



Universidade de Brasília

**Instituto de Ciências Exatas
Departamento de Ciência da Computação**

**Deskworld: Software Simulador de Física 2D para
Mesas com Superfície Multitoque**

Danilo Gaby Andersen Trindade
Victor Sampaio Zucca

Brasília
2011



Universidade de Brasília

**Instituto de Ciências Exatas
Departamento de Ciência da Computação**

Deskworld: Software Simulador de Física 2D para Mesas com Superfície Multitoque

Danilo Gaby Andersen Trindade
Victor Sampaio Zucca

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.^a Dr.^a Carla Denise Castanho

Coorientador
Prof. Dr. Marcus Vinicius Lamar

Brasília
2011

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Marcus Vinicius Lamar

Banca examinadora composta por:

Prof.^a Dr.^a Carla Denise Castanho (Orientadora) — CIC/UnB
Prof. Dr. Marcus Vinicius Lamar (Coorientador) — CIC/UnB
Prof. Dr. Pedro de Azevedo Berger — CIC/UnB
Prof. Dr. Ricardo Pezzuol Jacobi — CIC/UnB

CIP — Catalogação Internacional na Publicação

Trindade, Danilo Gaby Andersen.

Deskworld: Software Simulador de Física 2D para Mesas com Superfície Multitoque / Danilo Gaby Andersen Trindade, Victor Sampaio Zucca. Brasília : UnB, 2011.
66 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2011.

1. software interativo, 2. simulador de física, 3. superfície multi-toque,
4. mesa multi-toque, 5. FTIR

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília-DF — Brasil



Universidade de Brasília

**Instituto de Ciências Exatas
Departamento de Ciência da Computação**

Deskworld: Software Simulador de Física 2D para Mesas com Superfície Multitoque

Danilo Gaby Andersen Trindade
Victor Sampaio Zucca

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.^a Dr.^a Carla Denise Castanho (Orientadora)
CIC/UnB

Prof. Dr. Marcus Vinicius Lamar (Coorientador)
CIC/UnB

Prof. Dr. Pedro de Azevedo Berger Prof. Dr. Ricardo Pezzuol Jacobi
CIC/UnB CIC/UnB

Prof. Dr. Marcus Vinicius Lamar
Coordenador do Bacharelado em Ciéncia da Computaçâo

Brasília, 10 de fevereiro de 2011

Agradecimentos

Agradecemos, primeiramente, aos membros da banca, professores Carla Castanho, Marcus Lamar, Pedro Berger e Ricardo Jacobi, pela presença e pelas recomendações propostas durante a defesa deste trabalho, em particular aos nossos orientadores Carla Castanho e Marcus Lamar que nos auxiliaram nesta empreitada, orientando-nos em uma área de constante crescimento em nosso departamento.

Agradecemos também aos colegas Pedro Brandão e Saulo Camarotti pelo seu trabalho que inspirou este, e também pelo auxílio prestado no desenvolvimento.

Agradecemos ao nosso amigo Luciano Santos, que está desenvolvendo um outro projeto que utiliza a mesma mesa construída neste trabalho e nos ajudou imensamente na construção dela, e em especial ao seu tio, Antônio Carlos Santos, que construiu a estrutura de alumínio da mesa e nos forneceu ela gratuitamente viabilizando a realização deste projeto.

Agradecemos especialmente aos nossos amigos e familiares, que sempre nos apoiaram e nos ajudaram a seguir em frente.

Danilo e Victor

Em especial, quero agradecer ao meu pai, Roberto Zucca, por ter ajudado a solucionar problemas encontrados na montagem e ajustes da mesa com suas idéias criativas.

Victor

Queria agradecer ao meu amigo, Marco Gallego, que desenhou para nós os menus e botões do jogo, e em especial a minha avó Elna Trindade, que faleceu neste último natal. Ela sempre me incentivou e me inspirou, nunca deixando que sua idade fosse limite para lhe impedir de se modernizar. Tenho certeza que ela estaria na frente da fila para usar o nosso projeto assim que concluído, devido a sua grande paixão pela tecnologia.

Danilo

Resumo

Com a evolução tecnológica, são pesquisadas novas formas de interação com os dispositivos computacionais, de modo a facilitar e tornar mais amigável e intuitiva a interação do ser humano com a máquina. Atualmente, estão sendo desenvolvidos dispositivos de *hardware* que conseguem processar movimentos, gestos e toques. Para acompanhar esse novo paradigma de utilização dos dispositivos, os *softwares* devem tratar questões não tradicionais, como por exemplo, a interpretação de gestos que o usuário faz ou até sua a utilização por um número indeterminado de usuários simultaneamente.

Neste contexto, o objetivo deste trabalho é o desenvolvimento de um *software* simulador de física 2D, denominado *Deskworld*, para uma mesa com superfície multi-toque. Observa-se que as características inerentes a um *software* simulador de física se encaixam perfeitamente ao método de interface escolhido, uma vez que o desenho de figuras geométricas e de forma livre constituem grande parte dos inputs realizados pelo usuário.

Além da concepção do *software* interativo também é foco deste trabalho o projeto e montagem da mesa multi-toque que servirá como plataforma de execução para o *Deskworld*. A construção da mesa utilizou uma estrutura de alumínio com uma lâmina de acrílico na parte superior, integrada a um conjunto de *LEDs* infravermelhos, um projetor multimídia e uma *webcam*. A abordagem de iluminação utilizada para detecção dos toques foi *FTIR* (*Frustrated Total Internal Refraction*).

Palavras-chave: software interativo, simulador de física, superfície multi-toque, mesa multi-toque, FTIR

Abstract

With the technological evolution, new forms of interaction with computational devices have been researched as an attempt to make the human-computer interaction easier and friendlier. Nowadays, hardware devices that are capable of processing user's movement, gesture and touches, have been developed. In order to follow this new paradigm, the software has to deal with non-traditional issues, like gesture interpretation or even an unknown number of simultaneous users.

In this context, the goal of this work is to develop a 2D physics sandbox software, called Deskworld, for a multi-touch table. It is noted that the characteristics inherent to a physics sandbox software fit perfectly the interface method chosen. That is because geometrical and free form drawing are most of the users' inputs.

Besides the creation of the interactive software, this work also aims the construction of the multi-touch table which is the demonstration platform of the Deskworld. The table was made of an aluminum structure with an acrylic sheet on top, integrated with infrared LEDs, besides a multimedia projector and a webcam. The illumination approach for touch detection is FTIR (Frustrated Total Internal Refraction).

Keywords: interactive software, physics sandbox, multi-touch surface, multi-touch table, FTIR

Sumário

1	Introdução	1
2	Trabalhos Correlatos	8
2.1	Mesas com superfície multi-toque	8
2.2	<i>Softwares</i> interativos para mesas com superfície multitoque	10
2.3	<i>Softwares</i> simuladores de física	13
3	Fundamentação Teórica	17
3.1	Processamento de Toques	17
3.1.1	Métodos de Iluminação de Mesas Multi-Toque	18
3.1.2	Processamento das Imagens	19
3.1.3	Detecção de <i>Blobs</i>	21
3.1.4	Correção da câmera	22
3.2	Aspectos de mesas com superfície multitoque	25
3.2.1	Tamanho e Resolução	25
3.2.2	Interfaces	26
3.2.3	Número de Usuários	27
3.2.4	Detecção de Gestos	27
4	Construção da Mesa Multi-toque	30
4.1	Estrutura da Mesa	30
4.2	Superfície	30

4.3	Iluminação Infravermelha	31
4.4	Câmera	32
4.5	Projetor	33
4.6	Filtros de Processamento de Vídeo	34
5	<i>Software Deskworld</i>	38
5.1	<i>Design</i> do <i>Deskworld</i>	38
5.1.1	Requisitos de interface	39
5.1.2	Conceito	39
5.1.3	Regras e Objetos	39
5.1.4	Interface	41
5.2	Ferramentas de suporte	41
5.2.1	<i>Game Engine</i>	41
5.2.2	<i>Bibliotecas de apoio</i>	42
5.3	Arquitetura e Detalhes de Implementação	44
5.3.1	Arquitetura do <i>Deskworld</i>	45
5.3.2	Detalhes de implementação	45
6	Conclusão	49
	Referências	52

Lista de Figuras

1.1	Jogo para <i>EDSAC OXO</i> [46]	1
1.2	Jogo <i>Tennis for two</i> [46]	2
1.3	Jogo <i>Spacewar</i> [46]	2
1.4	Console <i>Odyssey</i> da <i>Magnavox</i> [16]	3
1.5	Console <i>Atari 2600</i> da <i>Atari</i> [47]	3
1.6	Console <i>NES</i> da <i>Nintendo</i> [10]	3
1.7	Consoles de última geração. Da esquerda pra direita: <i>Wii</i> , <i>Playstation 3</i> e <i>Xbox 360</i> [21]	5
1.8	Controle para <i>Nintendo Wii Wiimote</i> [19]	5
1.9	Controle para <i>Playstation 3 Playstation Move</i> [19]	5
1.10	Câmera para <i>Xbox 360 Kinect</i> [19]	5
2.1	<i>Reactable</i> - Mesa com marcadores fiduciais [40]	9
2.2	<i>Microsoft Surface</i> [31]	9
2.3	Jogo para mesas com superfície multitoque <i>IRTaktiks</i> [43]	10
2.4	<i>The Laundry Game</i> - Jogo para mesas com superfície multitoque [23]	11
2.5	<i>Game of Life - Software</i> para mesas com superfície multitoque [45]	11
2.6	<i>Ecodefense</i> - Jogo para mesas com superfície multitoque [5]	12
2.7	<i>Station Defender</i> - Jogo para mesas com superfície multitoque [17]	12
2.8	<i>Little Big Planet</i> - Jogo para <i>PS3</i> [32]	13
2.9	<i>Modnation Racer</i> - Jogo para <i>PS3</i> [32]	14
2.10	<i>Phun</i> - Jogo para PC [1]	15

2.11	<i>Crayon Physics</i> - Jogo para PC [25]	15
2.12	Seis fases adicionais da adaptação do <i>Numpy Physics</i> para mesas multi-toque. [38]	16
3.1	Projeto de mesa multitoque com DI [6]	18
3.2	Esquema de detecção de toque FTIR (adaptada de [6])	19
3.3	Projeto da mesa <i>Virttable</i> [29]	20
3.4	Imagens capturadas com métodos de iluminação diferentes [29].	20
3.5	Diagrama do processo de detecção dos toques.	21
3.6	Gráfico de P [5].	22
3.7	Gráfico de Q [5].	23
3.8	Correção da distorção focal da câmera [35].	23
3.9	Calibração de pontos na <i>CCV</i> [41].	24
3.10	O tradicional botão <i>OK</i> visto por dois ângulos distintos [5].	26
3.11	Gestos de <i>Click</i> e <i>Drag</i> [50].	27
3.12	Reconhecimento de gesto no jogo <i>Black and White</i> [4].	28
3.13	Gestos de <i>Rotate</i> e <i>Scale</i> [50].	29
4.1	Projeto da mesa com superfície multi-toque construída neste trabalho.	31
4.2	Visão lateral do projeto da mesa com superfície multi-toque construída neste trabalho.	32
4.3	Fotos da mesa multi-toque construída neste projeto.	33
4.4	Visão interna da mesa, ilustrando seus componentes.	34
4.5	Câmera <i>Playstation Eye</i> no suporte.	34
4.6	Furos efetuados na borda do acrílico.	35
4.7	<i>LEDs</i> soldados na borda do acrílico.	35
4.8	Esquema de montagem do circuito dos <i>LEDs</i> infravermelhos.	36
4.9	Projetor multimídia <i>BenQ MP772ST</i> [9]	37
4.10	Aplicação dos filtros à imagem capturada pela câmera do sistema.	37

5.1	<i>Deskworld</i> sendo utilizado na mesa.	38
5.2	Tela do <i>software Deskworld</i> .	40
5.3	Aproximação de formas a polígonos convexos.	40
5.4	Aplicativo de captura de toques <i>Community Core Vision</i> [8]	44
5.5	Diagrama de classes do <i>Deskworld</i>	48

Lista de Tabelas

5.1 Parâmetros de mensagens TUIO em superfícies interativas 2D	43
--	----

Capítulo 1

Introdução

A indústria do entretenimento é uma das que mais cresce, sendo que o Brasil lidera esta lista juntamente com a China [44]. A indústria dos *video games* já está estabelecida como a mais lucrativa no setor de entretenimento mundial, começou por volta dos anos 50 com alguns programas demonstrativos para computadores [46]. Foram criados pequenos jogos que utilizavam os inputs limitados disponíveis, como o jogo-da-velha eletrônico, chamado *OXO* (Figura 1.1), desenvolvido por *Alexander S. Douglas* em 1952, e o *Tennis for two* (Figura 1.2), criado pelo *Brookhaven National Laboratory* (do Departamento Americano de Energia) para o dia anual de visitação pública em 1958.

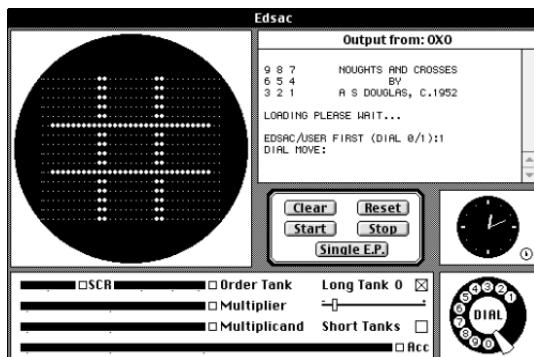


Figura 1.1: Jogo para *EDSAC OXO* [46]

Já em 1961, *Steve Russell*, juntamente com seus colegas do *MIT (Massachusetts Institute of Technology)*, desenvolveu um jogo chamado *Spacewar* (Figura 1.3) [39]. Inicialmente criado para o computador *DEC PDP-1*, consiste num jogo para dois jogadores, cada um no controle de uma nave, em que o objetivo é atirar mísseis para destruir a nave do oponente. O *Spacewar* era distribuído juntamente com o *PDP-1* como um programa de teste. Obteve tanta popularidade que em 1962 foi expandido e adaptado para outros sistemas populares na época. Tornou-se de domínio público e chegou a outras universidades pela *ARPAnet (Advanced Research Projects Agency Network)*. Isso fez com que o *Spacewar* fosse o primeiro jogo de computador altamente difundido.

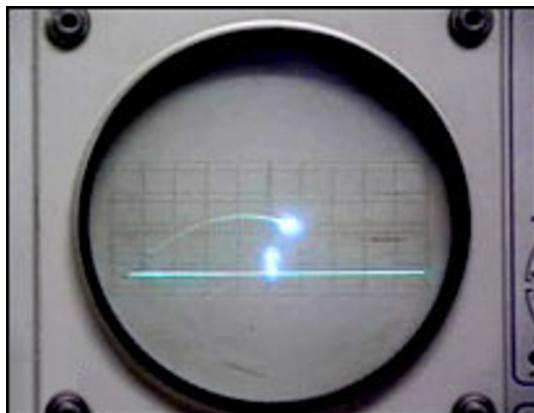


Figura 1.2: Jogo *Tennis for two* [46]

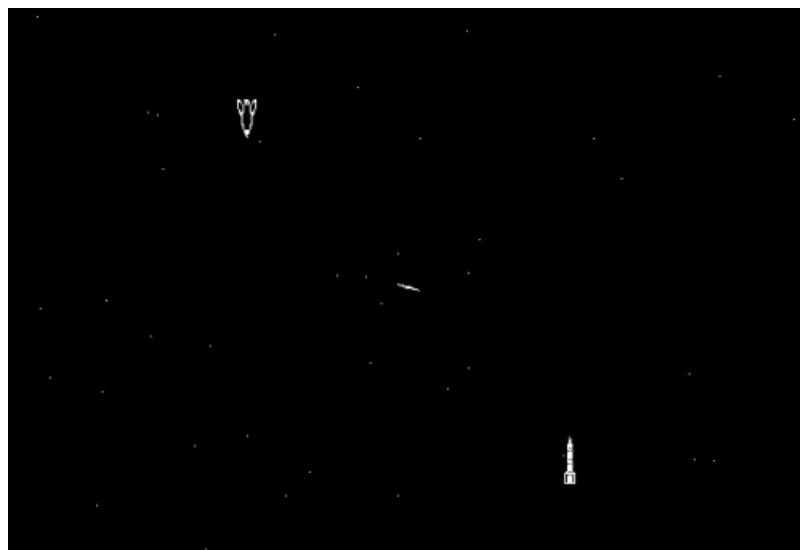


Figura 1.3: Jogo *Spacewar* [46]

O primeiro console doméstico foi o *Magnavox Odyssey* [16], que pode ser observado na Figura 1.4, lançado em 1972. O *Magnavox Odyssey* foi inovador no sentido que propôs a criação de um aparelho computacional sem *display* para ser conectado a televisores. Até aquele momento os computadores eram produzidos com *display* embutido. Além disso, ele também inaugurou outro conceito que é o de mídia removível para os consoles, cuja evolução migrou para cartuchos, *CDs*, até os atuais *DVDs* e *Blu-Rays*. Entretanto, o preço relativamente alto para a época e uma estratégia de marketing ineficiente contribuiram para que o *Odyssey* não atingisse a popularidade esperada. Neste cenário, em 1976, a concorrente *Atari* entrou no mercado lançando, com sucesso, o primeiro console doméstico que permitia a troca de jogos, o *Atari 2600* (Figura 1.5) [47], que utilizou o processador de baixo custo *MOS 6502*. Na verdade, a inserção da *Atari* no mercado de jogos iniciou com o jogo *Pong* - um simples jogo de tênis - cuja primeira versão foi criada em 1972 para uma máquina com *display* que permitia ao jogador sua utilização através da inserção de moedas. Em 1973 ele foi adaptado para uma versão doméstica e em 1976 foi distribuído

juntamente com o console da *Atari 2600*.



Figura 1.4: Console *Odyssey* da *Magnavox* [16]



Figura 1.5: Console *Atari 2600* da *Atari* [47]

Após esse período, os *video games* passaram por um declínio, e somente por volta de 1980 foi que a *Nintendo* resgatou esse mercado com lançamento do jogo *Donkey Kong*, inicialmente desenvolvido em conjunto com a *Atari*. Devido a divergências sobre os direitos autorais do jogo, a parceria foi desfeita [47]. Em 1985 a *Nintendo* lançou o console *NES* (Figura 1.6) e passou a dominar o mercado. A competição da concorrente *SEGA*, que produzia o console *Master System*, não era significativa a ponto de abalar a soberania da *Nintendo*.



Figura 1.6: Console *NES* da *Nintendo* [10]

A partir desse período, a indústria de entretenimento eletrônico teve um crescimento significativo, de modo que a evolução das gerações de consoles fomentaram o surgimento

de competidores e vice-versa. A *Nintendo* se manteve forte e tornou-se uma das três companhias que dominam esse mercado hoje em dia. Sua principal competidora, a *SEGA*, não conseguiu acompanhar o sucesso, fracassando nas vendas do seu último console, o *Dreamcast*. Atualmente a *SEGA* ainda é uma empresa grande, entretanto atua somente na produção de jogos para outros consoles.

Durante esta fase de ascensão dos jogos eletrônicos, as empresas *Nintendo* e *Sony* uniram esforços na tentativa de produção de um novo sistema que diminuísse custos e aumentasse a capacidade de armazenamento através da substituição dos cartuchos por *CDs*, como dito por *Rabin et al.* em [39]. Apesar do fracasso na parceria, a *Sony* insistiu na idéia e lançou, em 1994, o console *Playstation* que foi um sucesso de vendas. Considerado uma revolução no seu tempo, o *Playstation 2* foi lançado um ano antes dos consoles da sua geração, o que lhe rendeu o maior índice de vendas de um console de video game de todos os tempos, ultrapassando a marca de 140 milhões de unidades [49]. Atualmente a *Sony* compete no mercado como uma das maiores empresas de diversão eletrônica.

Outra empresa decidida a explorar o mundo dos consoles para video games foi a *Microsoft*. Gigante no mundo da computação, viu a oportunidade de aplicar seus conhecimentos para criar um sistema eficiente e inovador. Em 2001, lançou seu primeiro console, o *Xbox*. Embora considerado da mesma geração do *Playstation 2*, o *Xbox*, foi lançado relativamente tarde, se comparado ao concorrente, afetando consideravelmente suas vendas. Com arquitetura baseada em um *PC* tradicional, ocasionou perdas iniciais significativas para a empresa. Entretanto, a *Microsoft* persistiu nesse mercado e atualmente é competidora direta das gigantes *Nintendo* e *Sony*.

Todos os consoles acima mencionados utilizam basicamente o mesmo sistema de *input* de seus predecessores, ou seja, a utilização do *gamepads*, alguns exemplos que podem ser observados na Figura 1.7 junto de seus *consoles* respectivos. Alguns ainda possuem outros controles, como por exemplo pistolas que podem ser apontadas para a tela do jogo. Contudo, o custo elevado e o número reduzido de jogos compatíveis colaboraram para a pouca difusão destes dispositivos especiais de entrada de dados.

Atualmente, o mercado dos jogos eletrônicos conta com três consoles competindo entre si, o *Nintendo Wii* da *Nintendo*, o *Xbox 360* da *Microsoft* e o *Playstation 3* da *Sony*, que podem ser observados na Figura 1.7. Além de estarem disputando espaço com os jogos para computadores pessoais (*PCs*), já amplamente difundidos, os *consoles* ainda enfrentam outros concorrentes, como aparelhos celulares com tela sensível ao toque, cada vez mais consolidados como plataforma para o desenvolvimento de jogos.

Durante a evolução natural da interação entre o homem e a máquina, tem se buscado cada vez mais a adaptação das nossas características fisiológicas com o objetivo de tornar a experiência mais agradável e mais produtiva quanto ao uso de computadores. Nesse contexto, a utilização de comandos gestuais como meio de *input* tem sido foco de diversas pesquisas, como pode ser observado, por exemplo, em [28], [48], [18] e [36]. Do mesmo modo, as grandes companhias do setor de entretenimento digital, tem conseguido avanços significativos no que diz respeito a utilização de gestos como meio de interação com o sistema computacional. O *Nintendo Wii* possui controles sensíveis ao movimento, permitindo o uso de gestos para o controle em seus jogos; o *Wiimote* (Figura 1.8), assim como



Figura 1.7: Consoles de última geração. Da esquerda pra direita: *Wii*, *Playstation 3* e *Xbox 360* [21]

o *Playstation 3* é dotado de controles sensíveis ao movimento e com noção de profundidade; o *Playstation Move* (Figura 1.9), e o *Xbox 360* apresenta o *Kinect* (Figura 1.10), uma câmera inteligente capaz de identificar gestos.



Figura 1.8: Controle para *Nintendo Wii Wiimote* [19]



Figura 1.9: Controle para *Playstation 3 Playstation Move* [19]



Figura 1.10: Câmera para *Xbox 360 Kinect* [19]

Como mencionado acima, os principais consoles disputam o mercado de softwares de entretenimento com outras plataformas, tais como *PCs* e celulares. Uma plataforma ainda pouco explorada nesse contexto são as mesas com superfícies multi-toque, como a *Reactable* [40] e a *Microsoft Surface* [31]. Devido ao alto custo apresentado pelas opções comerciais, a difusão das mesas multi-toque no mercado ainda é lenta. No entanto, esta interface, no formato de mesa que viabiliza a interação através de múltiplos toques sobre sua superfície, apresenta um grande potencial, principalmente para aplicações relacionadas ao entretenimento digital. Alternativas de baixo custo para construção caseira de mesas multi-toque também tem sido foco de experimentos, como em [29], [5] e [43], por exemplo.

Apesar das inúmeras possibilidades de interação que as mesas multi-toque apresentam, existem poucos softwares disponíveis para este tipo de plataforma. Dentre os existentes,

pode-se mencionar o *Tuio Smoke* [24], um aplicativo que desenha uma fumaça saindo do local onde é feito toques, *Community Earth* [2], um aplicativo com um mapa global naveável através do toque, e o *Numpty Physics* adaptado para mesas multi-toque [38], que será explicado mais adiante. Embora sejam *softwares* que demonstram algumas funcionalidades desta interface, nenhum deles demonstra inteiramente as capacidades de *input* da mesa, seja ignorando a facilidade de interação de vários usuários em um único dispositivo ou deixando de aproveitar gestos de fácil utilização.

Nesse contexto, este trabalho tem como objetivo o desenvolvimento de um *software* simulador de física bidimensional, denominado *Deskworld*, para uma mesa com superfície multi-toque. Observa-se que as características inerentes a um *software* simulador de física se encaixam perfeitamente ao método de interface escolhido, uma vez que o desenho de figuras geométricas e de forma livre, constituem grande parte dos *inputs* realizados pelo usuário. Além da concepção do *software* interativo também é foco deste trabalho o projeto e montagem da mesa multi-toque que servirá de plataforma de execução para o *Deskworld*, detalhada no Capítulo 4.

Os *softwares* simuladores de física são, essencialmente, programas que começam com um mundo em aberto, onde o usuário pode desenhar os objetos e atribuir a eles as propriedades, tais como massa, fricção ou coeficiente de restituição de força. Estes objetos interagem, então, de acordo com as leis da física, obedecendo gravidade, aceleração e empuxo. Este estilo de *software* interativo possibilita a implementação do reconhecimento de gestos sobre a superfície da mesa, tais como escalar, rotacionar, arrastar, além do desenho de formas padrões como um quadrado ou círculo. O desenvolvimento de *softwares* simuladores de física tem se concentrado basicamente na plataforma *PC*, onde *Phun* [1], *Crayon Physics* [25] e *Numpty Physics* [11] são as implementações mais populares. Apesar do *Numpty Physics* possuir uma adaptação para receber *input* através de uma mesa multi-toque, seus recursos são limitados pois é um jogo com um objetivo definido, e uma vez completo este objetivo, o jogo acaba, não permitindo a criação livre. Além disso, não suporta um número indeterminado de usuários o utilizando, o que é fundamental neste tipo de implementação.

Além das características e regras tradicionais dos simuladores de física, o *software* *Deskworld*, desenvolvido neste trabalho e detalhado no Capítulo 5, possui alguns aspectos inovadores. O mundo interativo pode ser dividido, em tempo de execução, pelo usuário de modo que cada parte se torna um mundo independente dos demais, permitindo que cada mundo utilize uma ferramenta diferente ao mesmo tempo, facilitando a utilização de qualquer número de usuários simultaneamente. Além disso, o *Deskworld* identifica o formato de desenhos de forma a aproximar as linhas traçadas a retas, facilitando a criação de polígonos e melhorando, assim, a interação dos usuários com o *software*. Por isso, ele é adequado para demonstrar as capacidades desse tipo de interface.

O restante deste trabalho está dividido conforme segue. O segundo capítulo apresenta os trabalhos correlatos, tanto no que diz respeito a mesas multi-toque quanto a softwares de simulação de física. O terceiro capítulo é dedicado à fundamentação teórica necessária para o entendimento deste trabalho. Explica, entre outros, os princípios de processamento de imagens envolvidos na identificação dos toques sobre a mesa. O quarto capítulo relata

os detalhes da mesa construída neste trabalho, tais como, o projeto, decisões de hardware, iluminação, etc. O quinto capítulo foca na desenvolvimento do software *Deskworld*, apresentando tanto os aspectos de *design* e concepção, como arquitetura, e implementação. Por fim, o capítulo seis elenca algumas considerações finais e trabalhos futuros.

Capítulo 2

Trabalhos Correlatos

Neste capítulo, primeiramente são expostos os principais trabalhos relativos a implementações de mesas com superfícies multi-toque. Em seguida, no intuito de demonstrar o potencial deste tipo de interface, são apresentados uma série de *softwares* interativos, incluindo jogos, desenvolvidos para mesas multi-toque. Ao final são mostrados alguns *softwares* simuladores de física em geral, para diversos sistemas, incluindo mesas com superfície multitoque.

2.1 Mesas com superfície multi-toque

As mesas com superfície multi-toque são relativamente recentes no mercado, porém seu alto custo fez com que não fossem muito proliferadas. Devido a isto, alternativas foram feitas para construção de mesas de baixo custo. Uma alternativa criativa foi proposta por [42], onde demonstra-se a viabilidade da montagem de uma pequena mesa com superfície multitoque com menos de US\$50,00. Com materiais de baixo custo como papelão, papel e vidro, construiram uma mesa utilizando um computador pessoal e uma *webcam*. Outras alternativas artesanais incluem, por exemplo, a construída para o jogo *Eco-Defense* [5], a elaborada para o jogo *IRTaktiks* [43], bem como a *Virttable* [29].

Diversas utilidades são propostas para essas mesas, não somente para toques, como também para marcadores fiduciais, que são peças com desenhos diferentes neles onde, uma vez em contato com a mesa, seu sistema de processamento de imagens consegue identificar esses padrões e associá-los a diferentes ações. Tem-se por exemplo a *Reactable* [40] (Figura 2.1), disponível desde 2008, que, apesar de não possibilitar o toque do usuário, utiliza marcadores fiduciais para habilitar a interação de seus usuários com a música, fornecendo um instrumento musical diferente. Ela é utilizada em concertos e eventos.

Outro exemplo é a *Microsoft Surface* [31] (Figura 2.2), que proporciona funcionalidades diversas, tais como a habilidade de compartilhar figuras e arquivos por meio da interação com objetos eletrônicos colocados sobre sua superfície e comandos via o toque do usuário.



Figura 2.1: *Reactable* - Mesa com marcadores fiduciais [40]



Figura 2.2: *Microsoft Surface* [31]

Essas duas mesas são as mais populares, apesar de existirem outros projetos comerciais e desenvolvidos para exibições, como a *Touch Magix* [30] ou a desenvolvida pela companhia *Ideum*, visto em [20].

A idéia presente em todas é a mesma, ou seja, utilizar um projetor multimídia para projetar a imagem na superfície da mesa, na qual o usuário pode tocar para interagir com o *software*. A superfície é iluminada com a utilização de *LEDs* (*Light-Emitting Diode*) infravermelhos e os toques são detectados por uma câmera sem filtro infravermelho que os interpreta em relação à posição na mesa. Existem várias maneiras de iluminar uma mesa. As principais são DI (*Diffuse Illumination*) e FTIR (*Frustrated Total Internal Reflection*), que serão explicadas mais detalhadamente no Capítulo 3.

2.2 *Softwares* interativos para mesas com superfície multitoque

Existem vários *softwares* produzidos para utilização em mesas com superfície multitoque. Por exemplo, o *IRTaktiks* [43] (Figura 2.3), um jogo de estratégia desenvolvido para uma mesa com iluminação *FTIR* construída durante o mesmo projeto. Este jogo foi desenvolvido para dois jogadores, onde cada um tem suas unidades e deve derrotar as unidades inimigas. Um problema, no entanto, é a identificação do jogador, pois não há verificação a qual jogador pertence os toques feitos sobre a mesa, o que permite aos jogadores, acidentalmente ou não, controlar as unidades do inimigo. Além disso, o jogo foi projetado para apenas dois jogadores, o que não tira proveito da característica multiusuário inerente às mesas multitoque.



Figura 2.3: Jogo para mesas com superfície multitoque *IRTaktiks* [43]

Por se tratarem de superfícies normalmente sem limitações de número de toques, os jogos desenvolvidos para mesas devem focar em interatividade de vários jogadores. *Softwares* como o *The Laundry Game* [23] (Figura 2.4), que é um jogo de separação de roupas em categorias, e o *Game of Life* (Figura 2.5), feito pelo *Verve Project* [45], que é o tradicional jogo da vida, onde uma bactéria vive dependendo do número de bactérias ao seu

redor. Apesar de estes jogos serem simples, divertidos, e multiusuário, a limitação das ações possíveis faz com os usuários abandonem seu uso precocemente.

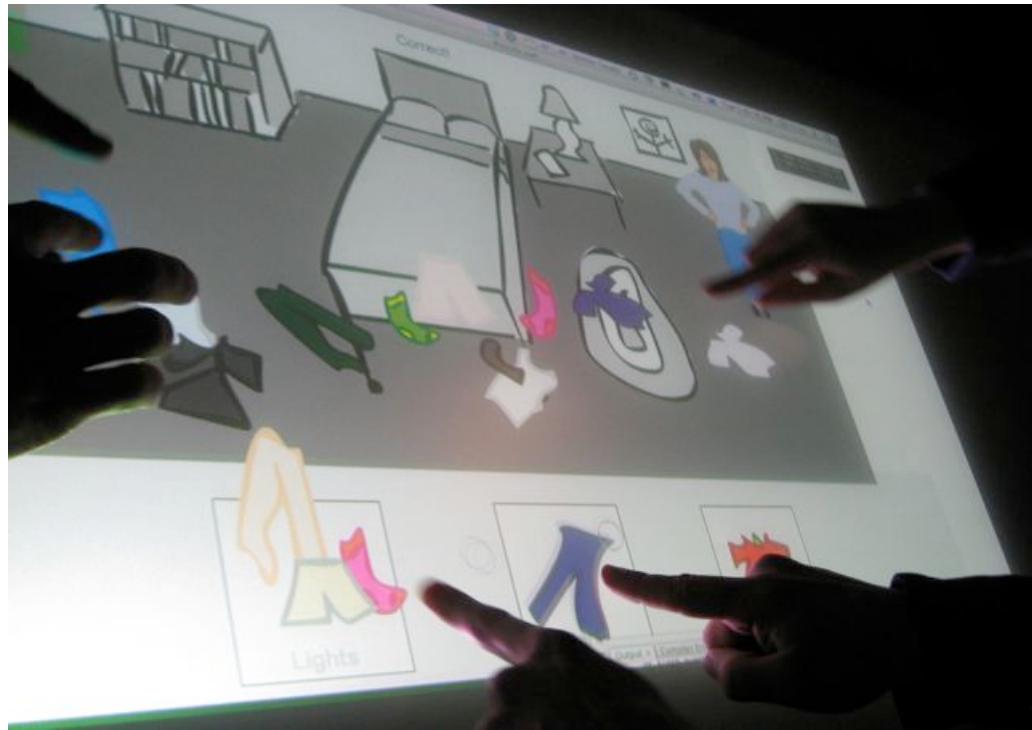


Figura 2.4: *The Laundry Game* - Jogo para mesas com superfície multitoque [23]



Figura 2.5: *Game of Life* - Software para mesas com superfície multitoque [45]

Outros *softwares* para mesas são mais completos, possuindo graus de dificuldade e sistema de pontuação. Um desses jogos é o *Eco Defense* [5] (Figura 2.6), onde o jogador deve destruir nuvens de poluição antes que elas afetem a floresta posicionada nas bordas da tela, podendo colocar torres de defesa para ajudar. O grau de dificuldade vai aumentando de acordo com o tempo de jogo, pois as nuvens ficam cada vez mais rápidas e numerosas,

até que jogadores percam. Outro exemplo é o *Station Defender* [17] (Figura 2.7), onde os jogadores devem defender sua base construindo paredes ao redor dela. Ambos jogos proporcionam melhor jogabilidade, pois há mudança no grau dificuldade e a busca pela sobrevivência pelo maior tempo possível envolve os jogadores, gerando competitividade.

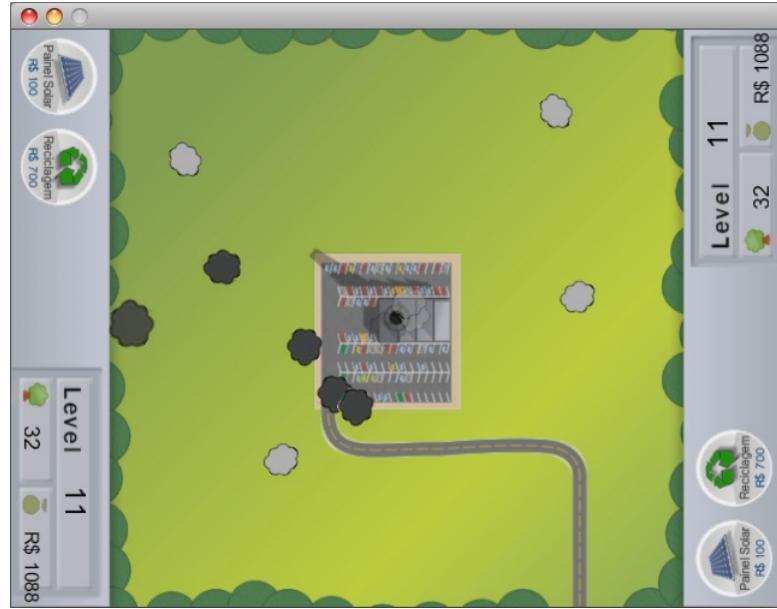


Figura 2.6: *Ecodefense* - Jogo para mesas com superfície multitoque [5]

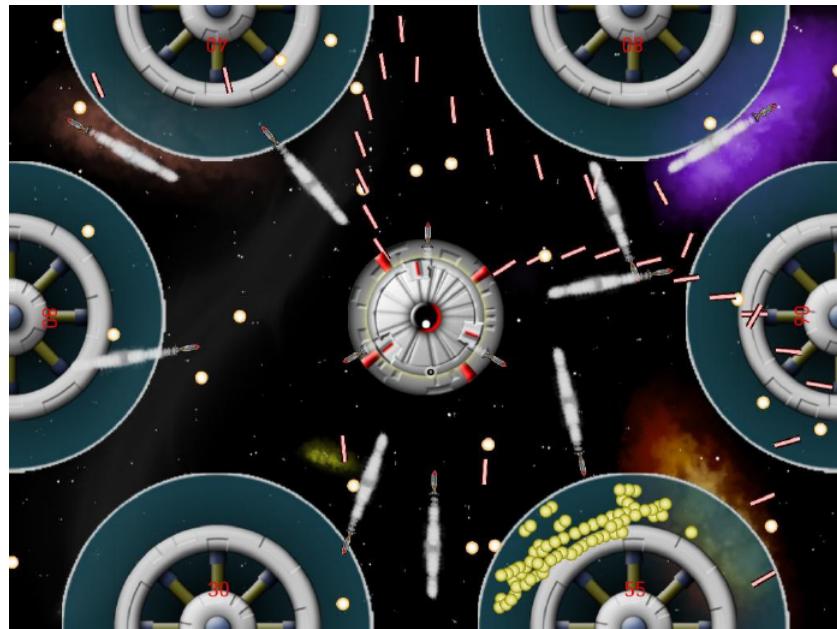


Figura 2.7: *Station Defender* - Jogo para mesas com superfície multitoque [17]

2.3 Softwares simuladores de física

Um *software* simulador de física consiste de um programa que possui um mundo aberto onde o usuário pode criar o que desejar. Estas criações recebem propriedades de um objeto tradicional e interagem com o mundo de acordo com as leis da física.

Um jogo que abriu as portas para a popularização desse tipo de *software* é o *Little Big Planet* [32] (Figura 2.8). Nesse jogo, inicialmente desenvolvido para *Playstation 3* [12] e depois para *Playstation Portable* [13], o jogador controla um boneco chamado *Sack Boy*, que pode mudar sua aparência como quiser, e deve seguir por uma fase no estilo de jogo de plataforma até chegar ao seu final. A diferença está no tipo do jogo, que os criadores chamaram de *Play, Create, Share*, onde os jogadores são livres para criar suas próprias fases como quiserem, compartilhando-as com todos os outros jogadores do mundo. Neste modo, o *Sack Boy* pode colocar objetos em jogo para construir a fase, e os objetos interagem de acordo com as leis da física, podendo possuir movimento e atrapalhar ou ajudar uns aos outros, podendo ser perigosos ou não para o *Sack Boy*. Com esta ferramenta, os jogadores construíram inúmeras fases inovadoras ao redor do mundo, inventando coisas que até os criadores nunca haviam imaginado. O sucesso foi tanto que os criadores do *Little Big Planet* lançaram recentemente a sua continuação, *Little Big Planet 2* [33], onde a criação de fases vai além de jogos de plataforma como na primeira versão, com a possibilidade do jogador criar regras no jogo que permitam a criação de vários tipos de jogos diferentes, como por exemplo, jogos do tipo *shooter* ou de estratégia.



Figura 2.8: *Little Big Planet* - Jogo para *PS3* [32]

Ainda dentro da categoria de *Play, Create, Share*, a mesma empresa criadora do *Little Big Planet* lançou o jogo *Modnation Racer* [34] (Figura 2.9), um jogo de corrida onde os jogadores podem criar as suas próprias pistas de corridas e compartilhar com todos os demais. Lançado em maio de 2010, já possui milhares de pistas criadas por usuários.

Esses jogos demonstram a popularidade do estilo onde o usuário é quem cria seu próprio jogo.



Figura 2.9: *Modnation Racer* - Jogo para PS3 [32]

No cenário de *softwares* simuladores de física 2D, existe um *software* chamado *Phun* [1] (Figura 2.10), que não possui objetivo. Ele é uma ferramenta simuladora de física 2D que permite ao usuário construir o ambiente que desejar, podendo desenhar formas geométricas tradicionais ou livres, modificar a taxa de gravidade e a fricção dos objetos, transformar objetos para forma líquida e controlar densidade dos elementos, entre outros.

Um jogo muito popular neste estilo é o *Crayon Physics* [25]. Neste jogo, é apresentado ao jogador um cenário que possui uma bola e inicialmente uma estrela. O jogador deve guiar a bola de modo a coletar todas as estrelas que lhe são apresentadas para completar a fase. Para isso, o jogador possui um giz de cera virtual e deve-se com ele modificar livremente o cenário, podendo desenhar a forma que quiser, adicionando-a ao mundo assim que ele acabar de desenhá-la. Uma vez no mundo, o objeto obedece a gravidade e adquire um momento, obtendo assim um movimento. O jogador ainda pode utilizar pontos de fixação para prender objetos juntos e correntes para fazer dispositivos improvisados como elevadores. Tudo isso é feito levando-se em conta as leis da física, como as três leis de *Newton* aplicadas a um mundo bidimensional. Este jogo foi adaptado para mesas com superfície multitoque utilizando *Action Script* por um grupo chamado *Multitouch Barcelona* [3]. Contrário a sua inspiração original, ele possui somente o modo de criação do *Crayon Physics*, não tendo objetivo e por isso não pode ser chamado de jogo.

Inspirado no *Crayon Physics*, foi desenvolvido o *Numpty Physics* [11] cujos objetivo e ferramentas são os mesmos do seu precursor, com a vantagem de possuir código aberto. Este jogo foi adaptado para funcionar em mesas com superfícies multitoque, a partir de um trabalho de graduação da Universidade Técnica de Viena [38]. Nesta adaptação, além



Figura 2.10: *Phun* - Jogo para PC [1]

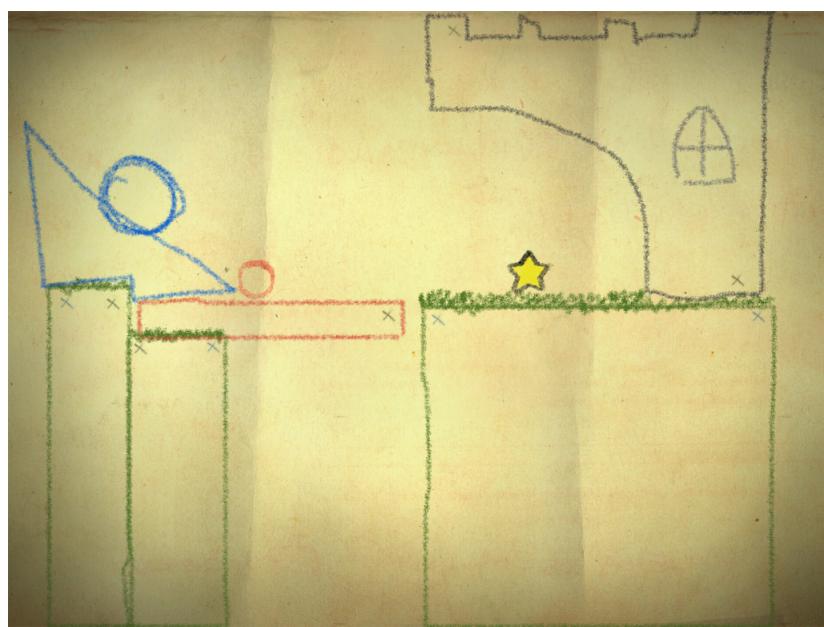


Figura 2.11: *Crayon Physics*- Jogo para PC [25]

das fases existentes no *Numpty Physics*, foram incluídas seis novas fases com uma bola adicional, permitindo a participação de um segundo jogador, como pode ser observado na Figura 2.12. No entanto, este jogo apresentou-se instável, pois é suscetível a erros de execução.

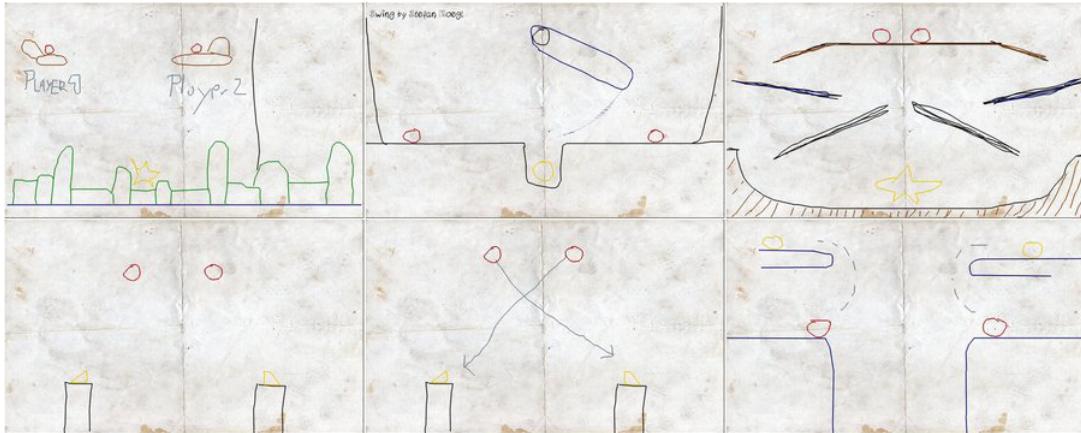


Figura 2.12: Seis fases adicionais da adaptação do *Numpty Physics* para mesas multi-toque. [38]

Neste capítulo foi possível observar o potencial de criação livre e independente que os *softwares* simuladores de física apresentam. Além disso, são intuitivos e podem ter sua interação fortemente melhorada se implementados para uma interface sensível ao toque. Em termos de usabilidade, neste tipo de interface, *softwares* simuladores de física podem ser implementados para múltiplos usuários interagindo simultaneamente. Contudo, dentre os trabalhos desenvolvidos e analisados, não existem *softwares* simuladores de física desenvolvidos para mesas com superfície multi-toque que apresentem implementações robustas e satisfatórias.

Capítulo 3

Fundamentação Teórica

Tradicionalmente, existe a prática de implementar jogos e outros *softwares* direcionados a um único usuário. Algumas iniciativas até consideram dois ou mais usuários, porém, cada um realizando seu *input* via teclado, *joystick* ou *mouse*. *Softwares* desenvolvidos para superfícies multitoque, no entanto, possuem um paradigma diferente, sendo fundamental o desenvolvimento de suporte a um número indeterminado de usuários, que podem compartilhar uma única superfície multitoque por meio da qual é realizado *input* simultâneo. Por isso são utilizadas técnicas diferentes para reconhecimento de cada toque e para o processamento das imagens que traduzem cada *input*.

Neste capítulo são apresentadas as técnicas para detecção de toques. Ou seja, é detalhado como as imagens obtidas pela câmera são processadas para detecção da formação e movimentação de *blobs*(*Bright Luminescent Objects*) que correspondem aos toques e gestos sobre a mesa. Este processamento envolve a aplicação de filtros, além da correção da imagem para eliminar a distorção. Também são discutidos aspectos gerais das mesas multitoque, tais como seu tamanho e a interface com o usuário. Por fim, analisa-se como trabalhar com um número indeterminado de usuários e como é feita a detecção de múltiplos gestos sobre a mesa.

3.1 Processamento de Toques

Para o reconhecimento de toques é fundamental a implementação de um sistema de visão computacional. Como é explicado em [5], a visão computacional trata da capacidade dos computadores de capturar informações visuais sobre o ambiente e processá-las, interpretando o que foi capturado. Assim, se faz necessário um sistema capaz de interpretar o toque em uma mesa, identificando seu local de ocorrência e extraíndo, das imagens obtidas, informações sobre ele. Esse sistema é composto por um computador tradicional, uma *webcam* sem filtro infravermelho, porém, com um filtro para luz visível e um projetor multimídia. O projetor, localizado sob a mesa, projeta imagens na superfície de acrílico. Enquanto isso, a *webcam*, também localizada sob a mesa, capture imagens da superfície que serão interpretadas por um *software* responsável por identificar o local dos toques.

3.1.1 Métodos de Iluminação de Mesas Multi-Toque

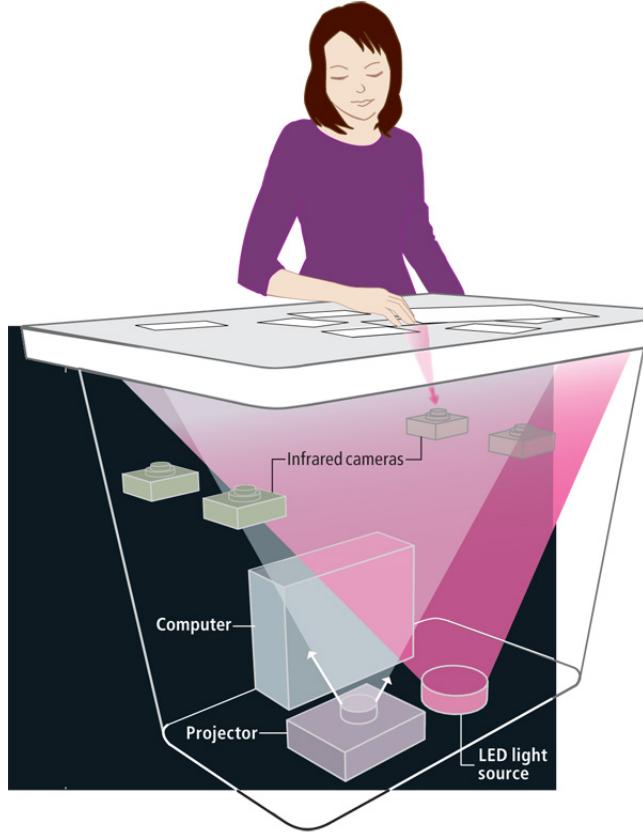


Figura 3.1: Projeto de mesa multitoque com DI [6]

Como mencionado no capítulo anterior, existem dois tipos de iluminação principais, *DI*(*Diffuse Illumination*) e *FTIR*(*Frustrated Total Internal Refraction*). A iluminação por *DI*, ilustrada na Figura 3.5, consiste de *LEDs* que iluminam a mesa por baixo do acrílico. Esse método é utilizado, por exemplo, na *Microsoft Surface* [31] e na mesa construída para o jogo *Eco-Defense* [5]. Ao se utilizar este método, tem que se cuidar para que a iluminação seja igualmente espalhada pela superfície, pois diferenças de iluminação podem dificultar a detecção de toques corretamente em partes diferentes da mesa. Além disto, esta iluminação ilumina além do toque tudo que está acima da superfície da mesa, possuindo muita interferência do fundo. Assim, o ambiente no qual se encontra a mesa deve ser controlado para seu correto funcionamento.

O outro método de iluminação é o *FTIR*, acompanhado na Figura 3.2. Nesta técnica, os *LEDS* são dispostos paralelos ao acrílico da superfície da mesa, ou diretamente dentro do acrílico através de furos ao seu redor. Assim, a luz proveniente dos *LEDs* sofre de um fenômeno denominado reflexão interna total frustada [37]. Isso significa que a iluminação fica retida dentro do acrílico sem conseguir sair. Ao momento que um dedo toca na superfície, a capacidade de refração da superfície muda, o que permite que a luz saia e ilumine o toque, onde uma câmera infravermelho consegue captar este dedo, detectando o local do toque. Esse tipo de detecção é normalmente mais precisa que a detecção por *DI*,

pois sofre menos interferência do fundo. Entretanto, ela é inadequada no reconhecimento de marcadores fiduciais, pois dependendo do material desses marcadores, não há nitidez na iluminação *FTIR* para poder identificar e distinguir as marcas se a superfície não tiver seu coeficiente de refração alterado.

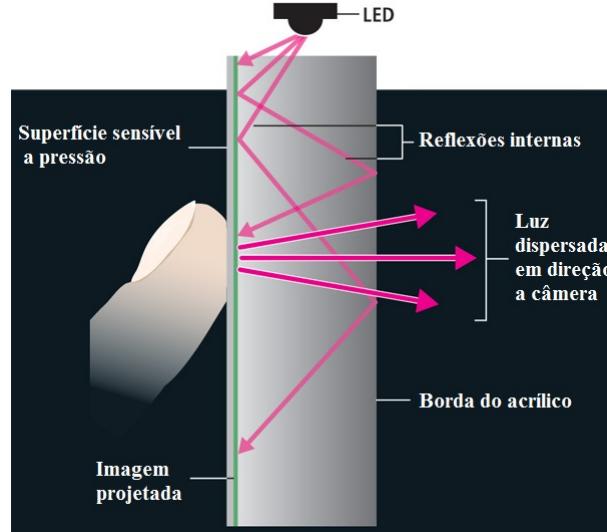


Figura 3.2: Esquema de detecção de toque *FTIR* (adaptada de [6])

Outra abordagem trata da combinação entre as duas abordagens anteriores, exemplificado pela *Virttable* [29] (Figura 3.3), que utiliza este método procurando proporcionar a melhor detecção possível de toques e marcadores fiduciais simultaneamente. Um problema desta abordagem é a interferência do fundo para a detecção de toques, fazendo com que o ambiente no qual a mesa se encontra tenha que ser controlado para sua correta detecção análogo a iluminação *DI*. No entanto, em relação a uma mesa com iluminação exclusivamente *DI*, os toques são mais bem iluminados, facilitando as detecções, o que torna este método mais adequado ao tratamento de fiduciais e toques juntos.

3.1.2 Processamento das Imagens

As imagens devem ser capturadas rapidamente em sequência, de modo que a detecção de movimento ocorra isenta de erros. A captura dessas imagens se dá por intermédio de uma câmera sem filtro infravermelho com um filtro para luz visível, como o negativo de filme fotográfico, de forma a simplificar seu formato, facilitando o seu tratamento e evitando que se confundam com as demais imagens de fundo que não são inerentes ao toque na superfície do acrílico. Depois de capturadas, as imagens são filtradas de modo que os toques estejam destacados. Por fim, os conjuntos de *pixels* agrupados que possuem intensidade semelhantes, os chamados *blobs*, são detectados e mapeados, demarcando a posição exata onde o toque foi realizado.

Conforme mencionado na seção anterior, existem dois principais métodos de iluminação de uma mesa multitoque, *FTIR*(*Frustrated Total Internal Refraction*) e *DI*(*Diffuse*

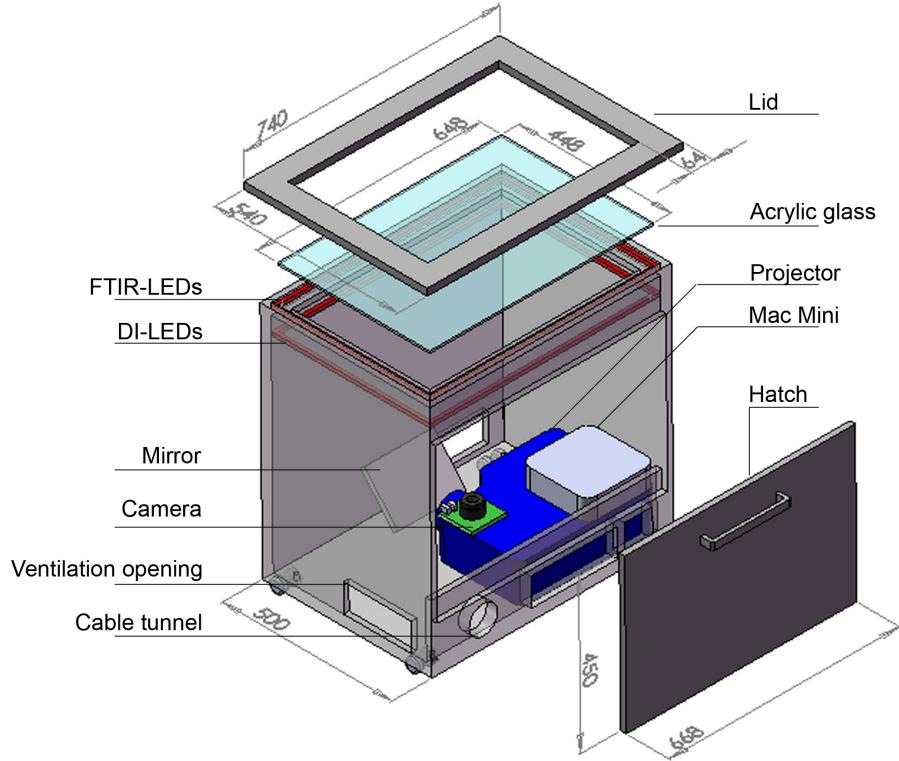


Figura 3.3: Projeto da mesa *Virttable* [29]

Illumination). Como é apresentado em [29], a iluminação por *FTIR* é mais adequada para detecção de toques, enquanto a iluminação por *DI* é mais indicada para se detectar marcadores fiduciais sobre a mesa. A Figura 3.4 demonstra as imagens capturadas por uma câmera localizada dentro da mesa *Virttable* para os diferentes tipos de iluminação, ilustrando seu efeito sobre toques e marcadores fiduciais.

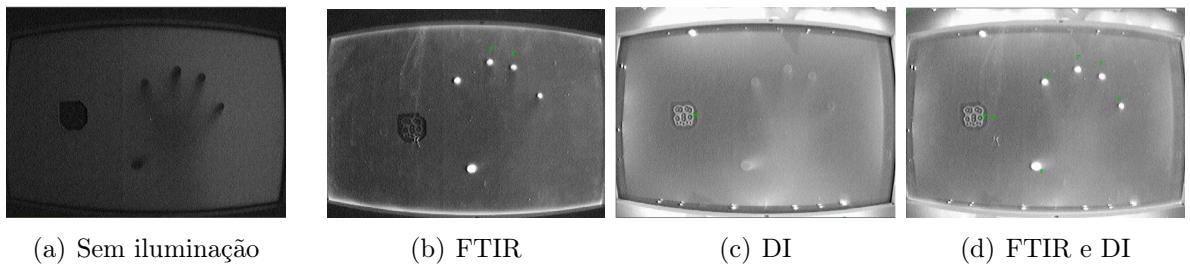


Figura 3.4: Imagens capturadas com métodos de iluminação diferentes [29].

Dependendo do método de iluminação, filtros diferentes devem ser aplicados, como explicado em [35], de forma a se excluir interferências e selecionar apenas os toques. A biblioteca *Community Core Vision* [8], utilizada neste trabalho, possui diversos filtros. Os filtros utilizados serão descritos no Capítulo 4.

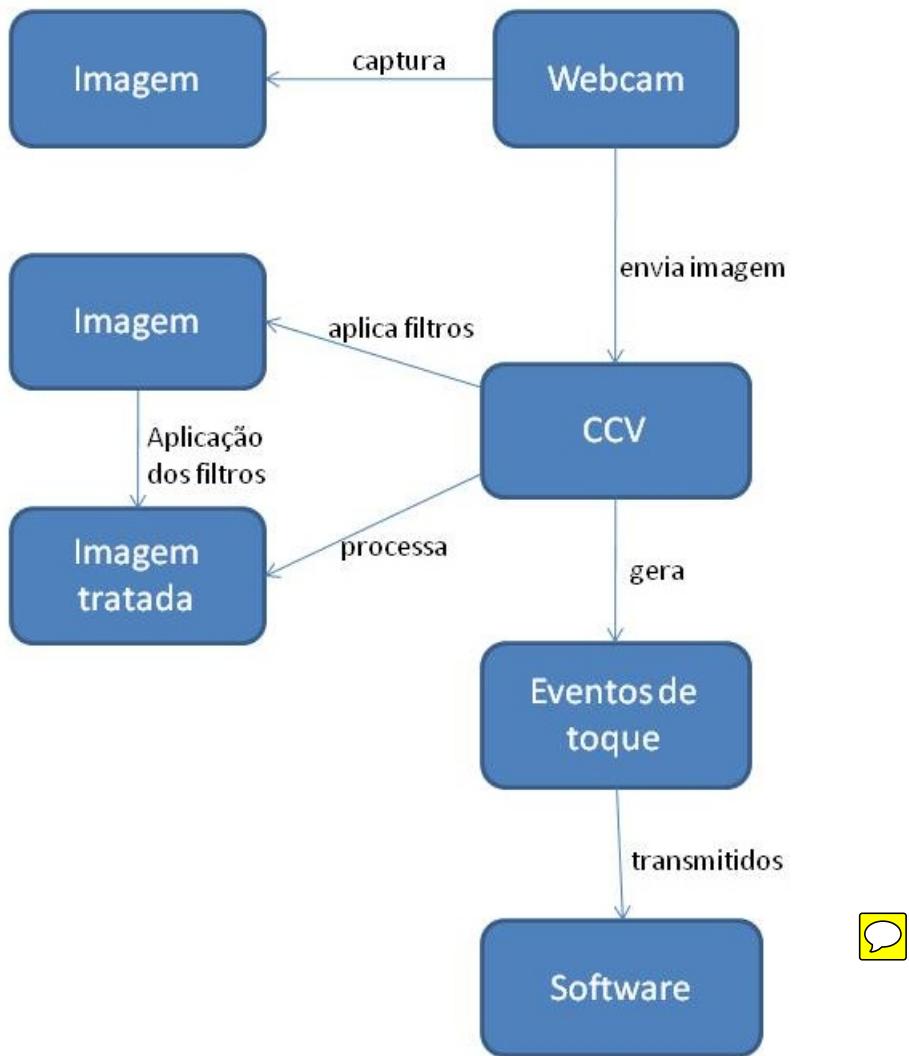


Figura 3.5: Diagrama do processo de detecção dos toques.

3.1.3 Detecção de *Blobs*

Conforme apresentado na seção anterior, os chamados *blobs* (*Bright Luminescent Objects*) são os resultados da detecção dos toques na superfície. Para interpretar esses *blobs*, existem dois processos essenciais, denominados de *Blob-detection*, que interpreta o local em que o *blob* foi criado nas imagens filtradas anteriores, e *Blob-tracking*, que é a análise dos *blobs* em uma sequência de imagens. Utilizando essas duas análises, pode-se identificar a criação de *blobs*, verificar seus movimentos durante sua vida útil, e identificar a criação de novos *blobs* no sistema. É importante ressaltar que a detecção de *blobs* deve ser realizada a cada frame.

Após passar pela cadeia de filtros aplicados sobre a imagem da câmera, ela alcança seu estágio final aonde são destacadas as interações com a interface. A partir dessa imagem final é observado os contornos dos *blobs* que aparecem e são analisados para identificar se

ele correspondem ao toque de dedos ou a marcadores fiduciais. Se for um toque, uma elipse é feita sobre ele, e com base nela, pode ser identificada a posição, orientação e tamanho do *blob*. Esse *blob* então é adicionado a uma lista para ser processado.

Com base nessa lista, o *blob-tracking* tem que ser feito para identificar a movimentação e criação dos *blobs*. Usando o frame anterior de referência, verifica-se se o *blob* foi criado, removido ou atualizado. Para se fazer essa análise, são utilizados dois conjuntos: P e Q . O conjunto P contém a lista de pontos que representam os *blobs* do frame anterior. Como exemplo, apresenta-se um conjunto P com pontos $(1,1)$ e $(4,4)$, ilustrado no gráfico da Figura 3.6.

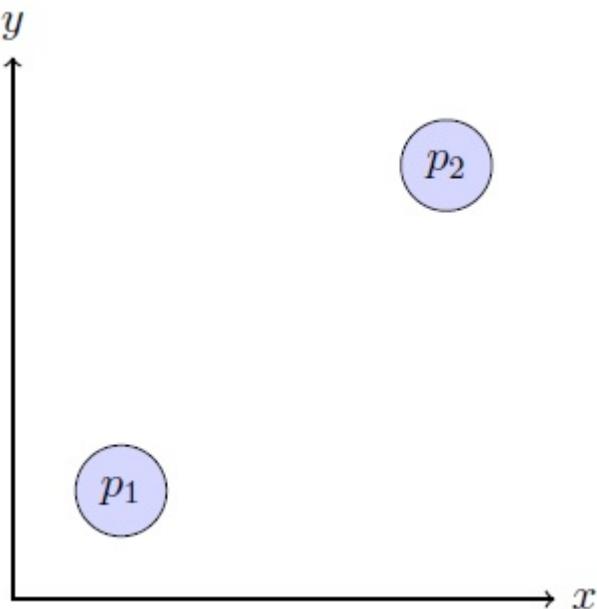


Figura 3.6: Gráfico de P [5].

O segundo conjunto, Q , contém a lista de pontos que compõe os *blobs* do frame atual. O gráfico da Figura 3.7 apresenta um exemplo para o conjunto Q com pontos $(1,2)$, $(1,4)$ e $(4,3)$.

O número de elementos de Q é maior que o número de elementos em P , logo, pelo menos um *blob* novo foi criado. Para saber qual o *blob* foi criado e quais foram somente atualizados, deve-se analisar as distâncias entre os *blobs*. Neste exemplo é demonstrado que o *blob* p_1 se moveu para q_1 , o *blob* p_2 se moveu para q_3 , e o q_2 é um novo *blob* criado neste frame. A cada frame, esse conjunto Q passa a ser o novo P , um novo conjunto Q é gerado, e o procedimento se repete.

3.1.4 Correção da câmera

Quando uma câmera captura uma imagem sobre uma superfície plana, essa imagem possui uma distorção, onde os pontos mais distantes do centro da imagem ganham um

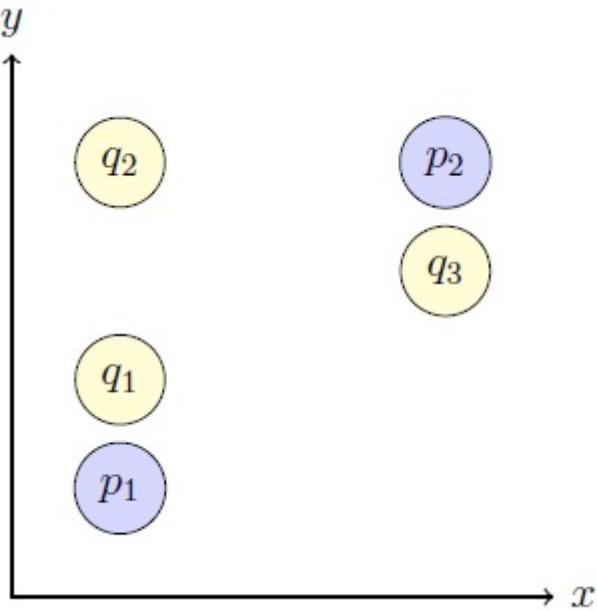
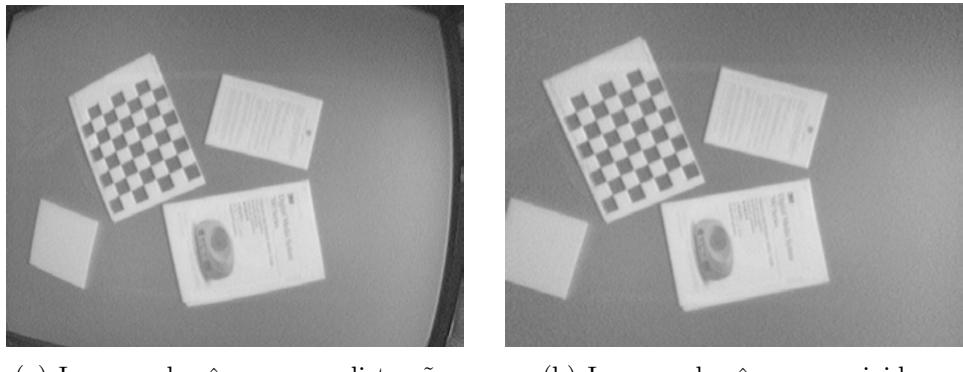


Figura 3.7: Gráfico de Q [5].

pequeno ângulo em relação à imagem original, pois são exibidos de forma reduzida. Ao se detectar *blobs*, é necessário executar alguns algoritmos a fim de corrigir essa distorção e colocar a imagem inteira em um mesmo plano. De acordo com *Muller* em [35], várias bibliotecas de processamento de imagem e detecção de multitoque já fazem isso. Na Figura 3.8 é possível visualizar uma imagem antes e depois da correção.



(a) Imagem da câmera com distorção. (b) Imagem da câmera corrigida.

Figura 3.8: Correção da distorção focal da câmera [35].

Segundo *Muller* [35], essa correção realiza um mapeamento de pontos na imagem em relação a um ponto do espaço. Para tal, é necessário considerar a distância focal, o centro da imagem, o tamanho de cada pixel, o coeficiente de distorção radial da lente, as propriedades da própria câmera, bem como o vetor de translação e a matriz de rotação que analisam o espaço em que se encontra a câmera.

Assim, um ponto na câmera pode ser corrigido de acordo com a equação

$$m = A[Rt]M \quad (3.1)$$

onde A é a matriz que possui os fatores da câmera (Equação 3.2), M é um ponto do espaço, R é a matriz de rotação da câmera e t é o vetor de translação da mesma.

$$A = \begin{vmatrix} f_x & 0 & c_x \\ 0 & f_x & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (3.2)$$

Equação dos fatores da câmera, onde f_x, f_y e c_x, c_y correspondem, respectivamente, a distância focal e o centro da câmera em um ponto (x, y) arbitrário da imagem.

Essas medições podem ser realizadas analizando uma imagem com um padrão reconhecido sendo colocado em pontos diferentes da câmera. Ao se processar esta imagem, pode-se medir a diferença entre o seu tamanho ideal e o tamanho obtido para se obter os valores ditos antes. Porém, filtros que aplicam esta correção podem ser pesados, deixando mais lenta a captura de toques.

Uma alternativa a esta correção seria a adaptação de toques, como é feito pela *CCV* (*Community Core Vision*). Neste caso, são desenhados pontos na superfície em que o usuário deve tocar quando requisitado, como pode ser observado na Figura 3.9. Assim, o programa grava a posição em que o toque foi realizado, e ao se realizar um toque futuramente, sua posição é recalculada imediatamente baseada na proximidade com os pontos obtidos na calibração. Esta forma é mais eficiente que a aplicação do filtro de *barrel correction*, e possui uma precisão boa dependendo de quantos pontos forem usados na calibração.



Figura 3.9: Calibração de pontos na *CCV* [41].

3.2 Aspectos de mesas com superfície multitoque

As mesas com superfícies multitoque mudam a forma como o usuário interage com o computador. Devido a esse novo paradigma, é necessário repensar as capacidades que elas oferecem aos usuários. Os *softwares* atuais levam em consideração que o usuário está sentado na frente de uma tela, controlando um dispositivo apontador, o *mouse*. Uma mesa multitoque, no entanto, convida várias pessoas a partilharem simultaneamente a sua utilização. Neste contexto, observa-se que a maioria dos *softwares* atuais não estão aptos a serem utilizados neste tipo de estrutura de interface, pois não possuem suporte a toques múltiplos e simultâneos.

Além disso, para que as mesas tornem-se comum no dia-a-dia das pessoas, elas devem ter um desempenho eficiente compatível com a realidade dos usuários. Nesta seção é discutido questões referentes ao número de usuários, o número de toques, os tipos de toque, a resolução e o tamanho da mesa.

3.2.1 Tamanho e Resolução

O tamanho de uma mesa multitoque deve ser proporcional a sua utilização. Se for projetada para ser utilizada por poucos usuários, deve ter um tamanho que permita para que um único usuário alcance todos seus lados. No entanto, se a mesa for dimensionada para o uso em grupo de um número indeterminado de usuários, pode ter seu tamanho aumentado até os limites impostos pela projeção da imagem sob o acrílico, realizada pelo projetor multimídia, e pelo ângulo de abertura da câmera utilizada para identificar os toques.

Idealmente, os *softwares* devem funcionar independentemente do tamanho da mesa, permitindo que o mesmo seja utilizado em projetos similares. No entanto, mesmo sem este planejamento, existem maneiras de tratar isso. A mesa pode ser projetada para se utilizar um dispositivo apontador, permitindo a um usuário alcançar pontos mais distantes, ou então possuir uma função que mude o tamanho da tela e a posicione mais perto do usuário.

Em geral, as mesas possuem um tamanho maior que os monitores de vídeo tradicionais, o que produz um problema sobre a resolução da imagem projetada. A maioria dos projetores possuem resolução padrão de 800x600 ou 1024x768 pixels, o que, para mesas grandes, é insuficiente e produz uma imagem distorcida. Para contornar isso, a solução é utilizar um projetor *short-throw* de alta resolução *WXGA* (*Wide eXtended Graphics Array*), capaz de atingir qualidade de imagem projetada em alta definição como o *720p*(resolução 1280x720 com varredura progressiva) ou até o *1080p*(resolução 1650x1080 com varredura progressiva). Projetores *short-throw* são capazes de projetarem imagens maiores em relação a sua distância da superfície de projeção, quando comparados a projetores convencionais.



3.2.2 Interfaces

Como citado por Brandão et al. [5], interfaces sensíveis ao toque trazem ao usuário a possibilidade de interagir diretamente com os dados. Isso conduz um grande apelo ao usuário, pois além de ser natural, é de fácil aprendizado. Por não possuir partes móveis, as interface sensíveis ao toque também podem ser usadas em equipamentos acessíveis ao público, tais como caixas automáticos e pontos de acesso a serviços específicos.

Como mencionado no início deste capítulo, a maioria *softwares* existentes são desenvolvidos para utilização por meio de teclado e *mouse*. No caso do *mouse*, a área apontada pelo cursor é bem pequena, logo, as interfaces possuem poucos píxeis para a seleção. No entanto, em interfaces sensíveis ao toque há uma configuração diferente, pois o dedo humano é maior que o ponteiro do um *mouse*, podendo acarretar em perda de precisão na detecção do local do toque. Segundo Brandão et al. [5], a qualidade da câmera é um fator determinante na precisão da interface, pois como ela captura a imagem a ser processada, diversos aspectos como luz, distância focal e resolução alteram a imagem capturada afetando a usabilidade para o usuário. Isto porque, em imagens muito claras é difícil identificar corretamente os *blobs*, sendo assim, a câmera deve possuir um filtro de luz. A distância focal da câmera deve suportar a distância entre a superfície da mesa e a câmera para que a imagem não seja capturada sem foco. Além disso, a câmera deve possuir uma boa resolução para capturar imagens com melhor qualidade.

É fundamental salientar que o aumento da resolução resulta em maior quantidade de *pixels* a cada imagem, o que auxilia nas dificuldades relativas à precisão da câmera, pois permite que as capturas apresentem com mais fidelidade o local dos *blobs*. Contudo, a questão do tamanho do toque deve ser considerada durante a construção da interface, pois este pode ser maior que o botão desejado e abranger uma região maior que a esperada. Em geral, o ponto tomado como referência da posição do toque é o seu ponto central.

Além desses problemas, existe a questão do ponto de visão do usuário. Quando um usuário utiliza um computador, ele está acostumado a possuir uma interface virada para ele. No caso da mesa, há vários usuários e eles podem estar olhando a mesa de ângulos diferentes. A interface deve ser agradável para todos os usuários que forem utilizar a mesa. A diferença de ângulo de visões é melhor exemplificado na Figura 3.10.

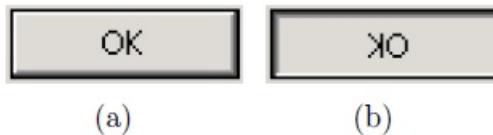


Figura 3.10: O tradicional botão *OK* visto por dois ângulos distintos [5].

Assim, ao se planejar um software para uma mesa multitoque, deve-se prever a rotação da interface de modo a acomodar a aplicação de acordo com a posição do(s) usuário(s).

3.2.3 Número de Usuários

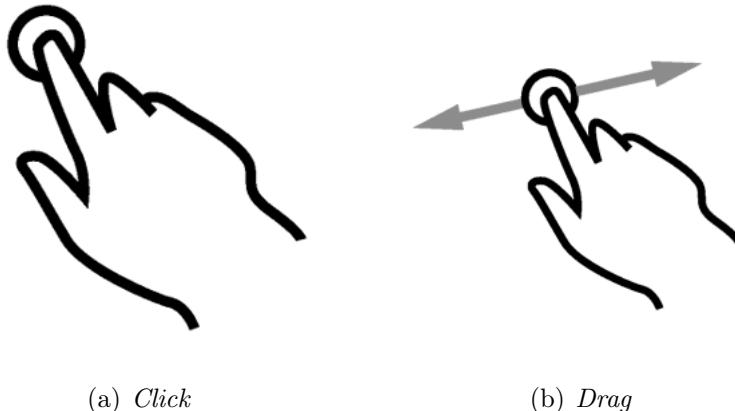
Neste contexto de mesas multitoque, uma das principais dificuldades com relação ao número indeterminado de usuários, é identificar a quem pertence determinado toque, bem como atribuir as propriedades necessárias a cada um deles. Toma-se o exemplo usado em [5], que cita o caso de uma aplicação para desenho em que um usuário deseja trocar a cor de seu pincel, de amarelo para azul, acarretando em um problema na decisão de trocar a cor para somente o usuário desejado.

A solução, nesse caso, é trocar a cor para todos os toques, ou então implementar a divisão de áreas de trabalho, onde cada usuário possui sua respectiva área e as modificações feitas em sua área não afetam as de outros usuários.

3.2.4 Detecção de Gestos

A maioria dos *softwares* leva em consideração que os gestos são oriundos apenas de um único *input*, o *mouse*. Assim, são implementados geralmente gestos com um ponto de contato. Tradicionalmente, existem os seguintes gestos:

- *Click* (Figura 3.11(a)): gesto mais simples e tradicional. É detectado ao se estabelecer um ponto de contato que é retirado logo em seguida.



(a) *Click*

(b) *Drag*

Figura 3.11: Gestos de *Click* e *Drag* [50].

- *Drag* (Figura 3.11(b)): continuação do gesto acima (*Click*), identificado ao se estabelecer um ponto de contato seguido de movimentação antes da retirada do contato.

Poucos *softwares* implementam algum tipo de gesto diferente usando somente um ponto de contato compostos de desenhos, como o que ocorre no jogo *Black and White* [4] em que o jogador pode selecionar um poder diferente dependendo do movimento que faz com o mouse, ilustrado na Figura 3.12.



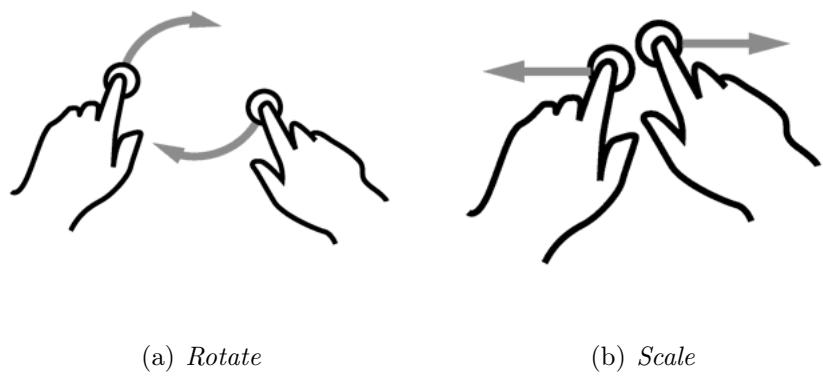
Figura 3.12: Reconhecimento de gesto no jogo *Black and White* [4].

Para identificar esse tipo de gesto é preciso identificar a forma que o usuário está desenhando na tela. Isso é feito a partir da identificação da direção em que o jogador está desenhando e o ângulo que ele está seguindo. Como o ser humano não possui uma precisão exata para desenhar, esses valores têm que ser aproximados e comparados a um banco de dados de gestos conhecidos, e aquele que se aproximar mais do padrão, dentro de um limite, é o gesto detectado. Caso não se aproxime de nenhum gesto conhecido, o gesto não é reconhecido e o usuário deve fazer outro gesto.

Em uma mesa multitoque a possibilidade de identificação de vários pontos de contato habilita a implementação de outros gestos, além dos mencionado acima. Como exemplo, podemos citar:

- *Rotate* (Figura 3.13(a)): dois pontos de contato são inseridos sobre uma superfície e são movimentados em direções opostas, enquanto se mantêm a distância entre os dois dedos constantes. O ângulo entre os dois dedos é calculado e denomina a rotação do objeto.
- *Scale* (Figura 3.13(b)): análogo ao *Rotate*, utiliza dois pontos de contato sobre uma superfície, porém, neste caso mantém-se o ângulo constante e varia-se a distância entre os dedos. Uma diminuição na distância significa uma diminuição na escala, enquanto um aumento de distância significa um aumento de escala.

Além destes, podem existir vários gestos não tradicionais, como múltiplos pontos de contato seguindo em direções diferentes para selecionar uma opção, por exemplo, como pode ser observado em [28]. Analisando a quantia de pontos possíveis em uma mesa, existem inúmeras possibilidades de gestos usando combinações de pontos de contato.



(a) *Rotate*

(b) *Scale*

Figura 3.13: Gestos de *Rotate* e *Scale* [50].

Capítulo 4

Construção da Mesa Multi-toque

Este capítulo apresenta o projeto, a construção e a montagem da mesa multi-toque concebida como plataforma de execução do *software Deskworld* proposto neste trabalho. Para iluminação da superfície da mesa e reconhecimento dos toques utilizou-se o método *FTIR*.

4.1 Estrutura da Mesa

A estrutura da mesa é feita de alumínio como material principal, de modo a prover estabilidade, leveza e mobilidade, sem prejudicar sua eficiência. Sua confecção foi sob medida, pois cada elemento influí na sua configuração. As dimensões da mesa estão inclusas na Figura 4.1 que apresenta o projeto da mesa. Visão de dentro e lateral do projeto da mesa está ilustrada na Figura 4.2.

Outra característica importante da mesa é a possibilidade de ser desmontada, o que facilita seu transporte. Apesar de possuir grandes medidas, uma vez desmontada pode ser facilmente carregada e armazenada, pois sua estrutura de alumínio é leve. Além disso, mesmo montada, a mesa possui rodas com travas para facilitar a movimentação, podendo ser travada no local de uso. A mesa completa pode ser observada, externa e internamente, nas Figuras 4.3 e 4.4, respectivamente.

4.2 Superfície

A superfície da mesa é feita de uma chapa de acrílico translúcido, que oferece uma alta resistência e transparência a um custo acessível. Seu coeficiente de refração é adequada e permite a iluminação de toda superfície. Sua transparência permite a captação dos toques com clareza pela câmera. O acrílico tem dimensões de 120 x 90 cm e espessura de 1 cm. Para reter a projeção é aplicado um papel vegetal sob a face interna do acrílico. O objetivo desta é impedir que a luz do canhão do projetor multimídia reflita no acrílico

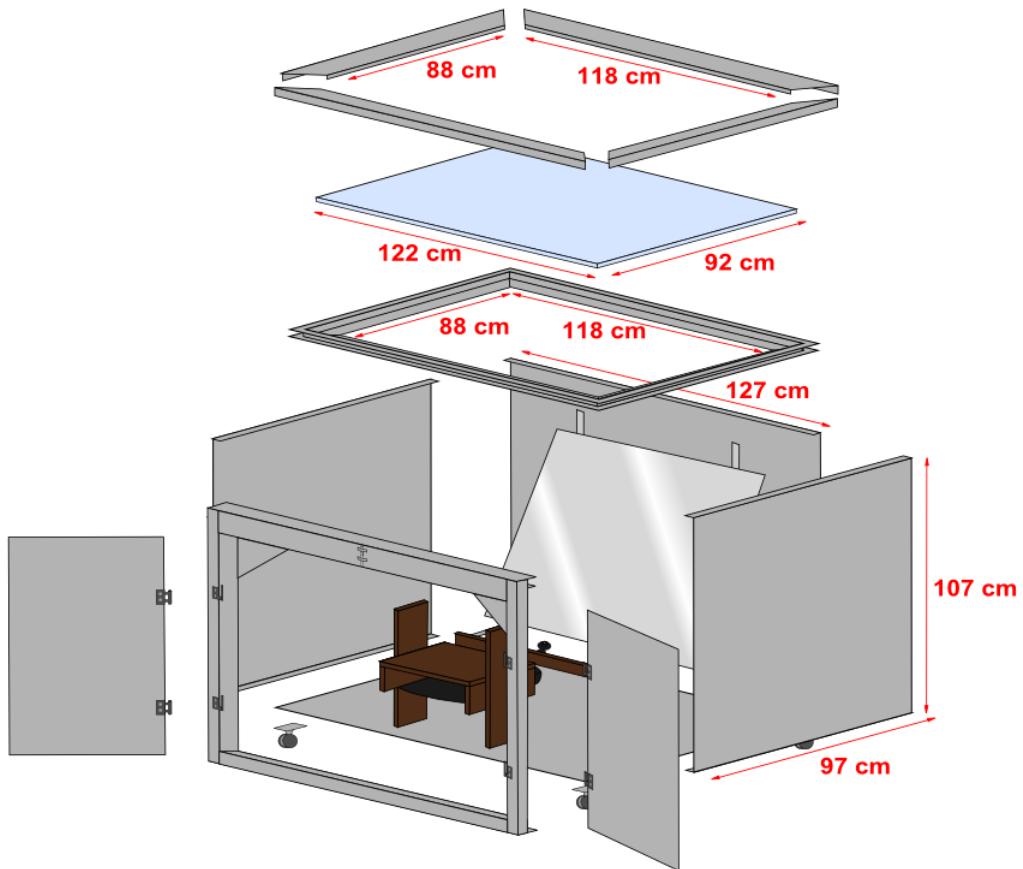


Figura 4.1: Projeto da mesa com superfície multi-toque construída neste trabalho.

causando uma interferência detectável pela câmera. Além disso, ele ajuda a destacar a iluminação dos dedos, quando comparadas a da mão, facilitando a detecção de toques.

4.3 Iluminação Infravermelha

Para um reconhecimento satisfatório dos toques, foi utilizado o método de iluminação FTIR, introduzido no capítulo anterior. Os *LEDs* infravermelhos foram posicionados em furos ao redor do acrílico apontados paralelamente a sua superfície (Figura 4.6). Aproveitando a resistência do acrílico, furou-se a lateral do acrílico a, aproximadamente, cada 1cm utilizando uma furadeira de bancada para o posicionamento e fixação dos *LEDs*. Foram realizados 200 furos no total, 115 em um dos lados da mesa com 120 cm e 85 em um dos lados com 90 cm. Os *LEDs* foram então inseridos nos furos e soldados (Figura 4.7). Os *LEDs* são diodos de 5 mm de diâmetro com potência 1,1 w máxima, tendo a sua tensão máxima 5 V e corrente 220 mA. Como é utilizada uma fonte com tensão de 12,2 V, **potência máxima de 60 w e corrente máxima de 5 A**, os *LEDs* foram soldados de acordo com o diagrama da Figura 4.8, (com séries de 8 *LEDs* e um resistor para regular a corrente, todas essas séries em paralelo entre si).



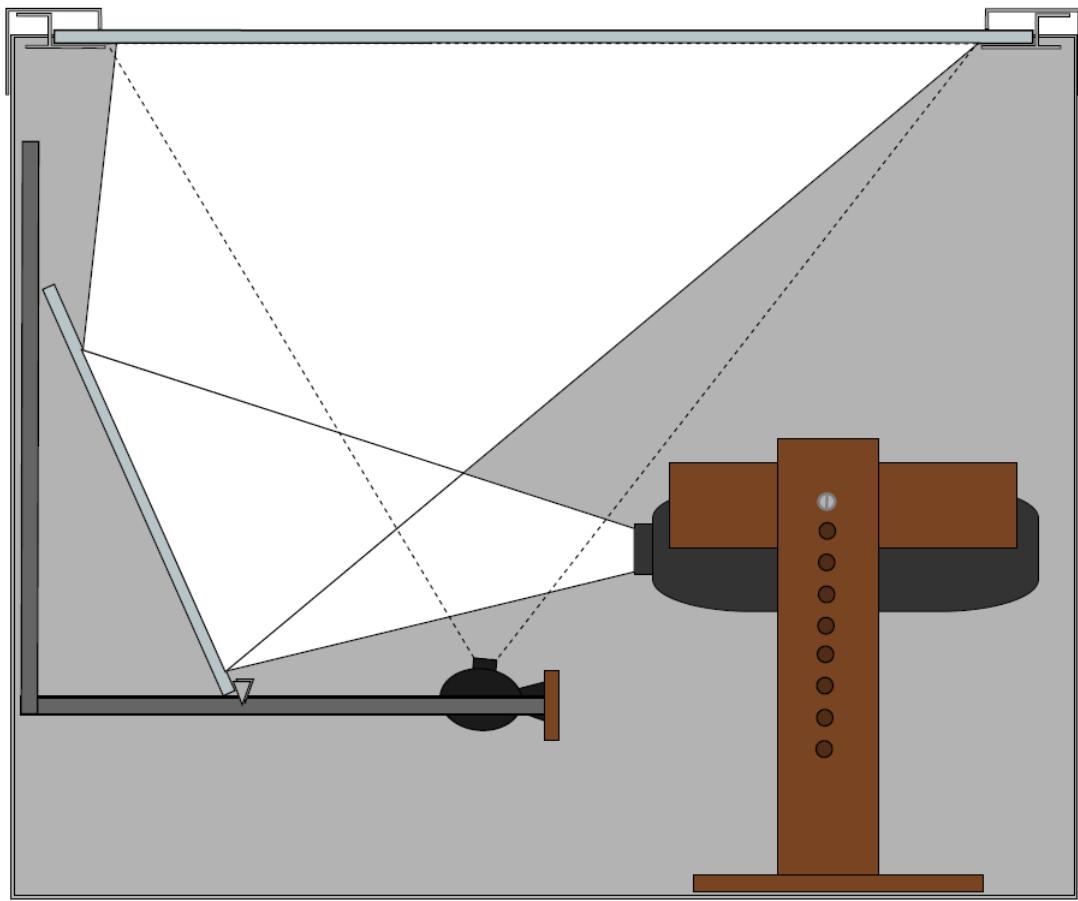


Figura 4.2: Visão lateral do projeto da mesa com superfície multi-toque construída neste trabalho.

4.4 Câmera

Para capturar a imagem a ser processada de modo a identificar os toques sobre a superfície, foi utilizada uma câmera *PlayStation Eye*, que pode ser observada na Figura 4.5. A escolha foi baseada no baixo custo da câmera, seu ângulo de abertura, pois permite a captura de imagens da superfície inteira com apenas uma câmera, e a alta taxa de frames por segundo em sua faixa de preço. Sua resolução pode ser configurada em 640 x 480 *pixels*, ou em 320 x 240 *pixels* numa taxa de atualização de 75 ou 120 frames por segundo, respectivamente.

Para capturar os movimentos sobre a mesa com maior precisão, modificou-se a câmera de modo a permitir a filtragem do espectro de luz visível e captar apenas luz infravermelha. Isto foi feito primeiro removendo o filtro nativo que a câmera possui, que bloqueava a iluminação infravermelha, e depois aplicando-se um filme fotográfico sobre a lente da câmera, que filtra a luz visível mas permite a passagem de luz infravermelha normalmente.



Figura 4.3: Fotos da mesa multi-toque construída neste projeto.

4.5 Projetor

Nesta mesa, o projetor utilizado é da marca *BenQ* modelo *MP772ST* (Figura 4.9). Trata-se de um projetor *WXGA*(*Wide eXtended Graphics Array*) de resolução padrão 1024x768 pixels com suporte a projeção até 1080i. O projetor foi posicionado a uma distância de 90cm da superfície do acrílico. O projetor é fixado ao fundo da mesa e utiliza-se de um espelho para refletir a imagem até o topo, sobre o papel vegetal que se encontra diretamente abaixo da chapa de acrílico.

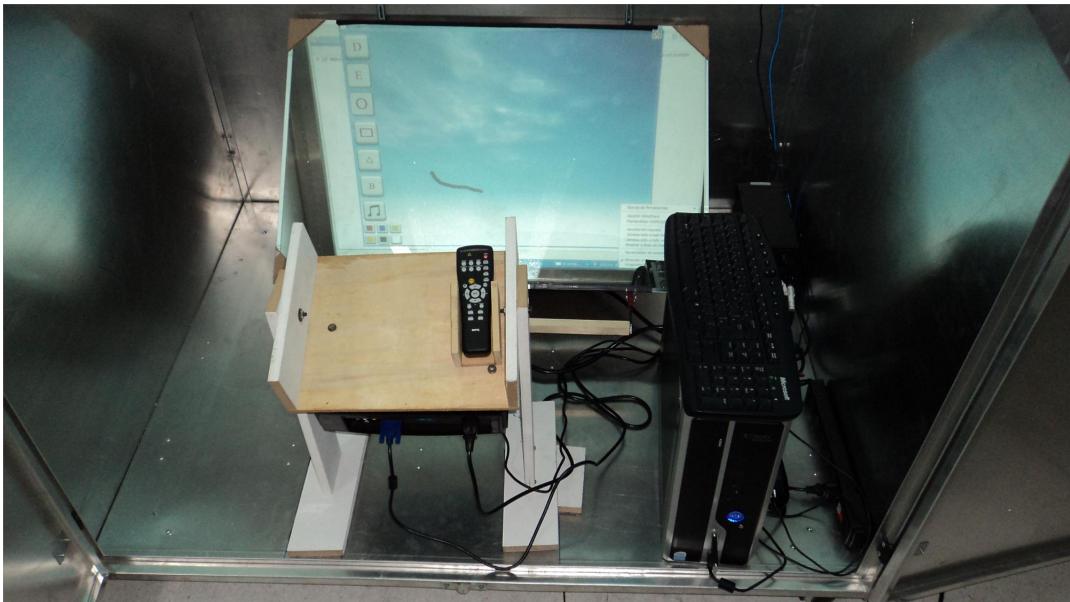


Figura 4.4: Visão interna da mesa, ilustrando seus componentes.



Figura 4.5: Câmera *Playstation Eye* no suporte.

4.6 Filtros de Processamento de Vídeo

Para detectar corretamente os toques, é necessário processar cada imagem recebida pela câmera de forma a separar os toques das interferências das imagens de fundo. Para isso, utilizou-se uma cadeia de filtros disponíveis na biblioteca *Community Core Vision (CCV)* [8].



Figura 4.6: Furos efetuados na borda do acrílico.

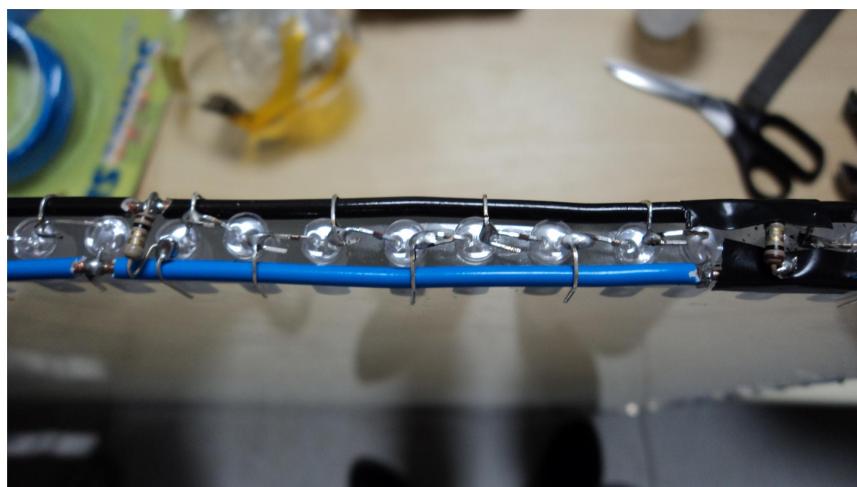


Figura 4.7: *LEDs* soldados na borda do acrílico.

A Figura 4.10(a) mostra a captura direta da câmera, sem nenhum filtro de processamento aplicado, apenas o filtro de hardware apresentado na Seção 4.4. Na Figura 4.10(b), pode-se ver o filtro de eliminação de fundo, que guarda uma imagem inicial como referência e depois subtrai as imagens subsequentes, de modo a remover grande parte da interferência do fundo e de imperfeições do acrílico. Em seguida, na Figura 4.10(c), é aplicado o filtro de passo-alto, que aumenta a luminosidade dos pontos que estão a determinada distância da câmera, removendo a luminosidade de pontos além destes, de modo a reduzir ruídos. Já na Figura 4.10(d), tem-se o filtro amplificação, que intensifica a imagem obtida até o momento, proporcionando *blobs* brilhosos e de fácil detecção. Finalmente, na Figura 4.10(e), é aplicado o filtro de retificação, que elimina pequenas interferências restantes, pois filtra *blobs* abaixo de um certo nível de tamanho. Neste caso, é imprescindível que somente os toques estejam visíveis, pois esta é a imagem usada para detectá-los.

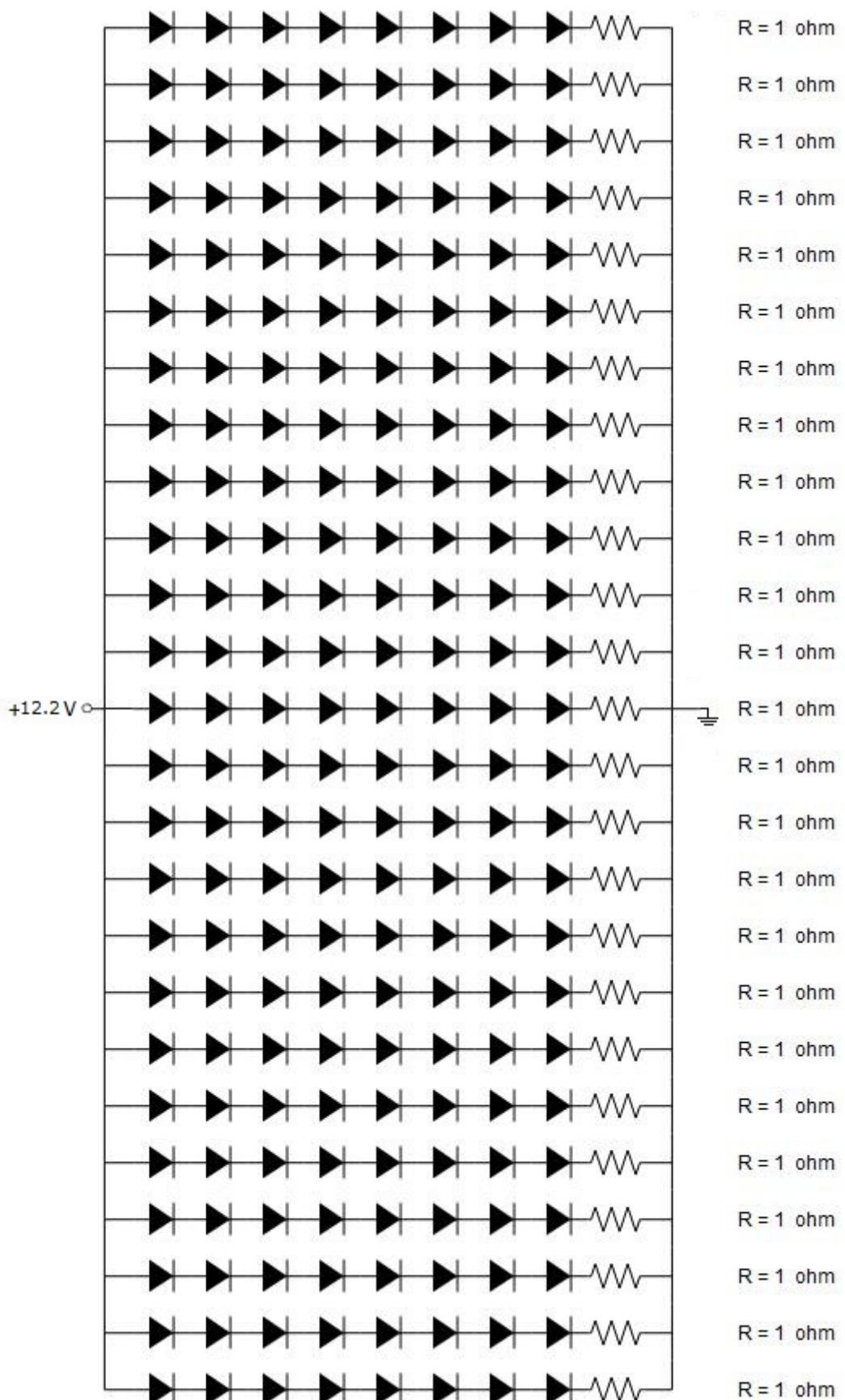


Figura 4.8: Esquema de montagem do circuito dos *LEDs* infravermelhos.



Figura 4.9: Projetor multimídia *BenQ MP772ST* [9]

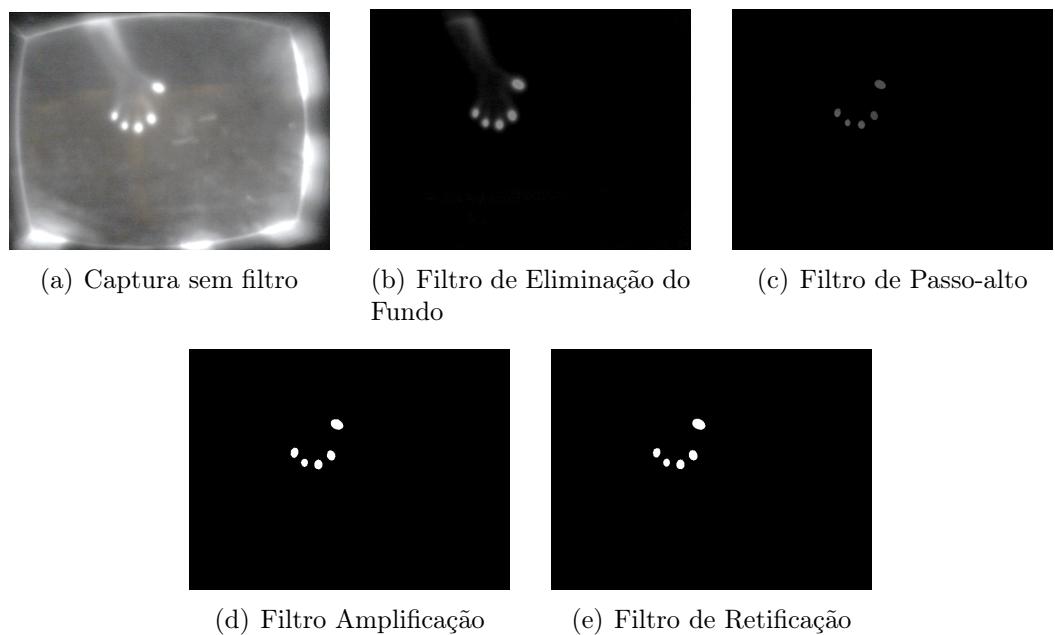


Figura 4.10: Aplicação dos filtros à imagem capturada pela câmera do sistema.

Capítulo 5

Software Deskworld



Neste capítulo é apresentado o projeto do *Software Deskworld* (Figura 5.1) e seus detalhes de implementação, tais como arquitetura e a estrutura empregada.

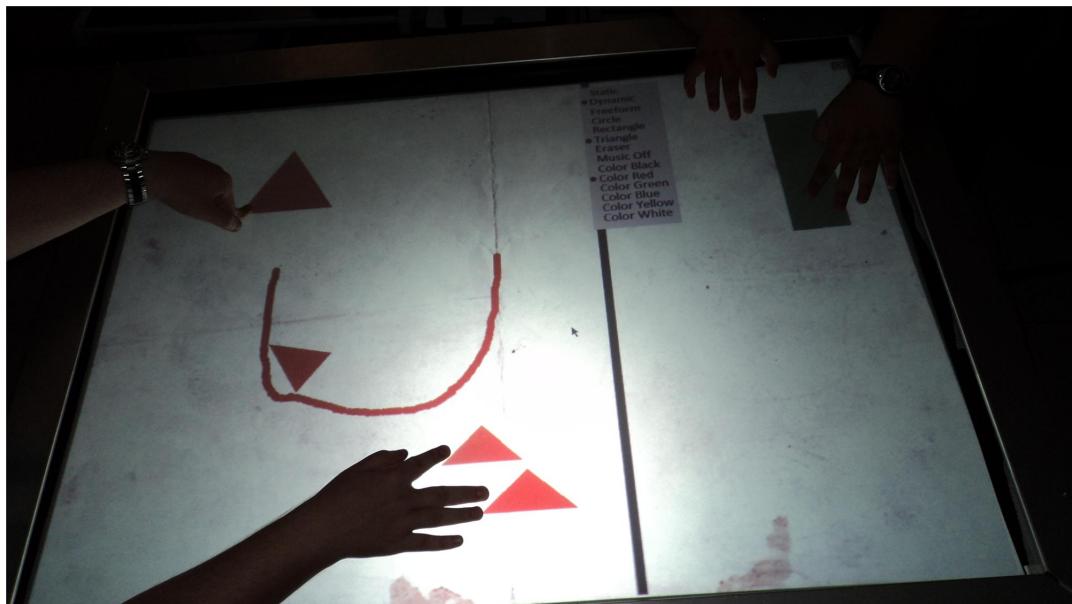


Figura 5.1: *Deskworld* sendo utilizado na mesa.

5.1 *Design do Deskworld*

Nesta seção, apresenta-se o projeto do *Software*, isto é, o projeto de sua aplicação com as características expressas em uma visão computacional de alto nível, ou seja, a nível de usuário.

5.1.1 Requisitos de interface

De maneira simplificada, os requisitos de interface do *Deskworld* podem ser expressos da seguinte forma:

- sua mecânica deve permitir a participação de diversos jogadores simultaneamente;
- deve suportar e processar toques simultâneos;
- deve possuir capacidade de aproximar linhas.

5.1.2 Conceito

O simulador de física *Deskworld* tem como principal proposta deixar a cargo da criatividade do(s) usuário(s) a construções de cenários, objetos e seus respectivos objetivos em seu mundo virtual. A meta principal é prover ferramentas aos usuários, auxiliando-os nessa construção. Assim, objetos e elementos físicos são representados em um plano bidimensional, regidos pelas leis da física. É possível desenhar um objeto e modificar suas propriedades tais como massa, coeficiente de atrito e elasticidade. O usuário pode aplicar uma força a este objeto como a gravidade, fixá-lo no plano, entre outras opções. Pode-se ainda, alterar as propriedades do mundo em que se encontra e, sem interrupções, dividí-lo, criando novos submundos para interagir separadamente.

5.1.3 Regras e Objetos

Basicamente, pode-se criar qualquer objeto bidimensional, com o auxílio das ferramentas fornecidas. A construção funciona similarmente ao desenho virtual, onde o dedo representa o lápis e a superfície da mesa o papel. Os objetos podem ser criados de forma livre, com o usuário desenhando-os como quiser, ou com auxílio do software, através de suas ferramentas ou utilizando a forma livre que, ao fechar uma forma, tenta aproximar os desenhos do usuário a polígonos convexos, como pode ser observado na Figura 5.3.

Para a alteração de regras do mundo, ferramentas utilizadas ou propriedades dos objetos, o usuário possui um menu, que pode ser acessado através de dois toques subsequentes em um local disponível da tela. Os itens do menu podem ser selecionados com um único toque neles.

Cada objeto pode ter as seguintes características:

- Densidade
- Coeficiente de atrito
- Coeficiente de restituição de força
- Força normal

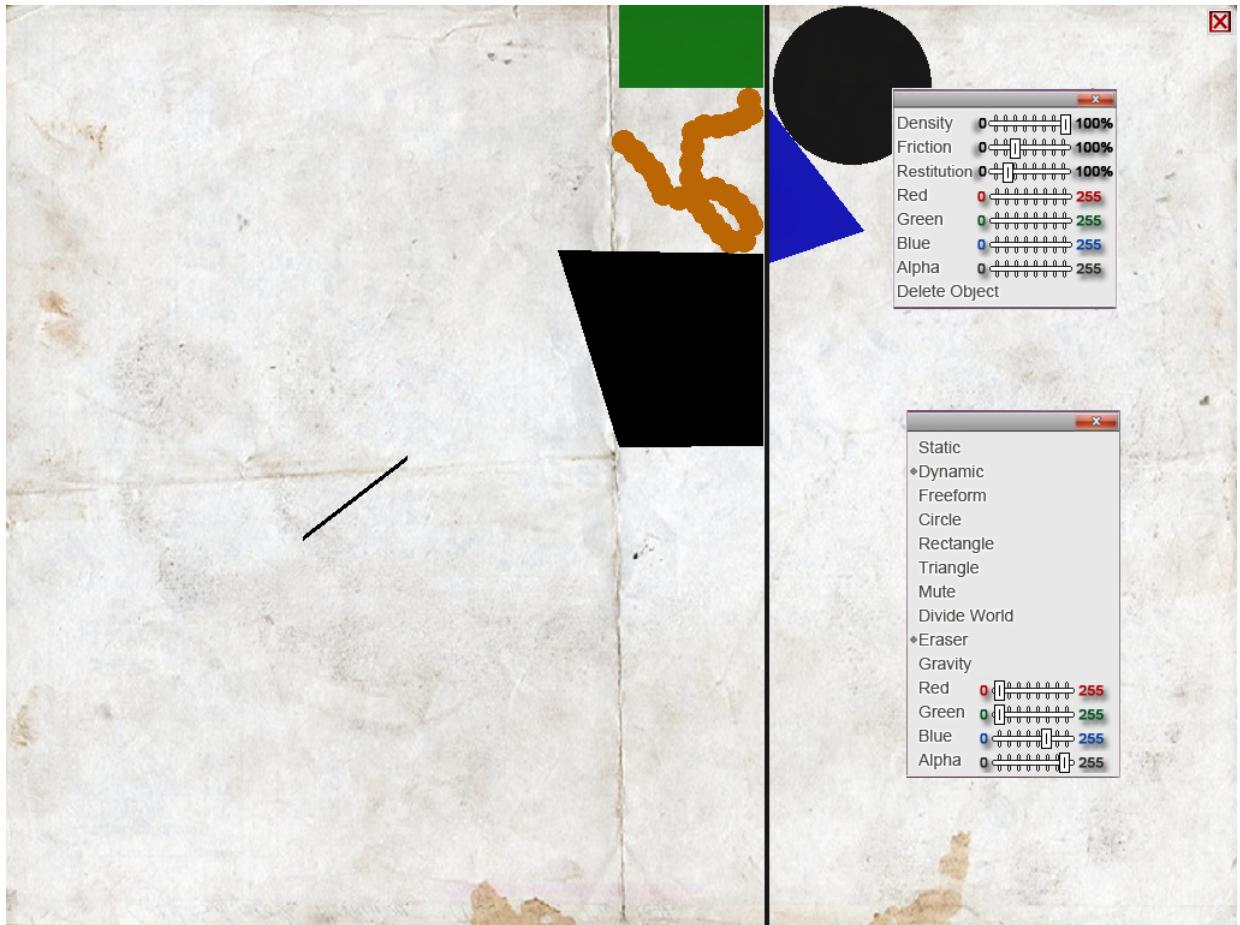


Figura 5.2: Tela do software *Deskworld*.

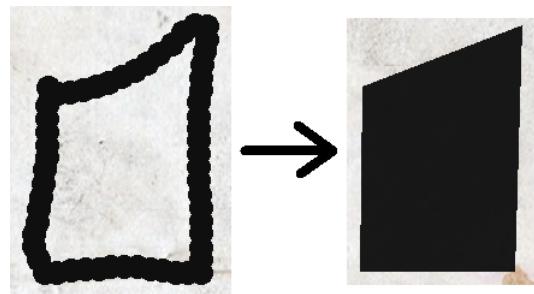


Figura 5.3: Aproximação de formas a polígonos convexos.

- Massa
- Volume
- Velocidade

Dentre as características acima, a densidade, o coeficiente de atrito e o coeficiente de restituição de força podem ser alterados pelo usuário a qualquer momento, utilizando o

menu de objetos, acessado da mesma maneira em que se acessa o menu tradicional, mas com o duplo clique sendo em cima de um objeto ao invés de um local disponível da tela. As demais são calculadas automaticamente, de acordo com as interações dos objetos com o mundo, como por colisões, movimentações ou alterações de gravidade.

Além dessas propriedades, a cor dos objetos pode ser alterada pelo menu, bem como a gravidade do mundo. Também é possível a separação de mundos com gravidades diferentes e cores de pincéis diferentes entre si. O usuário ainda tem acesso a um apagador, para retirar do mundo qualquer objeto que não queira mais. Caso o usuário apague uma divisão entre mundos, os mundos continuam divididos e com suas propriedades diferentes, mas os objetos existentes em cada mundo podem transitar livremente entre os mundos unidos. A última propriedade disponível no menu é o *Mute*, que pode ser selecionado para se desligar a música do jogo caso algum usuário deseje, ou ligar novamente caso esteja desligada.

As ferramentas e objetos disponíveis estão ilustradas na Figura 5.2.

Neste *software* interativo, não existem regras pré-definidas. Fica a critério do usuário definir suas regras enquanto interage com o *software* criando cenários e objetos.

5.1.4 Interface

A interface utilizada para testes é a mesa cuja construção foi descrita no Capítulo 4. Entretanto, salienta-se que o *software* *Deskworld* é portável para qualquer interface multi-toque que suporte o protocolo *TUIO*.

5.2 Ferramentas de suporte

Nesta seção, serão descritas as ferramentas de suporte, que fornecem aplicações reutilizáveis para facilitar o desenvolvimento de *softwares*, utilizadas na construção do simulador *Deskworld*, tais como game engine e bibliotecas.

5.2.1 Game Engine

A *engine* utilizada no desenvolvimento deste software foi a Box2D [7]. Esta *engine* é livre, ou seja, pode ser utilizada sem nenhum custo. A principal funcionalidade da *Box2D*, no contexto desse *software*, foi auxiliar na criação dos objetos e no tratamento de eventos relacionados à simulação das leis da física. Tais eventos são:

- Gravidade
- Colisão
- Força de ação e reação



- Força de atrito

Além dessas funcionalidades, a *Box2D* é responsável pela gerência de memória, junção de objetos e/ou a fixação deles. Sua escolha foi baseada em boa performance, simulação física de ambientes bidimensionais, além da estabilidade de seu código.



5.2.2 *Bibliotecas de apoio*

Para facilitar a implementação do *software*, foram utilizadas diversas bibliotecas. Uma das bibliotecas padrão do C++, a *STL* (*Standart Template Library*), possui algumas estruturas complexas de dados já implementadas, sendo amplamente utilizada neste simulador. Para a parte gráfica do jogo foi utilizada a biblioteca *SDL* [26] (*Simple Direct Layer*) em conjunto com a *OpenGL* [15] (*Open Graphics Library*). Com relação ao áudio, utilizou-se a biblioteca *SDL_mixer* [27]. Para tratamento de *input* foi utilizada a biblioteca do protocolo *TUIO* [22] (*Tangible User Interface System*) para receber as mensagens enviadas pelo *CCV* [8] (*Community Core Vision*) explicado mais a frente.

SDL e OpenGL

Por oferecer uma boa abstração de *hardware*, ser bem difundida no mercado de jogos e possuir ampla documentação, utilizou-se a *SDL* para disponibilizar o acesso ao ambiente. Em conjunto com a *SDL*, na parte gráfica, foi utilizada a *OpenGL*, que realiza a renderização das imagens.

A *SDL_mixer* é utilizada para permitir a reprodução de diversas faixas de áudio simultaneamente.

Protocolo *TUIO*

A comunicação entre o usuário da mesa e o aplicativo é realizada de acordo com o protocolo *TUIO* [22]. Este protocolo é especificado para atender as necessidades de comunicação das interfaces tangíveis, que são interfaces sensíveis ao toque, capazes de serem controladas por movimentos corporais e gestos. A implementação é simples e visa melhorar a performance na comunicação. Para isso, ele opera sobre a camada *UDP* (*User Datagram Protocol*) de transporte utilizando três tipos de mensagens: *set*, *alive* e *fseq*. As mensagens *set* são utilizadas para informar o estado de um objeto. Mensagens *alive* indicam o conjunto de objetos presentes na interface através de uma identificação única atribuída a cada novo elemento reconhecido. Mensagens *fseq* são transmitidas antes da etapa de atualização de cada quadro, para marcar-lo unicamente, associando-o a cada mensagem *set* e *alive*. A seguir é apresentado um resumo do funcionamento do protocolo:

- Parâmetros do objeto são enviados após mudança de estado, por intermédio da mensagem *set*.

- Objetos removidos da interface são comunicados por meio de mensagens *alive*.
- Cliente deduz a lista de objetos adicionados e removido por meio das mensagens *set* e *alive*.
- Mensagens *fseq* associam um ID a um conjunto de mensagens *set* e *alive* do quadro.

Apesar da camada UDP de transporte não oferecer garantia de entrega dos dados, o TUOI implementa redundância de informação para se precaver contra a perda de dados na transmissão. Além disso, o estado de um objeto, mesmo inalterado, é enviado periodicamente em uma mensagem *set*. Portanto, o protocolo é adequado para ser utilizado em aplicativos controlados por interfaces multitoque, otimizando a interação e confiabilidade da comunicação. Na Tabela 5.1, pode-se observar todos os parâmetros que são enviados a cada mensagem do protocolo *TUIO*. A posição, tal como vetor de movimento, aceleração, largura e altura do *blob* são variáveis do tipo flutuante, pois o protocolo TUOI mapeia a tela a nível com valores entre 0 e 1.

Tabela 5.1: Parâmetros de mensagens TUOI em superfícies interativas 2D

Parâmetros	Significado dos parâmetros	Tipo
s	sessionID (ID temporário do objeto)	int32
x, y	posição	float32
X, Y	vetor de movimento (velocidade de movimento e direção)	float32
m	aceleração de movimento	float32
w, h	largura e altura do <i>blob</i>	float32

Community Core Vision

O *CCV* (*Community Core Vision*) [8] é um aplicativo para auxiliar no processamento de imagens capturadas pela câmera em interações com superfícies multi-toque. A Figura 5.4 apresenta um *screenshot* do aplicativo. Ele lida com o acompanhamento dos *blobs* de luz infravermelha, enviando mensagens para o *Deskworld*, tais como o encostar do dedo, o deslocar do dedo e o retirar do dedo. Interage com a maioria dos dispositivos de captura de vídeo, sendo muito útil para um projeto de software que utilize uma mesa multitoque. Atualmente, possui suporte para plataformas *Windows* e *Linux*, 32 ou 64 bits, e *MacOS*. Ele permite, por padrão, uma quantia máxima de 20 toques simultâneos, porém este número pode ser modificado em seu arquivo de configuração. Neste trabalho, foram testados até 30 toques ao mesmo tempo diretamente com a mesa, porém o valor de toques máximos possui a possibilidade de ser incrementado ainda mais.

Seu funcionamento consiste na aplicação de filtros configuráveis, como os mostrados na Figura 4.10 no capítulo anterior. Em seu arquivo de configuração, *config.xml*, pode-se modificar os dados da câmera, tais como resolução e frames por segundo; o *CCV* irá automaticamente ajustar a câmera selecionada para os dados mais próximos suportados. A execução do *CCV* funciona de acordo com o paradigma cliente-servidor, onde o mesmo atua como servidor e a aplicação como cliente. Esta comunicação cliente-servidor é feita

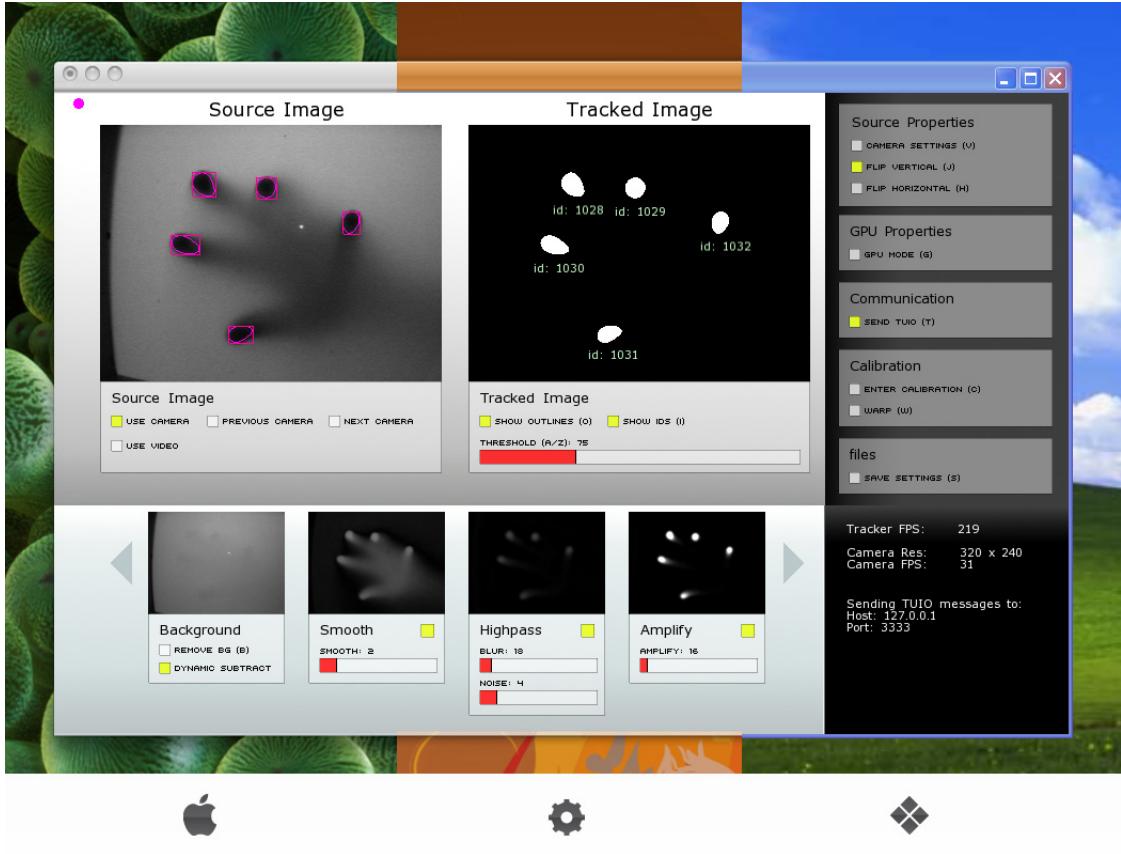


Figura 5.4: Aplicativo de captura de toques *Community Core Vision* [8]

utilizando o protocolo TUO [22] (*Tangible User Interface System*), utilizando pontos normais x e y enviados via *TCP*, ou em pacotes especiais para aplicativos *Flash*, permitindo uma arquitetura distribuída, onde o aplicativo é executado separadamente da aplicação de processamento de imagens.

5.3 Arquitetura e Detalhes de Implementação

A evolução da tecnologia dos *hardwares* para computadores pessoais proporcionou grande desenvolvimento à indústria de jogos eletrônicos, levando-a a aprimorar seu produto ao longo do tempo. Este caminho foi marcado pela formação de grandes equipes e pela extensiva carga de codificação. Os jogos eletrônicos mais modernos podem ultrapassar um milhão de linhas de código de acordo com *Rabin et al.* [39]. Portanto, foi necessário organizar estes extensivos códigos a fim de obter um melhor desempenho no desenvolvimento de novos jogos. Esta organização pode ser realizada por meio da adoção de uma arquitetura bem definida.

5.3.1 Arquitetura do *Deskworld*

Apesar de não se encaixar na definição de jogo eletrônico, o *Deskworld* possui muitas características semelhantes, pois é um *software* interativo, onde o usuário se relaciona com objetos em um mundo virtual. Dessa maneira, este *software* foi desenvolvido a partir da utilização de um padrão de desenvolvimento voltado a jogos. Para este *software* foi escolhida uma arquitetura modular orientada à objetos.

Vantagens da abordagem orientada à objetos

De acordo com *Rabin et al.* [39] a maioria dos jogos se desenvolvem em torno de objetos ou entidades em um mundo virtual, nos quais o usuário deve realizar ações. Antes da existência do paradigma da orientação a objetos, os jogos eram desenvolvidos com programação procedural e assim, a ênfase da programação baseava-se no código propriamente dito. Isto porque o jogo era observado, conceitualmente, como uma sequência de código com execução de procedimentos e funções. Na programação orientada à objetos, a perspectiva é outra, pois a ênfase ocorre no conceito de objeto que é uma coleção de informações conjuntas a uma série de operações para processar estes dados. Portanto, utilizando orientação à objetos é possível obter *softwares* que se adequam ao formato dos jogos, pois as entidades desses jogos podem ser expressas pelas classes, os objetos por instâncias da classe e ações entre eles pelos métodos.

Outra vantagem encontrada nesta abordagem é a herança de classes, que permite a reutilização de código em múltiplas classes que possuam parentesco. Com herança, é possível estender características das superclasses para as subclasses. No *Deskworld*, utilizou-se herança para, principalmente, padronizar e reutilizar parte do código dos *Game Objects*. Também importante, o suporte ao polimorfismo é crucial no desenvolvimento de jogos já que permite a realização de *update* e *render* de diversos componentes do *software* sem precisar separá-los em chamadas de execuções diferentes, pois todos serão reconhecidos pelo seu tipo e terão seus respectivos métodos executados corretamente.

A arquitetura do programa foi modularizada para especificar claramente os seus subsistemas, facilitando a manutenção e o entendimento do código. No diagrama apresentado na Figura 5.5 pode ser observada a composição de módulos do *Deskworld* e suas principais classes. A decomposição em módulos facilita a visualização da implementação, já que o *software* possui subsistemas bem claramente definidos como os de áudio, vídeo, *engine* e de *input*.

5.3.2 Detalhes de implementação

Esta seção apresenta a lógica de implementação do *software*. Primeiramente, é exposto uma visão geral do funcionamento do sistema. Em seguida, os subsistemas são detalhados e seus papéis traçados.

Visão Geral do *Software*

Como observado no diagrama da Figura 5.5, existe uma classe *Game Manager* responsável pelo gerenciamento da execução do programa. Suas principais ações são inicializar os subsistemas de áudio, vídeo, carregar o cenário e inicializar a *Engine* do *Deskworld*. A classe áudio é responsável por carregar arquivos de áudio, bem como manipulá-los, podendo tocar, parar ou resumir um arquivo carregado. A *Graphics* implementa soluções para renderizar os objetos na tela e gerenciar a memória de vídeo. O *InputManager* trata as mensagens de entrada recebidas, explorados na Seção 5.2.2, e gera eventos relativos aos toques. A *Engine* tem como objetivo criar, alterar e posicionar objetos no mundo, além de se comunicar de posse destas informações com a *Box2D*. A classe *LevelState* concentra o maior fluxo de informações, pois todas as instâncias de classe de objetos do mundo são criada nela. Os métodos de *Update*, muito importante em simulações físicas, e *Render* têm suas chamadas realizadas pelo *LevelState*. O *Game Object* armazena informações sobre os objetos como sua dimensão, posição e estado.

Subsistemas de Áudio e Vídeo

O subsistema de áudio possui atributos para identificar o nome do arquivo de áudio, o tipo do arquivo e um ponteiro para seu endereçamento na memória. Seu construtor carrega o arquivo de áudio para memória e retorna esse endereço ao ponteiro da classe. Utiliza-se esta classe para tocar e pausar a música de fundo.

A classe *Graphics* gerencia a memória de vídeo com a *OpenGL*, e a utiliza para desenhar formas geométricas na tela. A classe *ImageLoader* carrega imagens e texturas e renderizá-las. As texturas presentes no *Software* são o fundo, menu e seus componentes.

Singleton

A classe *Graphics* é um *Singleton*, isto é, possui, obrigatoriamente, apenas uma instância durante todo o programa. *Singleton* é um *design pattern* [14] muito utilizado em situações que necessitam de grande controle sobre uma classe para que seja instanciada somente uma vez. Esta técnica consiste em declarar um construtor privado que será executado uma única vez. Constrói-se um método que retorna a instância única por meio de um ponteiro estático. Caso seja a primeira vez que é chamado, a classe é instanciada, caso contrário, retorna a instância já existente. Além desta, o *InputManager* e a *Engine* são *singletons*. Isto acontece porque todas essas classes só devem possuir somente uma instância, pois não se deseja várias instâncias controlando a saída de vídeo, a entrada de dados ou a simulação de objetos.

Subsistema de *Input*

Neste subsistema, para dar suporte ao protocolo TUIO, tem-se a classe *InputManager*, derivada da classe *TuioListener*, que implementa métodos para escutar mensagens relativas à detecção de toques, gerando eventos. Estes eventos podem adicionar e remover

toques ou atualizar suas posições. Também suporta a detecção de eventos de input via mouse ou teclado. O método principal da classe é o *Update*, onde são percorridos todos os eventos enviados durante um *Game Loop*, cada um sendo analisado e seus dados separados para utilização das outras classes. Por exemplo, caso seja adicionado um toque, o número identificador deste toque é salvo, permitindo que outra classe possa utilizá-lo por meio do método *Is Touching*, de modo a descobrir se o toque com o identificador desejado está tocando ou não a superfície.

Subsistema da *Engine*

A *Engine* é composta pela classe *Engine*, além do pacotes de classes da biblioteca *Box2D*. Por intermédio destas classes, ocorre a criação dos objetos no mundo virtual. A simulação de gravidade, da força de atrito, e das forças de ação e reação, são calculadas pela *Engine*. Um importante recurso para uma boa simulação é a detecção de colisão, tratada por esse subsistema. Também são realizadas operações de destruição de objetos, além da localização dos objetos ser atualizada e repassada aos outros módulos pela *Engine*.

Subsistema de gerência de objetos

A classe *LevelState* é responsável pelo único estado do *software*. Estar no estado de *LevelState* significa que o *software* está em execução e o usuário pode interagir como desejar, criando objetos ou mudando propriedades. Nesta classe, todas as imagens que aparecem dentro do jogo são carregadas e renderizadas na tela. Seu método de *Update*, chamado uma vez pelo *Game Loop*, verifica todos os toques existentes ou removidos e decide o que cada um irá fazer, como por exemplo, criar um objeto novo, arrastar um objeto existente, abrir o menu ou alterar as opções. Seu outro método, *Render*, é responsável por montar a tela do jogo, renderizando na tela todas as imagens em sua sequência, deixando-as prontas para serem mostradas na tela, assim que tudo estiver em posição.

A *GameObject* é uma superclasse que agrupa subclasses referentes a todos objetos presentes *software*. Estas classes representam os objetos desenhados no programa que podem ser polígonos, círculos ou de forma livre. Além disso, objetos podem ser motores e juntas, explicadas no início deste capítulo. O próprio mundo do software é um objeto, pois podem existir vários mundos, e as próprias barreiras usadas para dividir os submundos também são consideradas objetos. Todos objetos possuem vértices que indicam suas posições e suas áreas. Todos os *GameObjects* possuem métodos de *Update* e *Render*.

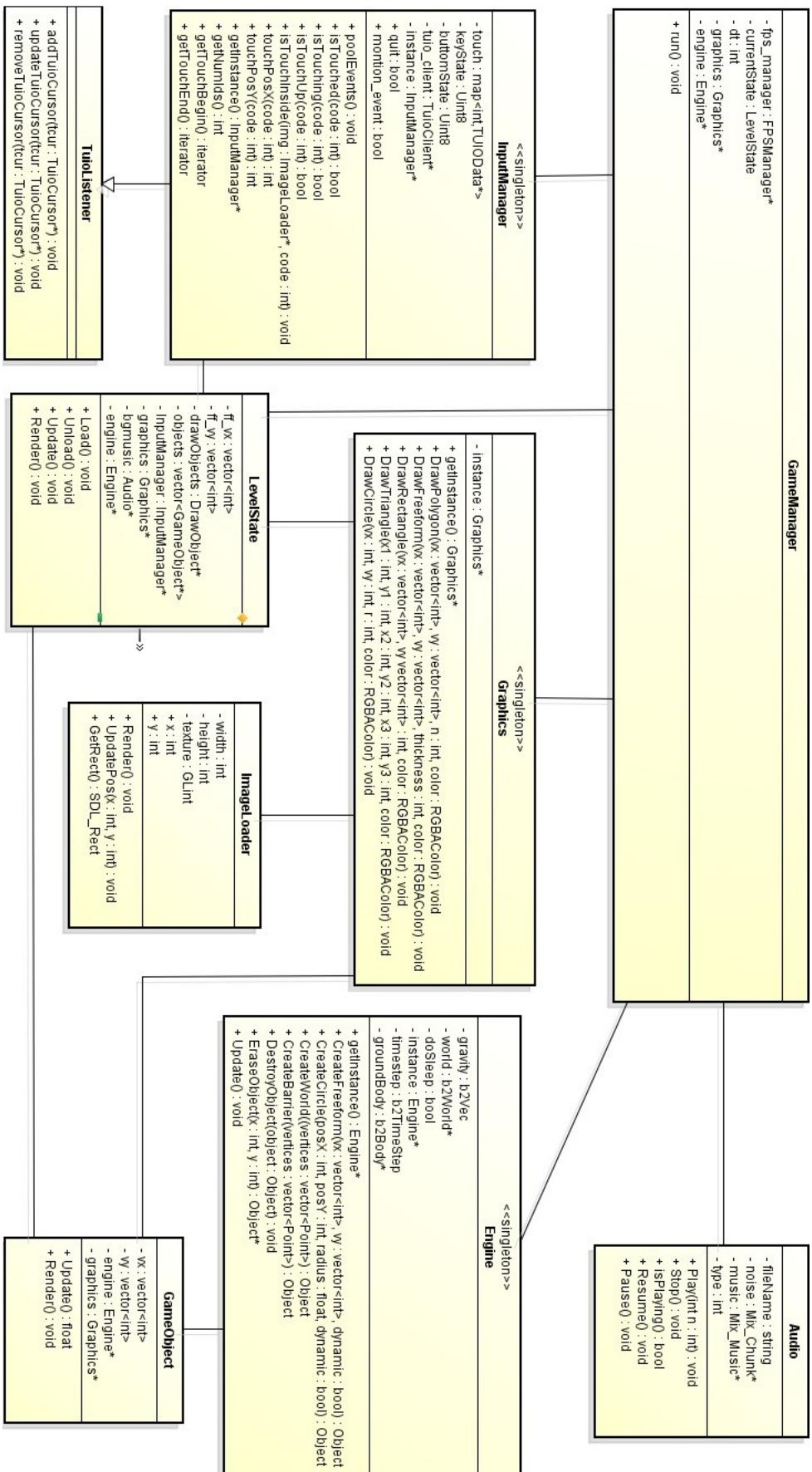


Figura 5.5: Diagrama de classes do *Deskworld*

Capítulo 6

Conclusão

Existem diversos tipos de *softwares* para mesas com superfície multitoque, no entanto, poucos são os que aproveitam todas as capacidades desta interface. Neste trabalho, foi desenvolvido um *software* simulador de física, que devido a suas características, consegue tirar proveito do input através de toques e gestos, além de incentivar seu uso por múltiplos usuários simultaneamente. Ao final, concluiu-se que a escolha do *software* desenvolvido foi adequada para o tipo de interface construída, pois a criação de mundos livres e independentes proporciona entretenimento para qualquer número de pessoas que consigam utilizar a mesa ao mesmo tempo, sem sentirem que estão atrapalhando umas as outras. Contudo, observou-se que determinadas decisões durante o desenvolvimento tiveram algumas consequências a serem destacadas a seguir e futuramente melhoradas.

Não foi modificada a engine utilizada, a *Box2D*, por falta de tempo para se analisar todo seu código a procura de pontos possíveis de se otimizar. Por causa disso, foram percebidos alguns problemas de performance que não puderam ser corrigidos. Por exemplo, ao se desenhar muitas formas livres, nota-se uma lentidão no *software*, pois esta forma acarreta muitas verificações por colisões. *Softwares* simuladores de física permitem a criação de qualquer formato de objeto, por isso a complexidade na detecção de colisões é um dos fatores mais agravantes.

Outro problema se encontra da aproximação de polígonos. Devido a alta imprecisão do toque para o desenho, as vezes a detecção de aproximação detecta lados onde não foram feitos e não detecta aonde foram. Seria necessário a criação de algorítmos ainda mais complexos para detectar essa aproximação e suportar erros, sem se influenciar o desempenho do *software*. A melhor maneira seria utilizar redes neurais como observado em [36] para se aproximar direto formas a gestos treinados, sem provavelmente trazer problemas de performance.

A utilização da engine *Box2D* na mesa construída não apresentou nenhum problema, somente a criação de objetos, que teve que ser tratada por algorítmos mais complexos que os tradicionais.

Em relação a própria construção da mesa, o projeto original teve que ser modificado algumas vezes. Alguns problemas surgiram da remontagem da mesa, que ao ser transportada a outra localidade sofreu deformidades e não foi possível ser remontada corretamente, deixando suas medidas levemente desiguais, o que afetou o posicionamento do acrílico na parte superior da mesa. Além disso, o tamanho total da tela de projeção teve que ser diminuído para acrescentar um suporte extra para o acrílico, pois o seu peso e comprimento provocaram uma inclinação em sua superfície. Por esta razão, o projetor teve que ser aproximado do topo da mesa, permitindo o correto enquadramento da imagem, o que alterou o *design* interior da mesa. Ao final da construção da mesa, no entanto, foi possível chegar a um projeto funcional e eficiente, que possibilita ajustes na calibração da posição dos componentes e, ainda assim, permite fixá-los no local desejado.

A utilização da mesa por um longo período de tempo resulta em um grande aquecimento interno, pois o projetor e o computador liberam grande quantidade de calor. Percebeu-se a necessidade de instalar um sistema de ventilação para resolver este problema, já que um ambiente quente pode reduzir a vida útil do projetor e do *hardware* do computador. A sensibilidade da mesa na detecção de toques é um pouco baixa, pois é necessário aplicar uma certa pressão sobre a superfície devido a espessura do acrílico ser um pouca alta. Porém, os toques são bem precisos, pois o local dos toques na mesa é reproduzido no *software* com boa aproximação da realidade.



A mesa obtida como um dos produtos finais possui um tamanho de tela adequado para a utilização de múltiplos usuários, porém dificulta o acesso por um único usuário para aproveitar toda sua extensão. Além disso, devido ao projeto inicial ser de uma tela que foi diminuída como explicado no parágrafo anterior, a mesa final ficou um pouco mais alta que o necessário, e sua altura pode tornar desconfortável o uso por pessoas de baixa estatura por um tempo prolongado. No entanto, basta adicionar um palanque ao redor da mesa que possibilite as pessoas a ficarem mais distantes da superfície, caso isso venha a ser um problema. Nos testes realizados, não houveram reclamações quanto a isto.

Numa análise geral dos produtos finais obtidos, observa-se que o *software* *Deskworld* possui uma ótima performance, sendo capaz de tratar criações de vários objetos intercaladas com a simulação de objetos já criados sem produzir lentidão ou instabilidades. A sensibilidade da mesa foi suficiente para se criar os desenhos desejados com uma boa aproximação. Os toques foram filtrados para somente considerar movimentos a partir de um determinado número de pixels de movimentação, pois a própria imprecisão da imagem podia acabar gerando mensagens de update indesejadas pelo *CCV*. O *software* foi desenvolvido na resolução padrão do projetor desta mesa, 1024x768, porém pode ser facilmente alterado para qualquer resolução dependendo da mesa disponível.

Além do *software* desenvolvido, a mesa pode rodar vários outros aplicativos que utilizem o protocolo *TUIO* para comunicação. Alguns exemplos são os explicados no Capítulo 2, mas ainda existe a possibilidade de se expandir ainda mais a sua utilização com programas para, por exemplo, auxiliar uma reunião possuindo blocos de notas distintos para cada usuário com captação de áudio, que interaja com objetos na superfície para transmitir uma ata para todos os presentes, ou então, caso a mesa seja usada em um bar ou casa noturna, um cardápio virtual junto de um aplicativo de *chat*, que permita bate-papo com outras mesas no ambiente, facilitando a integração de pessoas no local.

Como sugestões para trabalhos futuros, é possível melhorar o *design* da mesa adicionando iluminação *DI*, além da *FTIR* já existente, a fim de possibilitar o uso de marcadores fiduciais, ampliando as capacidades da mesa e interações possíveis. Além disso, o *Deskworld* pode ter seu desempenho e usabilidade melhoradas, a partir da otimização de seu código e da adição de novas ferramentas que utilizem marcadores fiduciais, como, por exemplo, associá-los a criação de objetos com formas específicas. Por último, a criação de novos recursos, que possibilitem a inclusão de regras no mundo do *Deskworld*, permitirá a geração de pequenos objetivos, como por exemplo, adicionar uma parede que, ao ser tocada por um objeto, produziria uma pontuação, dando origem a um jogo de futebol ou de *air hockey*, entre outros.

Referências

- [1] Algoryx. Phun. <http://www.phunland.com/wiki/Home>, acessado em 26/06/2010.
- [2] L. Ang, A. Sharma, C. Moore, and T. Bintahir. Community earth. <http://nuicode.com/projects/earth>, acessado em 03/02/2011.
- [3] M. Barcelona. Multitouch crayon physics. <http://blog.multitouch-barcelona.com/2008/05/multitouch-crayon-physics-is-available.html>, acessado em 26/06/2010.
- [4] B. Bors. The role of the mouse in audio games. <http://www.game-accessibility.com/index.php?pagefile=roleMouseAudioGames>, acessado em 26/06/2010.
- [5] P. G. Brandão and S. C. R. Braga. Construção de um jogo eletrônico multiusuário em uma superfície de projeção multitoque. Trabalho de graduação, Universidade de Brasília, Departamento de Ciência da Computação, Brasília, July 2009.
- [6] S. F. Brown. How it works: Multi-touch surfaces explained. *Scientific American*, June 2008. Supplement to the feature "Hands On Computing: How Multi-touch Screens Could Change The Way We Interact With Computers and Each Other", printed in the July 2008 issue of Scientific American. <http://www.scientificamerican.com/article.cfm?id=how-it-works-touch-surfaces-explained>, acessado em 26/06/2010.
- [7] E. Catto. *Box2D v2.1.0 User Manual*, 2007-2010. Disponível em <http://www.box2d.org/manual.html>.
- [8] N. G. Community. Community core vision. <http://ccv.nuigroup.com/>, acessado em 01/02/2011.
- [9] H. Computers. Benq mp772 st crni - hellas computers. <http://www.hellas.rs/index.php?action=opis&q=8144>, acessado em 02/02/2011.
- [10] T. D. Different. Ten great rivalries. <http://www.becks.com/2009/02/06/ten-great-rivalries/>, acessado em 26/06/2010.
- [11] T. Edmonds. Numpty physics. <http://numptyphysics.garage.maemo.org/>, acessado em 26/06/2010.
- [12] S. C. Entertainment. Playstation 3. <http://us.playstation.com/ps3/>, acessado em 26/06/2010.

- [13] S. C. Entertainment. Playstation portable. <http://us.playstation.com/psp/index.htm>, acessado em 26/06/2010.
- [14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley, Boston, 1995.
- [15] K. Group. OpenGL - the industry's standard for high performance graphics. <http://www.opengl.org/>, acessado em 07/02/2011.
- [16] M. Guy and D. Watcher. The video game console library. <http://www.videogameconsolelibrary.com/pg70-odyssey.htm>, acessado em 02/02/2010.
- [17] H. Hartman. Multiplayer games and physics on multi-touch screen devices. Bachelor's thesis, Luleå University of Technology, Departament of Skellefteå Campus, 2008.
- [18] G. A. T. Holt, M.J.T Reinders, and E. Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In *Thirteenth annual conference of the Advanced School for Computing and Imaging*, June 13-15 2007.
- [19] P. J. Hruschak. Wiimote vs. move vs. kinect: Comparing control schemes in the three-way battle for motion control. <http://www.gamertell.com/gaming/comment/wiimote-playstation-move-kinect-motion-controller-comparison/>, acessado em 26/06/2010.
- [20] Ideum. Ideum 100" multitouch table supports 50 users. <http://hothardware.com/News/Ideums-100-MultiTouch-Table-Supports-50-Users-Air-Hockey-Anyone/>, acessado em 29/01/2011.
- [21] Izanagi. Game locker: Ps3 vs xbox 360 vs wii. <http://techknowbabel.com/2010/02/27/game-locker-ps3-vs-xbox-360-vs-wii/>, acessado em 26/06/2010.
- [22] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Constanza. Tuio: A protocol for table-top tangible user interfaces. In *6th International Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.
- [23] R. Khaled, P. Barr, H. Johnston, and R. Biddle. Let's clean up this mess: exploring multi-touch collaborative play. In *CHI '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4441–4446, New York, NY, USA, 2009. ACM.
- [24] P. King. Tuio/osc smoke demo. <http://projects.edencomputing.com/projects/tuiosmoke>, acessado em 03/02/2011.
- [25] Kloonigames. Crayon physics. <http://www.crayonphysics.com/>, acessado em 26/06/2010.
- [26] S. Lantinga. Simple directmedia layer. <http://www.libsdl.org/>, acessado em 07/02/2011.
- [27] S. Lantinga, S. Peter, and R. Gordon. Sdl mixer. http://www.libsdl.org/projects/SDL_mixer/, acessado em 07/02/2011.

- [28] G. J. Lepinski, T. Grossman, and G. Fitzmaurice. The design and evaluation of multitouch marking menus. *ACM CHI 2010: Multitouch*, Abril 2010.
- [29] J. Luderschmidt. The multi-touch virttable. <http://johannesluderschmidt.de/lang/en-us/the-multi-touch-table-virttable/153/>, acessado em 26/06/2010.
- [30] T. Magix. Touch magix. <http://www.touchmagix.com/indexEnquiry.html?gclid=CKbJgvLV4qYCFa5k7AodSUDn2g>, acessado em 29/01/2011.
- [31] Microsoft. Microsoft surface. <http://www.microsoft.com/surface>, acessado em 26/06/2010.
- [32] M. Molecule. Little big planet. <http://www.littlebigplanet.com/>, acessado em 26/06/2010, October 2008.
- [33] M. Molecule. Little big planet 2. <http://www.littlebigplanet.com/en-us/2/>, acessado em 26/06/2010, November 2010.
- [34] M. Molecule. Modnation racer. <http://www.modnation.com/index/>, acessado em 26/06/2010, May 2010.
- [35] L. Y. L. Muller. Multi-touch displays: design, applications and performance evaluation.
- [36] E. S. Nygård. Multi-touch interaction with gesture recognition. Master's thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, 2010.
- [37] J. Peatross and M. Ware. *Physics of Light and Optics*. Brigham Young University, 2008.
- [38] T. Perl and S. Kögl. Adaptation and evaluation of numpy physics for multi-touch multi-player interaction. Bachelor's thesis, Vienna University of Technology, Institute of Computerized Automation, Vienna, August 2009.
- [39] S. Rabin, editor. *Introduction to game development*. Charles River Media, 2005.
- [40] Reactable. <http://www.reactable.com/>, acessado em 26/06/2010.
- [41] S. Sandler. Community core vision (ccv) - calibration. <http://sethsandler.com/multitouch/community-core-vision-calibration/>, acessado em 11/02/2011.
- [42] S. Sandler. Mtmini - diy multitouch mini pad. <http://sethsandler.com/multitouch/mtmini/>, acessado em 26/06/2010, 2008.
- [43] W. S. Schneider, N. C. Dias F., L. H. M. Mauruto, and F. R. Miranda. Irtaktiks: Jogo de estratégia para mesa multitoque. *SBGames 2008, Belo Horizonte*, November 2008.
- [44] SEBRAE. Brasil e china puxarão crescimento na área de entretenimento até 2014. http://www.sebrae-sc.com.br/novos_destaque/oportunidade/default.asp?materia=18891, acessado em 26/06/2010.

- [45] T. Streeter. The verve project. <http://verveproject.blogspot.com/2008/01/multitouch-game-of-life.html>, acessado em 26/06/2010, January 2008.
- [46] T. Teabag. The world's first videogames. <http://www.teamteabag.com/2008/05/17/retro-computing-corner-the-worlds-first-videogames/>, acessado em 26/06/2010.
- [47] Terra. Conheça a história da atari. <http://games.terra.com.br/interna/0,0I1966183-EI1702,00.html>, acessado em 26/06/2010.
- [48] M. Thörnlund. Gesture analyzing for multi-touch screen interfaces. Bachelor's thesis, Luleå University of Technology, Departament of Skellefteå Campus, 2007.
- [49] Wikipédia. Playstation 2 - wikipédia, a enciclopédia livre. http://pt.wikipedia.org/wiki/PlayStation_2, acessado em 03/02/2011.
- [50] G. Works. Established multitouch gesture support. <http://gestureworks.com/about/supported-gestures/>, acessado em 26/06/2010.