiu-se que os subitens seriam especializações dos itens do menu. Isso facilitaria a codificação, uma vez que ambos devem ser capazes de avisar ao restante do módulo quando foram tocados, através de eventos.

Durante a construção do menu, os ataques, habilidades e itens da unidade são lidos, e com base neles, os subitens são criados. Cada subitem exibe ao lado do nome da ação, um parâmetro, que representa, no caso das habilidades, a quantidade de mana gasta na execução, ou, no caso dos itens, a quantidade de itens que a unidade possui. Ataques, por serem ilimitados, são representados com o número zero.

Cada subitem possui uma propriedade que verifica automaticamente se a unidade pode executar a ação relacionada ao menu. Caso ela não possa ser executada, o submenu não responderá aos toques do usuário. O item *Mover* não possui subitens, já que não existem tipos de movimentação diferentes.

  
Itens e seus respectivos subitens

Ações que têm como alvo a própria unidade, não necessitam de informações extras para serem executadas. Porém, nem todas as ações têm como alvo o próprio jogador. Dois exemplos são: o item *Mover* e os subitens do item *Atacar*. O primeiro necessita que o jogo determine uma área pela qual a unidade pode se mover e movimente-a; enquanto os segundos necessitam de um alvo ou uma posição para desferir o ataque.

Com o intuito de auxiliar nas obtenções dessas informações, o módulo *Interaction* foi desenvolvido. Apesar de ser explicado em detalhes mais a frente, uma introdução se faz necessária para compreender a lógica de execução do menu. Este módulo é dividido em duas partes: a primeira é responsável por auxiliar na movimentação de uma unidade enquanto a segunda auxilia a na obtenção de alvos e posições.

Quando os subitens são tocados, verifica-se se este necessita de informações extras. Caso afirmativo, o respectivo item do módulo *Interaction* é iniciado e o subitem aguarda a obtenção das informações através de uma mensagem via evento. Quando a mensagem é recebida, as informações são passadas ao subitem e a ação pode enfim, ser executada.

{imagem::diagrama execução menu}

##### Módulo Interaction

É o módulo responsável por auxiliar o menu na execução de ações. Divide-se em dois submódulos: *Mover* e *Aim*, ambos baseados em uma máquina de estados. Se comunicam com o menu através de mensagens passadas via eventos.

O estado inicial dos submódulos aguardam a ativação por parte do menu para mudar de estado. Assim que um item ou subitem é tocado no menu, o menu ativa o respectivo submódulo e este fica aguardando a interação do jogador. Caso o jogador não interaja com o submódulo, ele é desativado pelo menu, retornando ao seu estado inicial.

###### Submódulo Mover

O movimento de uma unidade pelo jogador é realizado arrastando-a para a posição desejada, dentro do limite imposto pelo jogo. Após arrastar a unidade, o jogador deve escolher para qual direção a unidade deve olhar.

Ao ativar este submódulo, uma instância do submódulo *Area* do módulo *Drawable* é desenhada, com tamanho baseado no atributo calculado: *Alcance de Movimento*. A esfera desenhada indica a área pela qual a unidade pode se mover. Caso o jogador tente arrastar a unidade para fora desta área, a unidade permanecerá na última posição válida.

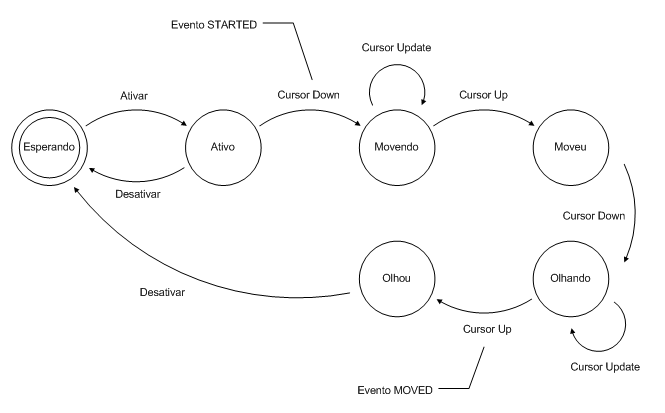
{imagem::unidade movendo}

Ao tocar sobre a unidade, o evento *CursorDown* é disparado pelo módulo *Input*, servindo de gatilho para a mudança de estado. Ao mudar de estado, o identificador relacionado ao dedo é armazenado e o evento *Started* é disparado; avisando o menu que a unidade está em movimento, portanto, não é possível mais mudar a escolha.

Ao arrastar a unidade, o evento *CursorUpdate* é disparado repetidas vezes. Verifica-se se o identificador do dedo que está se movimentando é igual ao armazenado, para evitar que outros dedos se movendo sobre a mesa no mesmo momento influenciem na movimentação da unidade. Se forem iguais, verifica-se se a posição do dedo está dentro da área imposta e em caso afirmativo, a posição da unidade é atualizada. O estado atual do submódulo não muda, pois a unidade só deixa de se mover quando o dedo é retirado da mesa.

Ao retirar o dedo da mesa, o evento *CursorUp* é disparado. O identificador novamente é comparado para evitar que outros dedos que estejam sendo retirados, influenciem no movimento da unidade. Se o identificador coincidir, a unidade encerra o seu movimento e inicia-se a orientação da unidade.

A orientação é feita de maneira análoga à movimentação da unidade, repetindo o ciclo de eventos *CursorDown*, *CursorUpdate* e *CursorUp* anteriormente explicados. Quando o jogador retira o dedo da mesa, terminando de orientar a unidade, o evento *Moved* é disparado avisando o menu que a unidade terminou de se mover. Após a finalização do movimento da unidade, o submódulo é desativado, retornando-o ao seu estado inicial.

  
Máquina de estados do submódulo Mover

###### Submódulo Aim

A obtenção de um alvo para a execução de um determinado ataque, habilidade ou item é realizado com o auxílio de uma mira. O jogador arrasta a mira para a posição onde a ação deve ser executada, dentro do limite imposto pelo jogo. Com a posição definida, a ação é executada.

Assim como o submódulo *Mover*, o módulo *Aim* também faz uso de uma área para determinar o limite em que a mira pode se mover. Os atributos calculados usados para a definição do tamanho são: *Alcance de Ataque* e *Alcance de Habilidade*, usados para ataques e habilidades, respectivamente. Caso o jogador tente mover a mira para fora desta área, a mira mantém a última sua posição válida.

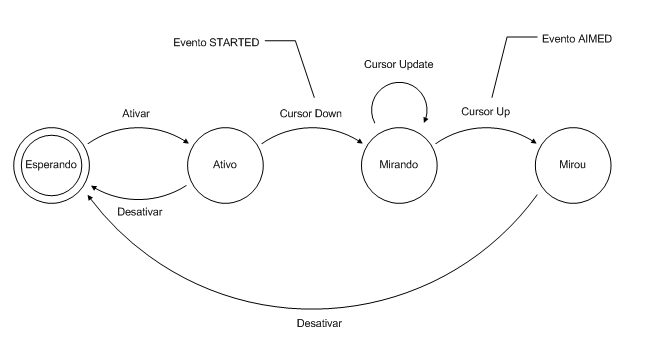
{imagem::mira}

A ordem de execução dos eventos *CursorDown*, *CursorUpdate* e *CursorUp* é muito semelhante à utilizada no submódulo *Mover*, não necessitando de maiores aprofundamentos. De maneira análoga, quando o jogador começa a mover a mira, o evento *Started* é disparado, avisando o menu que o jogador iniciou a execução da ação e esta não pode mais ser cancelada ou trocada.

Durante a movimentação da mira, é verificado, cerca de quatro vezes por segundo, se a mira está posicionada sobre alguma unidade. Se sim, e a unidade for uma unidade aliada, a mira muda para a cor verde. Se a unidade for inimiga, a cor adotada foi a vermelha. Quando a mira está sobre nenhuma unidade, ou seja, não está mirando em ninguém, sua cor é cinza.

Toda unidade que está sendo mirada é armazenada, de forma a evitar um processamento desnecessário para descobrir qual unidade está na última posição válida de mira. Quando a mira não está mirando sobre nenhuma unidade, a unidade armazenada é nula.

Quando a mira é liberada pelo jogador, retirando o dedo da mesa, o evento *Aimed* é disparado, informando o menu, sobre qual unidade que a mira se encontra e sua última posição válida. De posse destas informações juntamente com a unidade invocadora, é possível executar qualquer ação.

Máquina de estados do submódulo Aim

##### Módulo Action

blablabla... wiskas sachê... blablabla...

# Referencias

## Bibliográfica

## Sites

## Imagens

## Gráficos