FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Framework para Estação Multitoque de Grandes Dimensões

Pedro Miguel dos Santos Silva Pinto

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof. Doutor Rui Pedro Amaral Rodrigues

13 de Julho de 2011

Framework para Estação Multitoque de Grandes Dimensões

Pedro Miguel dos Santos Silva Pinto

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Prof. Doutor Jorge Manuel Gomes Barbosa

Vogal Externo: Prof. Doutor Paulo Miguel de Jesus Dias

Orientador: Prof. Doutor Rui Pedro Amaral Rodrigues

13 de Julho de 2011

Resumo

Este documento é realizado no âmbito da dissertação do Mestrado Integrado em Engenharia Informática e Computação, sendo o tema, Framework para Estação Multitoque de Grandes Dimensões.

A informação presente no documento foi recolhida e compilada ao longo do período dedicado à investigação da área das tecnologias de multitoque, englobando o estado da arte atual sob o ponto de vista das técnicas para captação de toques, das tecnologias de processamento de toques, e dos dispositivos eletrónicos que tiram partido deste tipo de interação. Neste relatório também é documentado o desenvolvimento de uma *framework* para facilitar a criação de aplicações multitoque, dando ênfase às características incluídas nesta *framework* e à aplicação de teste desenvolvida.

Durante a realização deste projeto foi montada uma estação multitoque para implementação e teste da *framework*, que se encontra no laboratório I123 da Faculdade de Engenharia da Universidade do Porto (FEUP). A tecnologia adotada para a captação de gestos foi a FTIR (Frustrated Total Internal Reflection). A *framework* produzida poderá ser utilizada para futuros projetos no âmbito da investigação e desenvolvimento de novas interfaces que utilizem toques.

Abstract

This document is written in the scope of dissertation of Integrated Master in Software Engineering and Computation, entitled, Framework for Multitouch Station for Large Dimensions.

The information in this document was collected and compiled during the time dedicated for investigation in multitouch technology, with a special focus in the state of the art in touches capture, technology of touch processing, and electronic devices that took advantages of this kind of interaction. In this report, is also included the documentation of the development of a framework to help create multitouch applications, emphasizing its characteristics and the test application developed.

During this project, a multitouch station was built to implement and test the framework, and is currently in the laboratory I123 in Faculdade de Engenharia da Universidade do Porto (FEUP). The technology adopted to capture gestures was FTIR (Frustrated Total Internal Reflection). The produced framework will be used in future projects of investigation and development of new interfaces using touches.

Agradecimentos

Em primeiro lugar gostaria de agradecer ao professor Rui Rodrigues pelo extraordinário apoio prestado durante toda esta dissertação.

Agradeço a todos as pessoas com quem contactei na FEUP, incluindo colegas, docentes e funcionários da faculdade. Seguramente que não me esquecerei de ninguém.

Pedro Miguel dos Santos Silva Pinto

Índice

1 Introdução 1
1.1 Conceitos 2
1.2 Motivação 3
1.3 Objetivos 3
1.4 Estrutura do Documento 4
2 *Hardware* e *Software* Multitoque 5
2.1 O Início do Multitoque 5
2.2 Hardware 7
2.2.1 Tecnologia Resistiva 8
2.2.2 Tecnologia Capacitiva 8
2.2.3 FTIR 9
2.2.4 LLP 10
2.2.5 Comparação entre as diferentes tecnologias 11
2.3 Dispositivos 12
2.3.1 iPhone/iPod/iPad 13
2.3.2 Microsoft Surface 13
2.3.3 Second Light 14
2.3.4 Side Sight 15
2.3.5 Reactable 15
2.3.6 Perceptive Pixel 16
2.3.7 Edigma 17
2.4 Software 17
2.4.1 Camada de *input* 18
2.4.2 Camada de *middleware* 21
2.4.3 Camada de aplicação 21
2.5 Perspetiva final sobre o multitoque 23
3 *Framework* Multitoque para Manipulação de Informação 25
3.1 Especificação da *Framework* 27
3.1.1 *Factories* 28
3.1.2 *Widgets* 29
3.2 Detalhes de Desenvolvimento 32

3.2.1 Extração da informação 32
3.2.2 Gestos 32
3.2.3 Criação de *widgets* 34
3.3 Caso de Estudo – SiFEUP MT 35
3.3.1 *Widget* de Autenticação 39
3.3.2 *Widget* de Aluno 39
3.3.3 *Widget* de Horário 40
3.3.4 *Widget* de Curso 41
3.3.5 *Widget* de Percurso Académico 41
3.3.6 *Widget* de Disciplina 42
3.3.7 *Widget* de Docente 43
3.3.8 *Widget* de Lista de Alunos 44
4 Estudo de Aplicabilidade e Testes 47
4.1 Estudo de aplicabilidade do SiFEUP MT 47
4.2 Testes 49
5 Conclusões 53
5.1 Apreciação do Trabalho Realizado 53
5.2 Trabalho Futuro 54
Referências 57
Anexo A - Instalação do Multitoque em Windows 7 61
Anexo B - Instalação do Multitoque em Linux 63
Anexo C - Estudo das Tecnologias 65
Anexo D - Preparação do Hardware/Software 69
Anexo E - Exemplos de Utilização da *Framework* 73
Anexo F - API 77

Lista de Figuras

Figura 1 – O primeiro rato era feito em madeira. 6
Figura 2 – Arquitetura genérica de um sistema multitoque. 7
Figura 3 – Funcionamento de uma superfície com tecnologia resistiva. 8
Figura 4 – Funcionamento de uma superfície com tecnologia capacitiva. 9
Figura 5 – Funcionamento da técnica FTIR. 9
Figura 6 – Imagem iluminada por luz IR. 10
Figura 7 – Funcionamento de uma superfície de tecnologia *laser*. 10
Figura 8 – Efeito de oclusão na técnica LLP. 11
Figura 9 – Previsões para o mercado das tecnologias MT. 12
Figura 10 – Apresentação do iPad. 13
Figura 11 – Microsoft Surface. 14
Figura 12 – Second Light. 14
Figura 13 – Side Sight. 15
Figura 14 – Reactable. 16
Figura 15 – Perceptive Pixel. 16
Figura 16 – Utilização da tecnologia Displax num shopping. 17
Figura 17 – Arquitetura de *software* de um sistema MT. 18
Figura 18 – Touchlib. 19
Figura 19 – *Screenshot* do CCV em execução. 20
Figura 20 – Reactivision. 20
Figura 21 – Exemplo da arquitetura TUIO. 21
Figura 22 – Exemplos de gestos. 26
Figura 23 – Arquitetura da *framework*. 28
Figura 24 – Exemplo de utilização do *Beautiful Soup*. 29
Figura 25 – Exemplo da criação do protótipo de um gesto. 33
Figura 26 – *Screenshot* da página principal do SiFEUP. 36
Figura 27 – Arquitetura da aplicação. 36
Figura 28 – Ecrã inicial do SiFEUP MT. 37
Figura 29 – Autenticação no SiFEUP MT. 37
Figura 30 – Área pessoal do aluno. 38
Figura 31 – SiFEUP MT. 38
Figura 32 – Funcionamento da *widget* de autenticação. 39
Figura 33 – Aspeto da *widget* de autenticação. 39
Figura 34 – Funcionamento da *widget* de aluno. 39
Figura 35 – Aspeto da *widget* de aluno. 40
Figura 36 – Funcionamento da *widget* de horário. 40
Figura 37 – Aspeto da *widget* de horário. 40

Figura 38 – Funcionamento da *widget* de curso. 41
Figura 39 – Aspeto da *widget* de curso. 41
Figura 40 – Funcionamento da *widget* de percurso académico. 41
Figura 41 – Aspeto da *widget* de percurso académico. 42
Figura 42 – Funcionamento da *widget* de disciplina. 42
Figura 43 – Aspeto da *widget* de disciplina. 43
Figura 44 – Funcionamento da *widget* de docente 43
Figura 45 – Aspeto da *widget* de docente 43
Figura 46 – Funcionamento da *widget* de lista de alunos 44
Figura 47 – Aspeto da *widget* lista de alunos. 44
Figura 48 – Ferramenta de calibração do CCV. 70
Figura 49 – *Screenshot* do CCV. 71
Figura 50 – Programação de aplicações multitoque com o PyMT. 72

Lista de Tabelas

Tabela 1 – Comparativo entre tecnologias para captação de toques. 11
Tabela 2 – Resultados do questionário à comunidade FEUP. 48
Tabela 3 – Resultado do questionário aos utilizadores após os testes. 49
Tabela 4 – Resultado das medições de tempo nos três casos de utilização. 50

Abreviaturas e Símbolos

API	*application programming interface*
FEUP	*faculdade de engenharia da universidade do porto*
FTIR	*frustrated total internal reflection*
GUI	*graphical user interface*
HCI	*human computer interfaces*
IV	*infravermelha*
LED	*light emitting diode*
MPX	*multi-pointer x*
MT	*multitoque* ou *multitouch*
NUI	*natural user interface*
OSC	*open sound control*
SiFEUP	*Sistema de Informação da FEUP*
TUI	*tangible user interfaces*
TUIO	*tangible user interface objects*

Capítulo 1

Introdução

O desenvolvimento das tecnologias de informação e a vulgarização de dispositivos como computadores ou telemóveis, atingiram níveis impensáveis até há alguns anos atrás. De tal modo que hoje em dia, pessoas de praticamente todas as camadas sociais e de diversos contextos educacionais e civilizacionais, usufruem das inúmeras potencialidades oferecidas por estes equipamentos.

A massificação da utilização destes equipamentos exige que os criadores de software coloquem um maior esforço na interação homem-máquina, na espectativa do desenvolvimento de novos tipos de interfaces e, ainda assim, manter o compromisso da usabilidade.

Fazendo uma comparação com aquilo que acontecia há apenas duas décadas atrás, o público consumidor de equipamentos informáticos é hoje muito superior em quantidade de utilizadores do que era, e já não inclui só utilizadores com grandes conhecimentos informáticos. Isto obriga a que a componente visual das interfaces homem-máquina seja cada vez mais importante, para não condenar alguns à infoexclusão. Em última análise, beneficia todo o tipo de utilizadores.

Há alguns anos atrás, Bill Gates confessou que tinha o sonho de ver um computador em todas as secretárias. Mais recentemente, disse que gostaria que todas as secretárias fossem um computador. Esta frase pode ser vista como um indicador do rumo da Microsoft para os próximos anos, mas também como um indicador para um novo paradigma na indústria informática, o multitoque (MT).

> *Multitoque é a capacidade de processar múltiplos pontos de contacto ao mesmo tempo, possibilitando a utilização das mãos para efetuar gestos, para assim interagir com os computadores de uma forma mais natural.* [ELI08]

Acima de tudo existe uma mudança de mentalidade por parte dos criadores de *software*, que agora anseiam por proporcionar melhores experiências aos utilizadores, e ao mesmo tempo, alargar o leque de possibilidades para a criação de aplicações.

No centro desta revolução tecnológica, procura-se criar ferramentas que facilitem o desenvolvimento rápido e eficiente de aplicações MT. Essa busca por melhores ferramentas passa obrigatoriamente pelo desenvolvimento de *frameworks* que englobem um conjunto de funcionalidades adequadas aos objetivos de uma aplicação deste tipo. É aí que se insere o tema desta dissertação, Framework para Estação Multitoque de Grandes Dimensões, onde se pretende implementar uma *framework*, que permita o desenvolvimento de aplicações MT, respondendo às necessidades das mesmas, identificadas ao longo do estudo realizado sobre esta área tecnológica.

1.1 Conceitos

Os dispositivos e aplicações que tiram partido de tecnologias MT são cada vez mais vulgares no nosso quotidiano. Ainda assim convém introduzir alguns conceitos para uma melhor compreensão dos conteúdos que se seguem.

Quando nos referimos a uma superfície MT, falamos de um ecrã que apresenta imagens tal e qual como um computador, mas que por especificidades do seu processo de montagem e do *software* instalado, permite uma interação por meio de toques simultâneos dos dedos. Um sistema deste tipo diz-se possuir uma interface natural ou *natural user interface* (NUI). Também podem ser utilizados objetos para a interação, mas neste caso diz-se que o sistema possui uma interface tangível *ou tangible user interface* (TUI).

Nem todos os ecrãs destes equipamentos são iguais no que toca à tecnologia de captação de toques. Algumas dessas tecnologias são apresentadas neste documento, bem como as vantagens associadas a cada uma delas.

Ao longo deste documento são referidas várias vezes termos como *blob*, *track* ou *motion-tracking*, que são termos utilizados para identificar algumas entidades do processamento MT. Em tecnologias baseadas em visão por luz infravermelha (IV), os toques numa superfície são identificados pela perturbação que causam na sua reflexão. A reflexão pode ser vista através de câmaras IR, que conseguem captar a luz desviada na ponta dos dedos. A esses pontos dá-se o nome de *blob*. Um *track* é o efeito de deslocamento de um *blob*, ou seja, corresponde à situação de um *blob* a mover-se, que não é mais do que o movimento de um dedo. O *motion-tracking* é o processo que localiza ao longo do tempo a deslocação dos *blobs*, dando origem às *tracks*.

Não existe uma escala que defina os limites que separam superfícies MT de pequena, média e grande dimensão. No contexto desta tese serão consideradas como superfícies de pequena dimensão as que fazem parte de *smartphones* e *tablets*, normalmente destinadas a um único utilizador com uso simultâneo de um número reduzido de dedos. Consideraremos as superfícies de média dimensão as contempladas em monitores de secretária e que podem ir até ao limiar de monitores de 32", tipicamente usadas por um utilizador (apesar de ser viável um número superior) mas com um número elevado de dedos. Como superfícies de grandes dimensões, consideraremos dimensões superiores a 32 polegadas, que permitam facilmente a utilização simultânea por mais do que um utilizador.

1.2 Motivação

Os dispositivos MT estão cada vez mais presentes na nossa vida, nomeadamente, *smartphones*, mas rapidamente foram encontradas outras vantagens na sua utilização. Ao mesmo tempo em que o mercado dá sinais que ainda tem espaço para crescer, alguns gigantes da produção de *software* e também no meio académico, veem no MT um paradigma da interação homem-máquina com um potencial de investigação tremendo [Lee11] [SKO09].
Esta procura por soluções mais interativas e naturais exige uma resposta adequada por parte de quem desenvolve *software*. Essa resposta pode ser ajustada se forem fornecidas ferramentas adequadas aos programadores.

Para permitir que cada programador possa produzir melhores aplicações, seria interessante haver uma ferramenta que facilitasse essa tarefa, criando uma abstração dos detalhes complexos inerentes à implementação de software. Uma abstração independente do sistema operativo, e mais independente dos conhecimentos de programação, para que assim se possa dedicar mais tempo à qualidade do produto final.

Estas premissas motivam a investigação nesta área, com vista à criação de mais e melhores soluções.

1.3 Objetivos

Um dos objetivos desta dissertação é efetuar um estudo aprofundado sobre o estado da arte das tecnologias MT. Este estudo passa por identificar e classificar as tecnologias mais utilizadas para captação de toques numa superfície, as ferramentas de *software* mais utilizadas no processamento dessa informação, e identificar exemplos de aplicações possíveis desta tecnologia. Esta informação permite atingir um nível de conhecimento suficiente para poder identificar as melhores ferramentas para o desenvolvimento de aplicações, e para identificar o que se espera de uma aplicação MT.

Esta base de conhecimento serve de suporte à implementação de uma *framework*, que facilite o desenvolvimento de aplicações interativas para estações MT de médias e grandes dimensões. A *framework* desenvolvida deve permitir a produção de aplicações para acesso ao Sistema de Informação da FEUP (SiFEUP), através de interfaces orientadas ao paradigma de interação natural, possíveis com o multitoque.

Esta *framework* deve auxiliar o trabalho do programador, guiando a implementação, e fornecendo abstrações para detalhes que se possam repetir no tipo de aplicações deste âmbito. Detalhes que se prendem fundamentalmente com a obtenção de informação do SiFEUP, e com a apresentação dessa informação ao utilizador. Durante o desenvolvimento devem ser identificadas outras tarefas que possam ser automatizadas pela *framework*. Por fim, também deveria ser disponibilizado um conjunto de utilitários, que pudessem ser integrados nas aplicações. Isto não só aumenta as funcionalidades da *framework*, como melhora a eficiência do desenvolvimento de aplicações.

No final, deve ser avaliado o potencial da *framework* para desenhar aplicações com conteúdos provenientes do SiFEUP, e para servir de base à investigação de novos paradigmas de interação. Como complemento, deverá ser feito um estudo da aplicabilidade de uma aplicação deste tipo nas instalações da FEUP, através da realização de inquéritos.

1.4 Estrutura do Documento

No segundo capítulo deste documento é apresentado o estado da arte das tecnologias MT. A ênfase é colocada nas tecnologias de captação de toques, nos dispositivos que mais eficazmente tiram partido da tecnologia MT, e nas plataformas mais utilizadas para a produção de aplicações.

No terceiro capítulo introduz-se o conceito de *framework* num contexto relacionado, e é feita uma descrição dos objetivos esperados para implementação da *framework*, e para as aplicações a serem desenvolvidas através desta plataforma.

No quarto capítulo faz-se a especificação da *framework*, e todos os detalhes relevantes da implementação. Por fim, é apresentada a aplicação de teste desenvolvida para demonstração da utilidade da *framework*.

No quinto capítulo estão documentados os testes efetuados sobre a aplicação de teste, para determinar a sua utilidade, e são apresentados os resultados de um estudo realizado junto da comunidade universitária da FEUP.

No sexto capítulo são retiradas conclusões do projeto, nomeadamente, a apreciação do trabalho realizado na perspetiva das mais-valias obtidas com a utilização da *framework* e de um hipotético desenvolvimento futuro do projeto.

Capítulo 2

Hardware e *Software* Multitoque

Neste capítulo é feita uma breve introdução à tecnologia multitoque e à forma como se estabeleceu, ao ponto de se tornar numas das tecnologias atuais mais inovadoras.

De seguida, é apresentado o estado da arte da tecnologia multitoque ao nível do *hardware* para captação dos toques, e é feita a identificação de alguns dos dispositivos que melhor aproveitam as vantagens desta tecnologia. Depois é explicada a arquitetura geral do *software* que processa toda a informação produzida num sistema multitoque, e são descritas algumas ferramentas que permitem desenvolver aplicações MT.

Por fim, são enunciadas algumas das aplicações e potencialidades desta tecnologia, que podem ser implementadas no nosso quotidiano.

2.1 O Início do Multitoque

As tecnologias MT têm vindo a crescer imenso nos últimos anos, apesar de não representarem uma área tecnológica totalmente nova.

Os primeiros estudos datam do início dos anos 80, perfazendo cerca de 25 anos até um verdadeiro aproveitamento das suas potencialidades. Pode parecer muito tempo num setor em constante evolução como o das tecnologias de informação, mas pondo em perspetiva com a história do rato, este necessitou de 30 anos para ser utilizado em massa, entre os primeiros protótipos em 1965, até por volta de 1995 com o lançamento do Windows 95 [Bux09]. Foi de facto um longo período de maturação da tecnologia até ser utilizado pela maioria dos utilizadores.

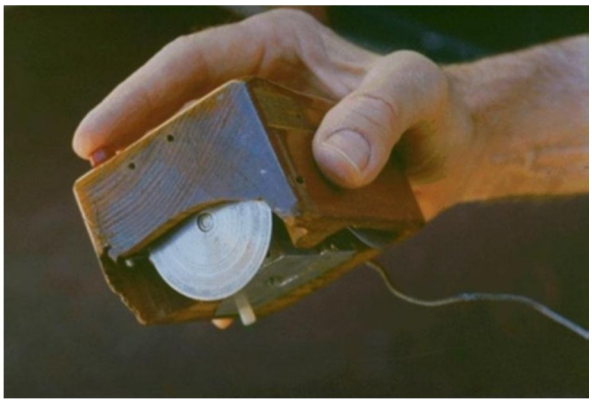


Figura 1 – O primeiro rato era feito em madeira.

Na tecnologia MT também é possível encontrar um momento semelhante, que coincide com o lançamento do iPhone em 2007, cujo grande trunfo estava no excelente aproveitamento da tecnologia, que oferecia uma interação intuitiva e proporcionava uma experiência totalmente direcionada para o utilizador final. Este fator abalou o mercado dos *smartphones*, fazendo crescer o interesse por esta tecnologia.

Mas nem só de *smartphones* se faz a história do MT. A aposta da Microsoft, ao incluir compatibilidade de raiz com MT no sistema operativo Windows 7, é uma aposta clara do gigante de *software* no sentido de promover a utilização de uma tecnologia emergente.

É possível identificar o momento em que a tecnologia MT ressurgiu depois de vários anos sem grandes desenvolvimentos. Esse momento aconteceu em fevereiro de 2006 na conferência TED (Technology Entertainment Design), durante a apresentação de Jeff Han, da *New York University* (NYU), que demonstrou o potencial deste tipo de tecnologia para o desenvolvimento das interfaces humano computador (HCI). Jeff Han aproveita o sucesso, e acabaria por fundar a empresa Perceptive Pixel para poder continuar com a investigação nesta área.

Desde então, o tema multitoque despertou um grande interesse nos internautas, como está provado no *Google trends*, que regista um grande aumento de pesquisas envolvendo a palavra "multi-touch" [SKO09], a partir dessa data.

Podemos identificar uma série de vantagens associadas à utilização desta tecnologia:

- Interação intuitiva devido à utilização de movimentos naturais das mãos e dedos.
- Em dispositivos de pequena dimensão dispensam-se teclas, aproveitando-se melhor o espaço disponível.
- Em dispositivos de grande dimensão, mais pontos de contacto significam a possibilidade de ter várias pessoas a trabalhar em conjunto num único ecrã.
- Manipulação de grandes volumes de informação.
- Rapidez na manipulação de vários objetos.
- Manipulação direta.
- O surgimento de uma nova gama de interações homem-máquina, até agora oculta nas limitações do rato e teclado.

2.2 Hardware

Para que a utilização desta tecnologia seja possível, é necessário o devido suporte em *hardware*. Existem várias soluções de estações MT disponíveis no mercado. Contudo, o funcionamento geral desse tipo de soluções não foge muito à regra. A Fig.2 ilustra esse funcionamento, com um alto nível de abstração.

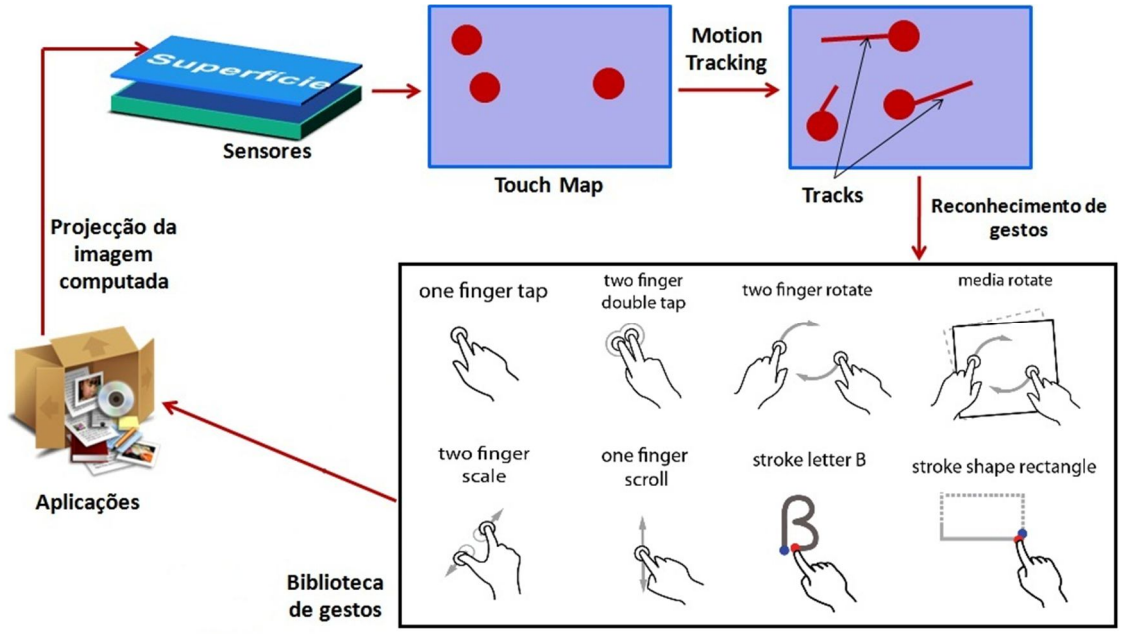


Figura 2 – Arquitetura genérica de um sistema multitoque.

A superfície interativa capta os *blobs* para formar um *touch-map*. O processo de *motion-tracking* trata de "seguir" o deslocamento dos *blobs*, de forma a obter um conjunto de *tracks*. Essa informação é depois comparada com uma biblioteca de gestos prototipados [GeW10]. Caso seja detetado um gesto conhecido, é gerado um evento que será integrado na aplicação. Essa informação é processada, dando origem à imagem projetada na superfície.

Convém ter em atenção que existem diferentes técnicas para captar toques numa superfície, que levam a algumas variações a este modelo, sendo duas delas:

- **Tecnologia ótica –** São relativamente fáceis de implementar, pois basta utilizar materiais relativamente acessíveis, como um projetor para projetar a imagem na superfície, uma tela para "reter" essa projeção, e uma câmara que capte a luz IV.
- **Tecnologia elétrica –** São mais difíceis de implementar. Os materiais não são tão comuns como os necessários para técnicas de tecnologia ótica, mas por outro lado, são mais utilizados em equipamentos de pequenas dimensões porque permitem montagens mais compactas.

De seguida, serão exemplificados alguns exemplos baseados nestas técnicas.

2.2.1 Tecnologia Resistiva

As superfícies que utilizam a tecnologia resistiva são compostas por um painel de vidro, e por duas camadas de um metal condutor, separadas por um pequeno espaço, como é demostrado na Fig.3. Quando o monitor está ligado, corre uma corrente elétrica através dessas duas camadas. O toque de um dedo na superfície exterior origina o contacto entre as duas camadas condutoras, com uma consequente descarga elétrica entre elas. Esta descarga provoca uma divisão de tensão entre os dois condutores nesse ponto, obtendo-se as coordenadas para o *touch-map* [Ove07].

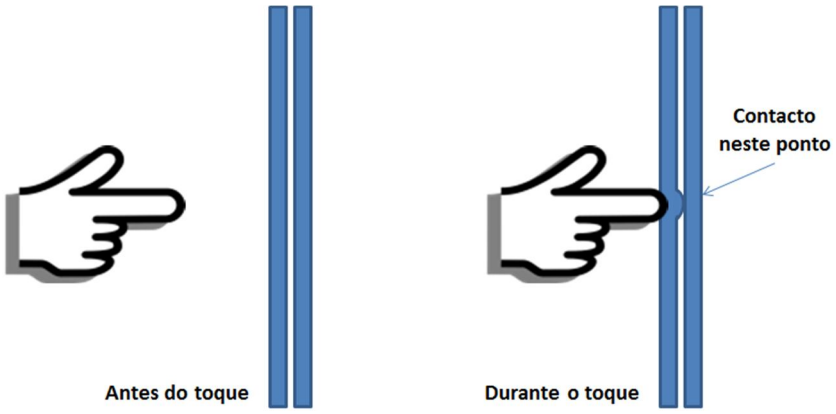


Figura 3 – Funcionamento de uma superfície com tecnologia resistiva.

Como a superfície responde à pressão dos toques, o contacto pode ser feito com os dedos ou qualquer outro dispositivo que funcione como um apontador. É uma técnica relativamente barata, mas que possui algum desgaste para os materiais necessitando de recalibração.

2.2.2 Tecnologia Capacitiva

Este é o tipo de tecnologia utilizado no ecrã do iPhone. Como é representado na Fig.4, no painel de vidro é colocada uma camada que armazena uma carga elétrica. Quando um dedo entra em contacto com o ecrã, no ponto desse toque, é originada uma pequena descarga elétrica na camada capacitiva, que é calculada pela diferença relativa de carga, e dessa forma são calculadas as coordenadas (x, y) do ponto de contacto [AlR08]. Isto funciona porque os humanos são condutores de eletricidade, mas deixa de acontecer se, por exemplo, o utilizador estiver a utilizar luvas ou *stylus*.

Uma vantagem desta tecnologia relativamente à tecnologia resistiva é, que esta permite a passagem de cerca de 90% da luz emitida pela projeção, contra os cerca de 75% permitidos pela tecnologia resistiva. Portanto, a perda de luminosidade original da projeção é maior numa superfície com tecnologia resistiva do que uma projeção numa superfície com tecnologia capacitiva, e isto acontece porque na tecnologia resistiva são utilizadas mais camadas para a captação dos toques. Outra vantagem desta tecnologia relativamente à tecnologia resistiva é a sua maior durabilidade.

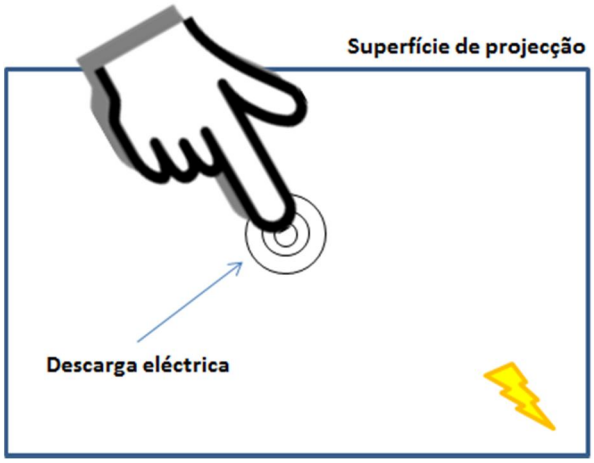


Figura 4 – Funcionamento de uma superfície com tecnologia capacitiva.

2.2.3 FTIR

A *frustrated total internal reflection* (FTIR) baseia-se num fenómeno ótico, que ocorre quando um raio de luz atravessa um material vindo de outro material, com um índice de refração superior e um ângulo de incidência maior. Nestas condições, não há refração desse raio, e a luz é totalmente refletida dentro da superfície, ficando "aprisionada". Esse efeito está representado na Fig.5, onde a reflexão interna da luz é exemplificada pelas linhas vermelhas no interior da superfície de interação.

A aplicação da técnica FTIR é simples. Nas bordas da placa de acrílico (plexiglas), são colocados LED's infravermelhos. Essa luz fica retida dentro do acrílico devido à reflexão interna. Quando um dedo toca na placa, a reflexão interna atinge-o, originando uma mancha luminosa nesse ponto, resultante da reflexão da luz. Esta mancha é captada por uma câmara de IR, colocada debaixo da superfície [AKA08]. Na Fig.5 podemos observar a reflexão interna, representada por linhas vermelhas.

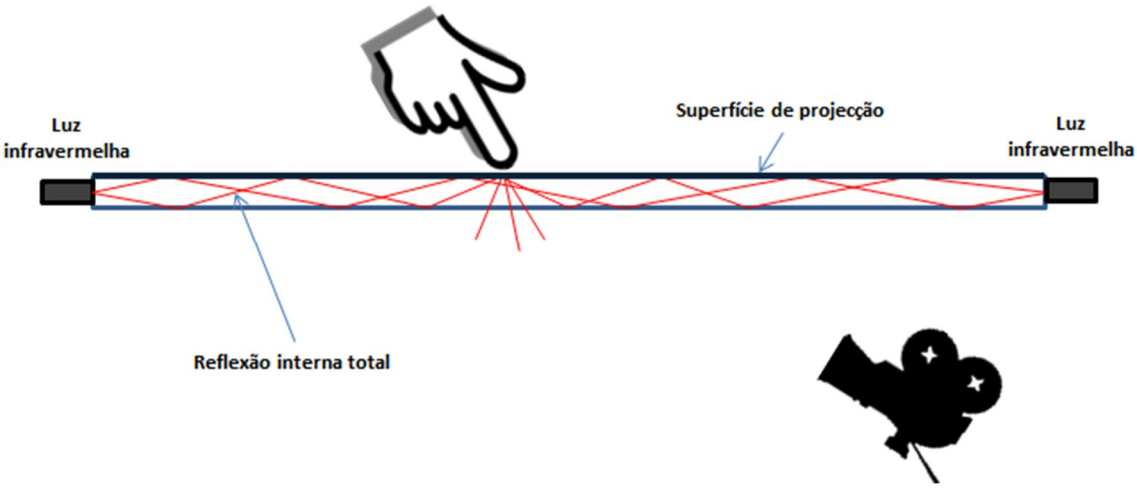


Figura 5 – Funcionamento da técnica FTIR.

Também é comum adicionar uma película de borracha de silicone na placa, na face virada para o utilizador. Esta película permite um aumento da área de contacto do toque que pode ser aproveitada para identificar variações de pressão sobre a placa, ajuda ao aumento de contraste dos *blobs*, e protege a superfície dos riscos.

Na Fig.6 pode-se ver a imagem captada pela câmara. Neste exemplo, são identificados cinco *blobs*, correspondentes aos dedos da mão que interage com a placa de acrílico.

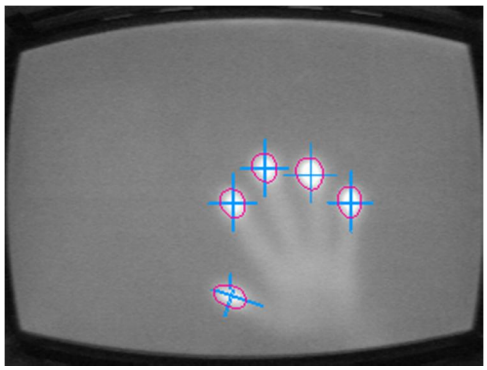


Figura 6 – Imagem iluminada por luz IR.

O fenómeno FTIR foi redescoberto e disseminado por Jeff Han, oferecendo uma solução de implementação de hardware MT, a baixo custo, potenciando a investigação na área e o desenvolvimento de novas aplicações [SKO09].

2.2.4 LLP

O Laser Light Plane (LLP) é um método no qual a luz IR é emitida logo acima do ecrã, como representado na Fig.7.

Quando um dedo toca no ecrã, a luz é refletida para baixo nesse ponto que é detetado devido ao surgimento de um *blob* na imagem captada pela câmara IR [NUI09] [Set08].

Figura 7 – Funcionamento de uma superfície de tecnologia *laser*.

A imagem captada pela câmara é semelhante à imagem obtida com a técnica FTIR, mas no caso da técnica LLP, pode verificar-se a ocorrência de anéis, uma vez que a luz IR incide diretamente sobre os dedos.

Um problema que pode ocorrer com esta técnica é o da oclusão dos dedos, Fig.8. Este problema surge quando uma grande quantidade de dedos está em contacto com a superfície, podendo criar zonas de sombra para os *lasers*, impossibilitando a deteção de toques nessas zonas. No entanto, pode ser minimizado com a inclusão de mais *lasers*, que assim "preenchem" melhor a superfície, colmatando as zonas de oclusão.

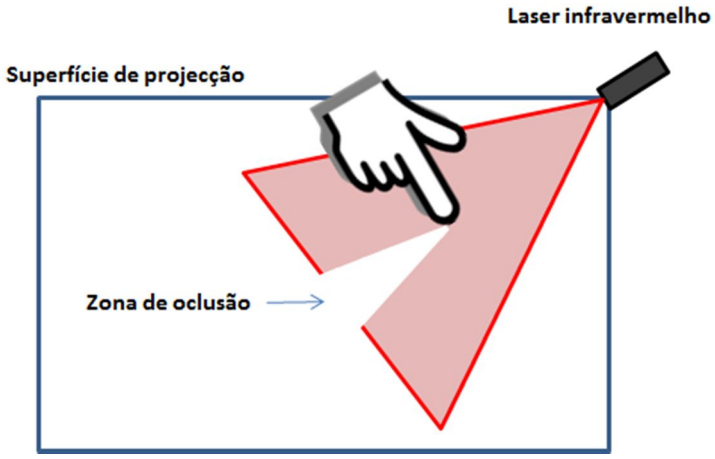


Figura 8 – Efeito de oclusão na técnica LLP.

2.2.5 Comparação entre as diferentes tecnologias

Os quatro exemplos apresentados representam algumas das tecnologias mais utilizadas para captação de toques, mas possuem caraterísticas diferentes que, dependendo das situações ou recursos disponíveis, podem ou não ser boas opções. Para facilitar estas decisões, a Tab.1 apresenta um estudo comparativo entre as tecnologias:

Tabela 1 – Comparativo entre tecnologias para captação de toques [Ove07].

Tecnologia	Custo	Claridade da imagem	Resposta aos toques	Dimensão	Vantagens	Desvantagens
Resistiva	Médio	Média	Média	Média/Pequena	Sensibilidade à pressão, pouco afetada pela sugidade da superfície.	Fraca resistência a riscos.
Capacitiva	Elevado	Elevada	Elevada	Média/Pequena	Durabilidade, precisão.	Não pode ser utilizado com luvas ou *stylus*.
FTIR	Baixo	Elevada	Baixa	Média/Alta	Facilidade de montagem.	Afetada pela sujidade da superfície.
LLP	Baixo	Elevada	Baixa	Média/Alta	Facilidade de montagem.	Baixa resolução de toque. Pode detetar *blobs* antes do toque.

Com base nestes resultados pode-se concluir que a melhor solução depende dos objetivos e meios disponíveis. Ao nível da qualidade de imagem e resposta ao toque a tecnologia capacitiva será a melhor opção, mas tem um custo elevado e só pode ser utilizado com o dedo humano. A tecnologia resistiva oferece boa qualidade a um preço mais competitivo, mas só para pequenas dimensões. As tecnologias óticas (FTIR e LLP) são as mais acessíveis no que toca a custos e facilidade de implementação, sendo boas opções para superfícies de grandes dimensões.

2.3 Dispositivos

Como foi referido, o número de aplicações que utilizam tecnologia MT tem crescido a um ritmo elevado nos últimos anos, e as previsões apontam para que assim continue. Estima-se que em 2010 o mercado mundial de dispositivos MT valia $4.58 mil milhões de dólares, e que esse valor será de $6.09 mil milhões no presente ano de 2011, e que em 2014 o valor poderá estar em $9.65 mil milhões de euros [Lee11]. Os gráficos na Fig.9 clarificam essa ideia.

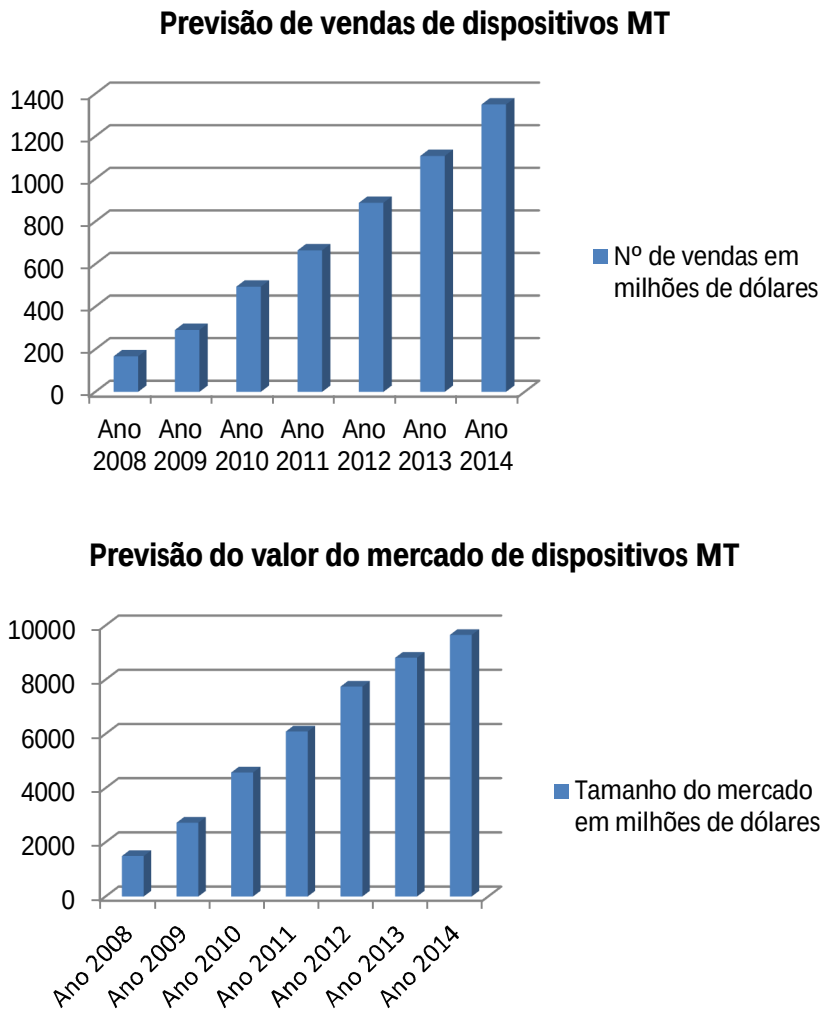


Figura 9 – Previsões para o mercado das tecnologias MT.

São números expressivos que correspondem a uma grande quantidade de dispositivos vendidos. Uma das razões para este crescimento é a crescente penetração desta tecnologia nos mais diversos dispositivos como *smartphones*, *tablets* ou *desktops*.

Já se desenvolvem equipamentos que tiram partido da colaboração entre várias pessoas (Surface), e da facilidade de aprendizagem das interfaces naturais (Reactable). Esses exemplos, entre outros, são apresentados de seguida.

2.3.1 iPhone/iPod/iPad

Ao iPhone é atribuída a responsabilidade do recente *boom* na procura de dispositivos táteis. Foi o primeiro dispositivo multitoque a conseguir impor-se no mercado e a tornar-se referência. Este produto faz sucesso principalmente pela forma como consegue integrar a tecnologia MT com serviços adequados a todo o tipo de utilizadores. O sucesso foi tão grande que a Apple investiu em novos dispositivos de interação táctil. Adicionou um ecrã tátil ao leitor de música iPod, e mais recentemente voltou a inovar com o iPad. Para muitos o iPad pode não ser mais do que um iPhone maior, mas também pode ser visto como uma prova de que ainda existe mercado para o MT à espera de ser explorado.

O iPhone utiliza tecnologia capacitiva no seu ecrã, permitindo uma ótima passagem da luz e qualidade de imagem.

Figura 10 – Apresentação do iPad.

2.3.2 Microsoft Surface

O Surface é um computador MT de médias dimensões, capaz de responder a gestos manuais e a objetos reais, e graças às suas dimensões (30 polegadas) oferece uma boa experiência de trabalho colaborativo permitindo dezenas de pontos de contato em simultâneo [MFW10].

A identificação de objetos, tais como telemóveis ou câmaras digitais, permite a troca de informação entre estes e o Surface. A identificação é feita através do reconhecimento das formas do objeto ou através do código de barras. Desta forma é possível transferir, por exemplo, informação entre dois telemóveis usando o Surface como intermediário.

Figura 11 – Microsoft Surface.

O rol de possíveis aplicações deste computador é alargado, podendo ser utilizado em serviços financeiros no contacto com o cliente, nos serviços hospitalares e em emergências médicas onde pode funcionar como um *White board*, no setor da restauração para o pedido do cardápio ou da conta, ou como apoio ao ensino nas escolas fazendo uso das suas capacidades colaborativas para envolver os alunos na aprendizagem.

A Microsoft disponibiliza o Surface SDK que inclui a API e respetiva documentação para facilitar o desenvolvimento de aplicações para esta plataforma.

2.3.3 Second Light

Resultado dos laboratórios de investigação da Microsoft. O Second Light utiliza uma tecnologia muito interessante que permite alargar o campo de interação para além do ecrã, oferecendo projeção de conteúdo secundário, tirando partido de gestos efetuados acima da superfície.

Para isso são projetadas duas imagens, uma para a superfície de projeção e outra para um plano mais acima. A segunda superfície de projeção pode ser utilizada para oferecer mais conteúdo, nomeadamente, detalhes relativos à imagem da superfície de projeção principal [IHT08]. Um exemplo desta tecnologia pode ser observado na Fig.12, onde o utilizador usa uma folha de papel para visualizar o conteúdo secundário projetado acima da superfície de projeção.

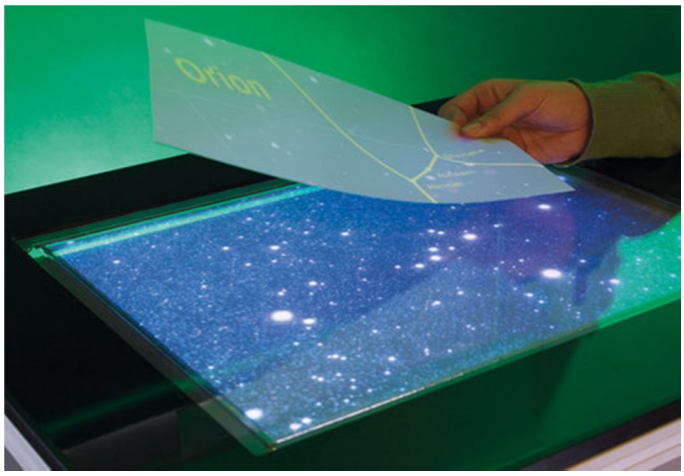


Figura 12 – Second Light.

2.3.4 Side Sight

Uma desvantagem da tecnologia MT quando é utilizada em dispositivos de pequenas dimensões, como os telemóveis, é que os dedos ocultam partes do ecrã sempre que há uma interação. Uma solução para este problema é avançada pela Microsoft Research, e tem o nome de Side Sight. Esta técnica utiliza sensores de proximidade, de luz IR, para detetar a presença dos dedos nas posições laterais ao dispositivo. A deteção dos pontos é semelhante à técnica LLP, em que a presença dos dedos é detetada quando a luz IR emitida é refletida ao tocar num dedo, e detetada por fotodiodos colocados junto ao emissor IR.

Desta forma a interação com o telemóvel é possível, sem ocultar nenhuma área do ecrã, uma vez que o espaço de interação é exterior ao próprio dispositivo. O espaço é limitado pelo alcance da luz IR utilizada, que neste caso varia entre 5 e 10 centímetros de distância lateral do dispositivo [BIH08]. Na Fig.13 é possível observar um exemplo dessa interação.

Figura 13 – Side Sight.

2.3.5 Reactable

A Reactable é uma superfície MT vocacionada para a criação e manipulação de música de uma forma visual e intuitiva. Começou por ser desenvolvida para músicos funcionando como uma espécie de mesa de mistura, mas atualmente já existem *designs* específicos para espaços públicos e instituições como museus, centros de ciência ou escolas. Pelas suas dimensões também permite a colaboração.

A interação com a mesa é feita por intermédio da utilização de objetos, que têm padrões fiduciais nas suas faces. Estes padrões são identificados pelo sistema de reconhecimento do sistema, que atribui operações conforme os padrões utilizados. Estes objetos são como ferramentas para síntese musical.

Como se pode observar na Fig.14, os objetos são reconhecidos e são estabelecidas ligações entre eles que permitem misturar a música em tempo real de uma forma fluída, e mais acessível ao público em geral do que as formas convencionais [Jor03].

Figura 14 – Reactable.

2.3.6 Perceptive Pixel

Perceptive Pixel é o nome de uma empresa que investiga e desenvolve algumas das soluções MT mais avançadas do mercado. Para isso, a empresa aposta forte na qualidade e inovação, através da ligação que mantém com o mundo académico, e das parcerias com empresas que apoiam projetos de investigação, como a 3M ou a Intel Capital.

Os sistemas desenvolvidos por esta empresa vêm acompanhados de um SDK que permite aos programadores integrarem as capacidades do MT em aplicações existentes, sendo uma ferramenta fundamental para a penetração destes sistemas no mercado.

Oferecem soluções para diversas áreas como design industrial, indústria militar, medicina ou transmissões televisivas. Um destes exemplos aconteceu no canal de televisão CNN (Fig.15), que acompanhou as eleições presidenciais americanas em 2008, numa tela MT, denominada "Magic Wall", desenvolvida pela Perceptive Pixel [Bro08].

Figura 15 – Perceptive Pixel.

2.3.7 Edigma

A Edigma é uma empresa portuguesa, que está a desenvolver uma tecnologia, que consiste numa película de polímero, denominada *Displax Multitouch Technology*, que permite aplicar o MT em qualquer superfície. Utiliza tecnologia projetada capacitiva, e usa um controlador que processa múltiplos sinais de *input*, que recebe de uma malha de nanotubos embebidos na película [WHF10]. As suas propriedades permitem que superfícies não condutoras, como por exemplo, o vidro ou a madeira, possam tornar-se superfícies MT.

Esta película pode reconhecer 16 pontos de contacto simultâneos, e possui capacidade para medir a intensidade e fluxo de movimentos do ar, permitindo atuar como detetor de sopro [WHF10].

Esta tecnologia pode aumentar o leque de aplicações possíveis, permitindo que mais superfícies possam acolher um sistema MT.

Figura 16 – Utilização da tecnologia Displax num shopping.

2.4 Software

Atualmente existem algumas bibliotecas para desenvolver software MT. As suas funcionalidades baseiam-se na interpretação de toques e gestos, geração de eventos de *input*, e protocolos de comunicação para passar essa informação às plataformas de desenvolvimento.

Como pode ser visto na Fig.17, ao nível da arquiterura de *software* de um sistema MT, podemos dividi-la em três camadas distintas, com diferentes níveis de abstração:

- **Camada de input –** Reconhecimento de toques e gestos.
- **Middleware –** Protocolos de comunicação entre a camada de input e a camada de aplicação.
- **Camada de aplicação –** Ferramentas para o desenvolvimento de aplicações.

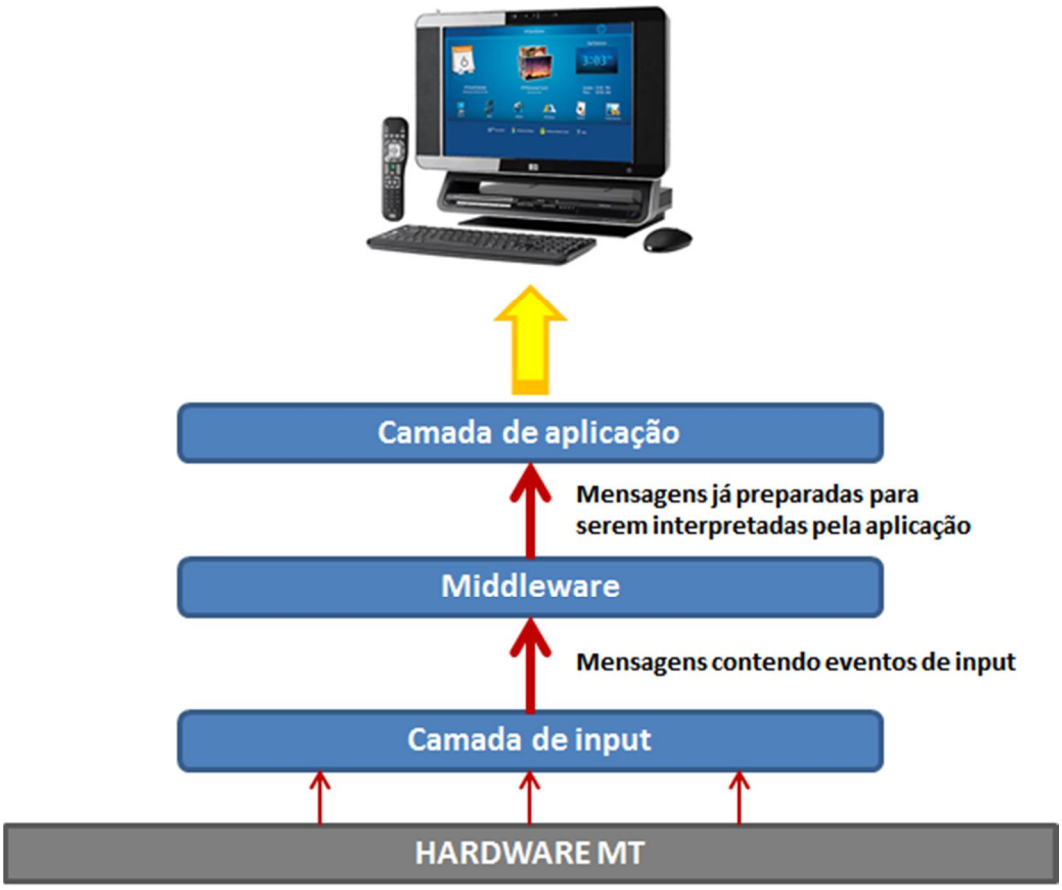


Figura 17 – Arquitetura de *software* de um sistema MT.

A camada de *input* contacta diretamente com a informação de entrada. Trata essa informação, e encaminha-a para o nível superior através de mensagens. A camada intermédia, *middleware*, diz respeito ao protocolo de comunicação estabelecido entre o software de baixo nível e o software de alto nível. A camada de aplicação representa as plataformas de desenvolvimento de software, que usam a informação proveniente das camadas de baixo nível, para produzir eventos ao nível das aplicações.

Isto é uma visão geral do que realmente acontece num sistema MT. O funcionamento de cada camada vai ser mais aprofundado na próxima subsecção.

2.4.1 Camada de *input*

O software de mais baixo nível recebe a informação dos toques proveniente do hardware MT, faz o respetivo tratamento dessa informação, e por fim envia-a para as camadas de abstração mais altas. O tratamento dessa informação consiste na deteção e *tracking* de *blobs*, e para isso, utiliza-se processamento de imagem e reconhecimento de padrões.

Existem aplicações que fazem esse processamento, entre as quais o Touchlib, o Community Core Vision e o Reactable, que vão ser apresentadas já de seguida.

Touchlib

O Touchlib é uma biblioteca *open source* para deteção de *input* dos dedos. É desenvolvida pela NUI Group Community, uma comunidade online que produz muita informação relacionada com a tecnologia. Esta biblioteca funciona com tecnologias óticas que usem câmaras, como por exemplo, as técnicas FTIR e LLP [Gro10].

Para cada toque detetado, a informação, contendo dados sobre as coordenadas do toque, o deslocamento, a área de toque, a variação dessa área, e o ID, é enviada para as aplicações que suportam o protocolo de comunicação. O protocolo utilizado é o TUIO, que será alvo de uma descrição detalhada, mais à frente neste documento.

O Touchlib utiliza também outras bibliotecas nos seus processos. A biblioteca OpenCV que contém algoritmos de visão por computador, o *wrapper* DSVideoLib para suportar acesso concorrente aos *buffers* utilizados, o VideoWrapper que funciona como uma API para a interface com a câmara de vídeo, e o OSCpack que é um conjunto de classes em C++ para a compressão e descompressão de pacotes OSC [Tho07].

Na Fig.18 podemos ver o Touchlib em execução, e os diferentes filtros aplicados à imagem obtida pela câmara.

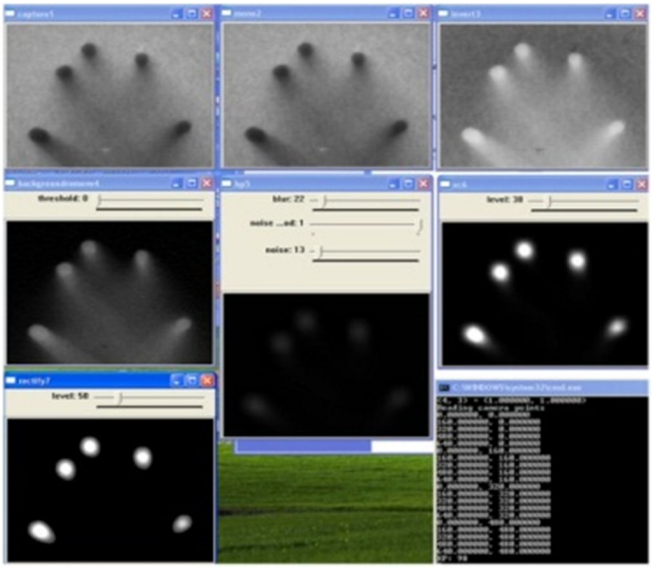


Figura 18 – Touchlib.

Community Core Vision (CCV)

O CCV é um projeto também desenvolvido pela NUI Group Community, sendo uma versão mais avançada do Touchlib, e com suporte para multiplataforma (Windows, Linux e Mac).

Como pode ser visto na Fig.19, o CCV tem as mesmas funcionalidades do Touchlib, como a identificação e *tracking* de *blobs*, mas possui outras como mais opções de calibração ou a possibilidade de usar mais do que uma câmara. No caso de não haver nenhuma disponível, podem-se fornecer vídeos de teste para o efeito. Mais detalhes sobre a utilização do CCV podem ser consultados no *Anexo D – Preparação do Hardware/Software*.

O CCV pode comunicar com várias aplicações que suportem os protocolos de comunicação TUIO ou XML, podendo assim enviar informação sobre os *blobs* para as aplicações. São estas mensagens que as aplicações utilizam como *input* das ações do utilizador [Liu10].

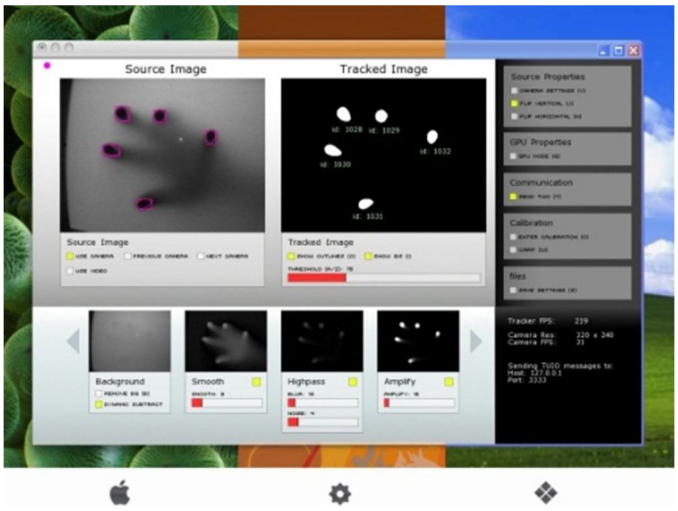


Figura 19 – *Screenshot* do CCV em execução.

Reactivision

O Reactivision, tal como o CCV, também é um sistema *open source* e multiplataforma, mas neste caso serve para a deteção de objetos fiduciais. Para o efeito são utilizados objetos para interagir com a superfície. Os objetos possuem padrões nas suas faces. Estes padrões funcionam como identificadores de operações da aplicação, que são descodificadas através de técnicas de processamento de imagem semelhantes às que são utilizadas na realidade aumentada. A este tipo de interface dá-se o nome de *tangible user interfaces* (TUI) [ShH10].

O Reactivision funciona como o software de input da Reactable. Processa a imagem obtida por uma câmara montada no sistema, processa os objetos fiduciais identificados, e envia a informação tratada para as aplicações cliente em mensagens compatíveis com o protocolo TUIO. O resultado pode ser observado na Fig.20, onde à esquerda temos o exemplo de um padrão fiducial, e à direita podemos ver a utilização dos objetos na interação [JKGB07].

Figura 20 – Reactivision.

2.4.2 Camada de *middleware*

O *middleware* funciona como uma camada intermédia, fazendo a tradução da informação obtida pelo software da camada de baixo nível para o software da camada de alto nível. Existem algumas ferramentas de reconhecimento e interpretação de toques que fazem precisamente esta ponte entre o software de baixo e o de alto nível.

Como já foi referido, aplicações que comuniquem através dos protocolos de comunicação TUIO/OSC/XML estão habilitadas a receber informação acerca de *blobs* e *tracks* para utilizar em eventos. Por exemplo, numa estação MT pode utilizar-se o CCV para detetar os *blobs* e as *tracks* em contacto com a superfície de projeção. O CCV identifica esses pontos, estabelece *sockets* para comunicação e envia mensagens para a camada intermédia de software. As coordenadas obtidas são então processadas como os pontos de contacto que o utilizador estabelece a cada momento, e que devem produzir uma resposta adequada da aplicação através da projeção da imagem na superfície.

Esta camada intermédia pode ser representada pelo protocolo TUIO, esquematizado na Fig.21.

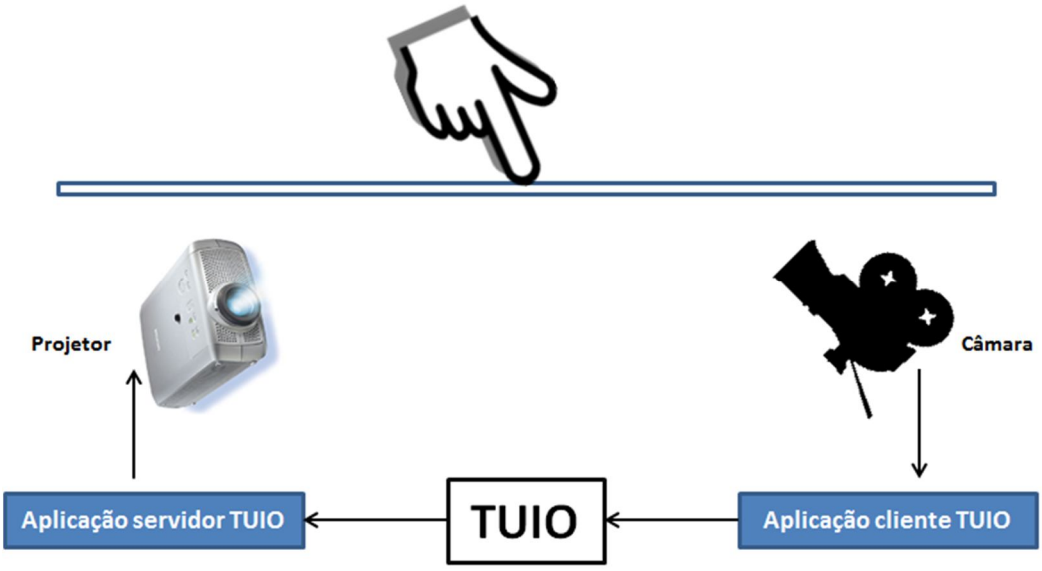


Figura 21 – Exemplo da arquitetura TUIO.

2.4.3 Camada de aplicação

Na camada de aplicação encontram-se as plataformas de desenvolvimento que tiram partido dos dados recebidos pelas camadas de baixo e médio nível, ou seja, as aplicações cliente. Esta é uma camada de alto nível, onde existe uma grande abstração dos detalhes relacionados com a deteção de *blobs* e consequente comunicação dos eventos.

Atualmente existem vários SDK's para desenvolver aplicações MT. Alguns deles foram testados ao longo desta dissertação, e serão brevemente analisados.

PyMT

A plataforma de desenvolvimento PyMT permite o desenvolvimento de aplicações MT, na linguagem Python.

As vantagens do PyMT são extensas. As propriedades dinâmicas de Python e a sua sintaxe concisa, permitem uma prototipagem rápida das aplicações. A documentação disponível é extensa, e sendo um projeto *open source*, possui uma grande comunidade de programadores que mantêm atualizada a plataforma. Tal como Python, o PyMT é multiplataforma, o que significa que corre em Windows, Mac e Linux. Para a produção gráfica é utilizado o OpenGL, e fornece chamadas diretas a operações OpenGL.

A programação com PyMT baseia-se na criação de *widgets*. A biblioteca disponível fornece um conjunto vasto de *widgets* prontas a utilizar, como por exemplo, botões, *layouts*, *sliders*, entre outros elementos típicos de uma interface. Essas *widgets* podem depois ser editadas com CSS, ou então animadas através da *framework* de animação integrada na plataforma.

PyMT suporta vários protocolos de *input* de dados provenientes dos dispositivos. O rol de protocolos inclui o TUIO, Windows 7 MT, OSX MT, Linux MT, entre outros. A informação de *input* é apresentada às *widgets* como instâncias de objetos, podem ser adicionadas novas funcionalidades aos dispositivos de entrada, facilitando o *debugging*. Durante a execução das aplicações, podem ser adicionados cursores com o botão direito do rato, e arrastados pelo ecrã, simulando vários eventos de toque. Esta capacidade para simular cursores permite que se possam desenvolver aplicações MT, mesmo sem o *hardware* disponível.

Todas estas características fazem do PyMT uma boa opção para desenvolver aplicações MT. O facto das aplicações funcionarem em vários sistemas é uma vantagem prática muito interessante e que pode ser explorada. Por se basear numa linguagem de alto nível e possuir uma grande variedade de bibliotecas disponíveis, facilita o trabalho dos programadores permitindo desenvolver aplicações interativas, bastante apelativas, num curto espaço de tempo [HDV10].

Windows 7 Multitouch

O Windows 7 trouxe uma série de inovações, entre elas, o suporte integrado para MT. É possível tirar partido da experiência MT do Windows 7, mesmo sem se possuir um *driver* nativo. Para isso é necessário proceder a uma instalação da *driver* Multi-Touch Vista. Esta *driver* habilita o Windows a receber dados de *input* provenientes do TUIO. Foi elaborado um tutorial onde é explicada a instalação e configuração da *driver*, e pode ser consultado no *Anexo A – Instalação do Multitoque em Windows 7*.

Possui duas características interessantes, a manipulação, que permite que vários gestos sejam combinados para executar tarefas mais complexas, e a inércia, que não é mais do que um modelo de física aplicada aos componentes, melhorando a experiência de utilização. Quando são detetados toques na superfície MT, a plataforma envia mensagens WM_GESTURE para a aplicação que está a ser executada, contendo toda a informação relativa ao toque.

A Microsoft tem contribuído para o desenvolvimento de aplicações MT, através de algumas ferramentas que tem criado. Uma delas, a Windows 7 Multitouch API, é uma biblioteca para o

desenvolvimento de aplicações MT em linguagem C#. Outro exemplo é o Expression Blend 4, que permite o desenvolvimento de aplicações para *web*, *desktop* e *Windows phone*, sendo possível integrar gestos nestas aplicações.

Gesture Works

O Adobe Flash é uma plataforma multimédia para desenvolvimento animações, vídeo, páginas *web*, jogos e outros conteúdos interativos. Esta plataforma também possui uma biblioteca de gestos *open source* para desenvolver software MT, o Gesture Works. O Gesture Works utiliza a linguagem de programação proprietária, o Action Script 3.0, possui suporte MT para o Adobe Flex, disponibiliza um conjunto de gestos prototipados, mas também permite que o programador os estenda ou crie os seus próprios gestos à medida das necessidades.

Processing

O Processing é uma linguagem de programação *open source* integrada num ambiente de desenvolvimento especialmente dedicado à criação de imagens, animações e interfaces interativas. É uma linguagem orientada a objetos e a sintaxe do código é semelhante a Java.

Para desenvolver aplicações MT, o Processing recebe mensagens TUIO com dados sobre os toques provenientes da aplicação servidor, e interpreta-as.

2.5 Perspetiva final sobre o multitoque

A tecnologia multitoque permite interagir com dispositivos eletrónicos, de uma forma mais interativa e natural do que a permitida com as tecnologias de interação habituais.

Neste capítulo foram apresentadas algumas das tecnologias mais utilizadas para captação de toques numa superfície. São das mais utilizadas porque apresentam bons resultados nos contextos em que são mais usuais, ou seja, nos dispositivos móveis (tecnologia resistiva e capacitiva), em ecrãs de média ou grande dimensão (tecnologia FTIR e LLP), e que representam aquilo que de melhor se faz a este nível.

Foram identificados alguns dispositivos que são bons exemplos de como se pode aproveitar a tecnologia MT. Uma área onde a presença da tecnologia é bem notória é a área dos *smartphones*, mas lentamente começa-se a verificar mais soluções em outras áreas. Nomeadamente, em soluções apresentadas no campo dos dispositivos com ecrãs de grandes dimensões, que são utilizados em espaços públicos (Surface ou Reactable), ou que começam a fazer parte de transmissões televisivas (*Sense Wall* da Perceptive Pixel), passando por soluções que poderão viabilizar outras aplicações (tecnologia Displax).

Parte da adoção generalizada a esta tecnologia, passa pela criação de *software* orientado ao paradigma de interação natural. Da mesma forma que foi preciso mudar a forma de pensar as aplicações na passagem das interfaces textuais para as interfaces gráficas, também é necessário pensar de forma diferente ao desenvolver aplicações para interfaces naturais. Já existem várias

ferramentas que permitem esse desenvolvimento, como o PyMT, Processing, Gesture Works, entre outras.

A tecnologia poderia ser aproveitada em equipamentos que utilizámos no nosso quotidiano, como máquinas de multibanco, nos equipamentos hospitalares, em restaurantes, ferramentas interativas para auxiliar o ensino, desenvolver novos paradigmas de interação no âmbito de jogos, etc. Todos os equipamentos que possuem uma interface podem obter ganhos de interatividade com a adoção desta tecnologia.

Também pode ser utilizada no âmbito da investigação de novas interfaces homem-máquina, como por exemplo, a realidade virtual e a realidade aumentada, ou como complemento a sistemas orientados por computador. Podemos imaginar o quão interessante seria integrar voz, som, imagem e gestos num só sistema computacional [CDG98].

Estas são algumas das áreas que podem beneficiar com a utilização da tecnologia MT nas suas atividades. Em suma, e sendo esta uma tecnologia que assenta na interação manual, o limite para as aplicações MT coincide com a nossa capacidade de interagir manualmente com o mundo à nossa volta.

Capítulo 3

Framework Multitoque para Manipulação de Informação

Uma *Framework* pode ser vista como um conjunto de ferramentas, bibliotecas e de boas práticas, assentes numa plataforma, cujo objetivo é assistir os seus utilizadores através de uma abstração de tarefas repetitivas, fornecendo módulos genéricos reutilizáveis. Desta forma, um programador pode focar-se nos detalhes específicos do projeto que tem em mãos, em vez de implementar funcionalidades comuns sempre que inicia um novo projeto. Isto permite que se poupe tempo do desenvolvimento, e simultaneamente, oriente a implementação [Cro07].

O objetivo da implementação é criar uma *framework*, que facilite o desenvolvimento de aplicações interativas para estações MT de médias ou grandes dimensões. Estas aplicações devem-se focar na consulta de informação proveniente do SiFEUP, fornecendo uma alternativa ao *site* atual.

Do ponto de vista do programador, a *framework* deve possuir propriedades como, a extensibilidade, a modularidade e a integração de elementos, e ao mesmo tempo, deve permitir o desenvolvimento de uma forma orientada. Estas três propriedades envolvidas num projeto deste tipo permitem:

- **Extensibilidade –** A inclusão de nova informação disponibilizada pela FEUP, ou a criação de casos de utilização, aumentando a longevidade do projeto.
- **Modularidade** – Existindo modelos pré-definidos para a apresentação da informação, a implementação é mais rápida, mas também deve permitir que se possam editar esses modelos, adaptando-os a novas ideias.
- **Integração de elementos –** Os elementos produzidos pela *framework* devem permitir realizar aplicações integradas, ou seja, que os elementos da aplicação possuam algum tipo de ligação entre si e que essa ligação possa ser controlada para permitir fluxos de utilização.

A verificação destas propriedades assegura uma *framework* que diminui o tempo de desenvolvimento, e produz aplicações perfeitamente integradas, mas também assegura a possibilidade de expansões e atualizações nos projetos realizados, de forma a adequá-los aos requisitos.

No ponto de vista do utilizador, as aplicações produzidas pela *framework* devem oferecer uma interação simples, rápida, e tendencialmente natural. Estas aplicações devem possuir:

- Componentes gráficos, como por exemplo, janelas, botões, teclado virtual, etc.
- Manipulação desses componentes, através do redimensionamento, arrastamento ou rodagem.
- Tarefas multiutilizador, para que seja possível a utilização das aplicações por mais do que uma pessoa de cada vez.
- Gestos que permitam efetuar ações.
- As aplicações devem apresentar dados reais.

Todos os componentes gráficos devem ser interativos. Sendo o objetivo criar aplicações MT com interfaces naturais, convém simular ao máximo uma interface natural onde todos os objetos podem ser manipulados. Os componentes gráficos a utilizar devem ser os necessários para cobrir um conjunto razoável de representações de informação encontradas no SiFEUP. As necessidades passam por representar listagens, tabelas com dados, imagens, campos de texto, e ligações para permitir ações.

A capacidade multiutilizador das aplicações faz sentido se estas forem utilizadas em estações MT de médias ou grandes dimensões. Como é o caso deste projeto, a *framework* deve possibilitar que vários utilizadores interajam com as aplicações em simultâneo.

Os gestos são uma questão central nas interfaces que atuam em tecnologia MT. Uma interface é mais natural quanto mais fácil e inconsciente for a sua aprendizagem por parte do utilizador [MoN03]. Um gesto nunca deve ser complexo ao ponto de obrigar o utilizador a "perder" vários segundos para desenhá-lo, deve ser perfeitamente distinguível de outros gestos, e lógico para a operação a realizar. Um gesto simples poderia ser utilizado, por exemplo, para a abertura de um menu com várias opções, ou então para fechar uma janela.

Como é exemplificado na Fig.22, a solução passa por utilizar gestos que representem figuras conhecidas, como por exemplo, um quadrado ou uma cruz. Por outro lado, devem ser evitadas figuras como, por exemplo, a clave de sol, que seria de dificil representação para a maioria das pessoas [HaB01].

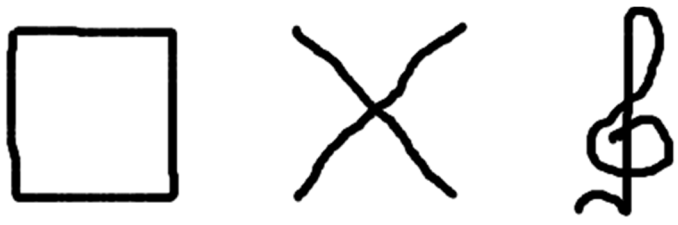


Figura 22 – Exemplos de gestos.

Sendo os gestos uma parte importante para as aplicações MT, convém que a *framework* permita aos programadores a prototipagem de novos gestos, e consequente integração destes nas aplicações. Este requisito torna a *framework* mais flexível, pois permite que novos gestos sejam adicionados às aplicações, à medida das necessidades.

A aplicação a desenvolver, como prova de conceito da utilidade da *framework*, será uma interface alternativa ao SiFEUP, logo, a informação tem de ser real. Sendo assim, a *framework* deve disponibilizar métodos que permitam a extração de dados de um *site*. Com essa possibilidade, pode-se criar uma demonstração com aplicações práticas, importante para a fase de testes.

No fim do desenvolvimento deve ser efetuada uma série de testes, idealmente com utilizadores finais, para melhor aferir o trabalho realizado. Estes testes devem, acima de tudo, avaliar aspetos da aplicação para os quais contribuiu a utilização da *framework*, e ajudar a perceber a potencialidade de uma aplicação deste tipo num ambiente como o da FEUP.

3.1 Especificação da *Framework*

Seguindo os objetivos propostos, a *framework* deve fornecer uma abstração para o desenvolvimento de aplicações MT, baseadas nas funcionalidades do SiFEUP. Tendo em conta estes requisitos, a implementação da *framework* dividiu-se em duas fases:

- Construção das *factories*.
- Construção de *widgets* e integração de gestos.

A construção das *factories* trata do *parsing* de informação do SiFEUP, e da formação de pacotes com os dados filtrados de acordo com os requisitos.

A construção de *widgets* serviu para dotar a *framework* de um conjunto de estruturas (*widgets* compostas[1]) para visualização e interação com a informação, tentando cobrir vários casos de uso, já preparados para serem implementados numa aplicação. Também foi criada uma pequena aplicação que possibilita a geração de protótipos de gestos, para depois serem integrados nas aplicações.

Para se obter uma visão geral da arquitetura, e do papel dos diferentes componentes, foi elaborado um esquema que pode ser observado na Fig.23.

[1] O nome “compostas” advém do facto de serem *widgets* que integram várias outras *widgets*, como botões, *sliders*, etc.

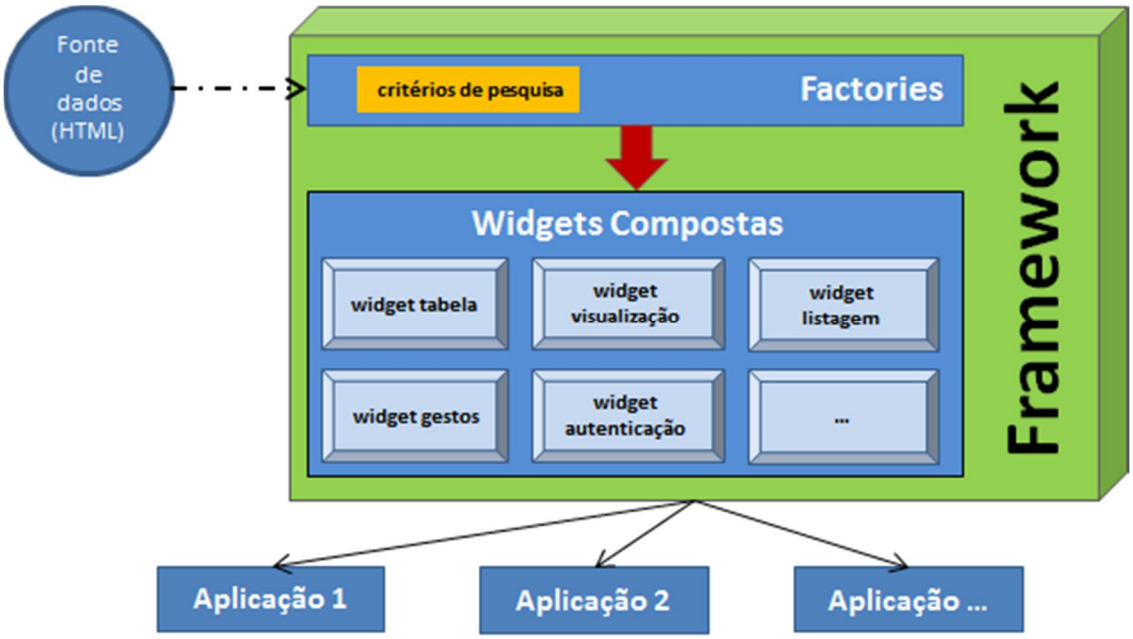


Figura 23 – Arquitetura da *framework*.

A *framework* está dividida em duas componentes. A componente principal, e que permite construir aplicações por si só, é a componente das *widgets* compostas. Estas funcionam como componentes básicas para a construção de aplicações. Necessitam de dados, que podem ser adicionados de uma forma dinâmica, provenientes das *factories* que filtram essa informação.

A segunda componente, as *factories*, permitem que o programador possa definir critérios para filtragem de informação do código (HTML) das páginas do SiFEUP, obtendo-se assim dados reais.

Estas duas componentes funcionam em conjunto, sendo a função das *factories* extrair dados que depois fornece às *widgets* compostas.

Depois de analisar o esquema com um nível de abstração elevado, é tempo de observar cada uma das partes detalhadamente.

3.1.1 *Factories*

A *framework* fornece uma estrutura sobre a qual se pode desenvolver aplicações, mas necessita de dados reais para gerar resultados úteis. Estes dados, no nosso caso de estudo, provêm das páginas do SiFEUP, cujo conteúdo se quer disponibilizar nas aplicações. Para o efeito é utilizado um *parser* HTML/XML para Python, o *Beautiful Soup*, que possui métodos para pesquisar por *tags* ou atributos numa árvore HTML. Esta decisão foi tomada devido à impossibilidade de utilizar diretamente a base de dados do SiFEUP.

A utilização do parser é demonstrada na Fig.24, onde se pretende obter o nome da faculdade através da página do SiFEUP. Para isso acede-se ao código fonte (HTML) e identifica-se a *tag* onde se encontra essa informação. O resultado do *parsing* é mostrado na janela de resultados.

Figura 24 – Exemplo de utilização do *Beautiful Soup*.

Regra geral, o objetivo das aplicações não será extrair um campo de dados, mas um conjunto de campos. A pensar nestes casos, foi desenvolvida uma plataforma dinâmica de extração de informação, que permite filtrar uma grande quantidade de dados de uma árvore HTML. Esta plataforma é formada por várias *factories*, em que cada uma é responsável pela filtragem de informação de um tipo de página. Pode ser dado o exemplo de uma factory para páginas de alunos, de onde são extraídos dados de interesse como o nome desse aluno, o código, ou o curso que frequenta. A informação resultante da filtragem de uma página é agrupada num formato pronto a ser interpretado pela *widget* composta correspondente a esse tipo de página.

Deve-se referir que a construção das *factories* depende dos dados que se pretendem para a aplicação. Se se pretender alterar a filtragem de dados de uma página, pode-se manter a estrutura da respetiva *factory*, alterando apenas as *tags* necessárias, mas se se pretender filtrar informação de outro site ou de uma página diferente, aí será necessário construir uma nova *factory* que se adapte ao novo código. No entanto, a construção de *factories* para esta *framework* foi feita com o intuito de ser o mais genérica possível, para que havendo necessidade de uma reestruturação, pudessem ser utilizadas como base.

3.1.2 *Widgets*

Esta *framework* possui um conjunto de *widgets*, que foram desenvolvidas tendo em conta a representação de informação necessária para apresentar algumas páginas do SiFEUP. Funcionam como blocos básicos para compor aplicações, onde serão colocados os dados para visualização, e onde será feita a interação do utilizador.

Durante o desenvolvimento, utilizaram-se as *widgets* fornecidas pelo PyMT para construir *widgets* compostas, e como tal, herdam as suas caraterísticas, como a capacidade de redimensionamento, de deslocamento e rotação, através de toques. As *widgets* da *framework* são compostas, pois nelas foram agrupadas várias funcionalidades, que fazem parte das abstrações a realizar a *framework*. São estas:

- Todas as *widgets* que apresentam informação possuem um botão que fecha a *widget*.
- Possuem um conjunto de funções para definir *labels*, botões, listas e imagens.
- Desenham uma ligação entre si e a *widget* que lhe deu origem, *widget* pai.
- Para cada tipo de página do SiFEUP cuja informação é alvo de *parsing*, existe uma *widget* composta já construída.

O botão de fecho das *widgets* funciona como uma forma rápida e intuitiva de organizar o ambiente de trabalho. Ao ser pressionado, a *widget* correspondente é eliminada da visualização. O mesmo acontece com todas as *widgets* de um fluxo de utilização originado a partir dessa *widget*, ou seja, todas as *widgets* filho.

As funções que permitem definir *labels*, botões, listas e imagens, fornecem uma forma de controlar a representação dos dados. No caso das *labels*, é possível utilizar texto proveniente de uma *factory* ou então texto escrito pelo programador, dando espaço a aplicações que utilizem dados reais, e aplicações de âmbito mais fechado. No caso dos botões, podemos fornecer uma instância do objeto a que queremos aceder, e também adicionar uma *label* ao botão. Para adicionar imagens às *widgets* basta utilizar o nome do ficheiro de imagem, que deverá estar na mesma pasta da *framework*. Em todos os casos, podem-se definir as posições dentro da *widget*, mas no caso das imagens também é possível definir as dimensões.

As ligações entre *widgets* têm como objetivo manter alguma consistência e organização na utilização das aplicações. Por um lado permite manter as *widgets* ligadas, o que pode ser útil para identificar o "nosso trabalho" quando existem vários utilizadores em simultâneo na mesma estação MT, e por outro lado, fornece uma visualização em árvore do fluxo de interações, dando uma visão global da aplicação.

Com isto obtemos um conjunto de *widgets* prontas a utilizar para determinados tipos de páginas do SiFEUP, agilizando o processo de desenvolvimento, mas sempre com a possibilidade de fazer alterações às *widgets*, que permitam adaptar as aplicações às exigências dos projetos.

Agora que estão identificadas as abstrações fornecidas pelas *widgets*, será apresentado o conjunto de *widgets* compostas que foram desenvolvidas para a *framework*. As *widgets* que se seguem, têm como principal objetivo representar informação.

Widget para Listagem de Informação.

Esta *widget* fornece uma visualização de uma lista de dados, apresentada na forma de uma *scroll list*. Com este formato é possível conter uma grande quantidade de informação numa *widget*.
Os elementos da lista podem ser *labels* ou botões. Quando o objetivo for obter uma listagem de informação para visualização, podem ser utilizadas *labels*, mas se o objetivo for obter uma lista

de ligações para outras *widgets* é necessário utilizar botões. A decisão fica ao cargo do programador.

Widget para Visualização de Informação.

Esta *widget* fornece uma visualização de um conjunto de dados, normalmente referentes a uma entidade, ao contrário da *widget* anterior.

Os elementos da *widget* podem ser *labels*, botões, ou imagens, sendo possível definir a posição de cada. No caso das imagens também é possível definir a dimensão. Este grupo de representação de dados permite que estas *widgets* sejam utilizadas sempre que se deseje obter uma visualização de informação, como por exemplo, a ficha de um aluno ou professor.

Widget para Tabela.

Esta *widget* fornece uma visualização de informação em matriz, podendo ser utilizada se a informação tiver de ser mostrada como uma tabela.

Os elementos da tabela podem ser *labels* ou botões. Quando o objetivo for obter uma tabela para visualização de dados podem ser utilizadas *labels*, mas se o objetivo for obter uma tabela cujos elementos podem ser ligações para outras *widgets*, é necessário utilizar botões.

Para além de *widgets* para representar informação, existem *widgets* responsáveis pelas operações de suporte às aplicações.

Widget para Autenticação.

Esta *widget* fornece os componentes básicos para programar a funcionalidade de autenticação num sistema.

Quando instanciada, a *widget* para autenticação constrói automaticamente uma janela com, dois botões (um para confirmar os dados da autenticação e outro para cancelar), e dois campos de texto para introduzir o nome de utilizador e a palavra-passe. A inserção de dados nos campos de texto é feita com recurso a um teclado virtual, que é invocado sempre que se tocar num destes campos. O teclado virtual pode ser manipulado como qualquer outra *widget*, ou seja, é possível redimensionar, rodar e deslocar. Para fechar uma instância desta *widget*, basta tocar no botão *Cancel*.

Widget para Gestos.

Esta *widget* fornece uma área na janela da aplicação, onde podem ser efetuados gestos para ativar alguns eventos.

A *framework* possui alguns protótipos de gestos prontos a utilizar, mas também é dada a possibilidade ao programador de criar e gravar os seus próprios protótipos, podendo depois integrá-los nas aplicações.

Esta *widget* ocupa todo o fundo da aplicação, para se poder efetuar gestos em qualquer ponto do ecrã, e não pode ser fechada, uma vez que será sempre necessário fazer identificação de gestos.

3.2 Detalhes de Desenvolvimento

A tecnologia escolhida para desenvolver a *framework* foi a linguagem de programação Python, e a biblioteca *open source* para multitoque, o PyMT (versão 0.5.1). Foi escolhido o PyMT pelas caraterísticas orientadas ao MT que apresenta, nomeadamente, a fácil prototipagem de aplicações e o seu conceito de *widgets*, que permitem desenvolver aplicações baseadas nestas componentes gráficas.

Para a identificação de *blobs*, e envio das mensagens TUIO para o PyMT, foi utilizado o CCV. Para obter mais detalhes sobre este processo, pode ser consultado o *Anexo D – Preparação do Hardware/Software*, que possui uma descrição sobre a utilização desta aplicação, e da montagem da estação MT.

Existem três etapas do desenvolvimento da *framework* que merecem uma explicação detalhada acerca do seu funcionamento. Estas componentes são a extração de informação, o processamento de gestos e a criação de *widgets*.

3.2.1 Extração da informação

A extração de informação é uma parte importante da *framework*. Com esta componente é possível desenvolver aplicações com dados reais. Essa extração é feita com o *Beautiful Soup*.

A extração de informação é feita por componentes específicas da *framework*, as *factories*. Como já foi referido anteriormente, estas componentes tratam de fazer o *parsing* do código fonte das páginas que lhe são fornecidas, e extraem a informação especificada nos filtros.

Para reduzir os recursos computacionais utilizados com a pesquisa, recorreu-se a um método do *parser*, o *Soup Strainer*. Este método permite fazer um "corte" no código, a partir do nó em que queremos pesquisar. Desta forma, o espaço de pesquisa é reduzido ao indispensável, e o desempenho melhorado.

A pesquisa efetiva é realizada com os métodos *find* e *findAll*. Estes métodos procuram na árvore HTML, *tags* especificadas como parâmetros de entrada, e devolvem os resultados. No caso do primeiro método, é devolvido o primeiro resultado encontrado, e no segundo método são devolvidos todos os resultados. Estes resultados são utilizados para a construção de pacotes de dados, que são fornecidos às *widgets* compostas.

3.2.2 Gestos

Esta *framework* possui alguns protótipos de gestos, que foram criados durante o desenvolvimento da plataforma. Estes protótipos estão prontos para ser utilizados nas aplicações desenvolvidas nesta *framework*, mas também podem ser criados outros protótipos e integrá-los nas aplicações. Para isso, pode ser utilizada uma pequena aplicação desenvolvida durante este projeto, que permite o desenho e gravação do protótipo de um gesto.

A utilização desta aplicação pode ser vista na Fig.25, onde se pode ver que os pontos do gesto desenhado são guardados numa lista, visível na janela de comandos.

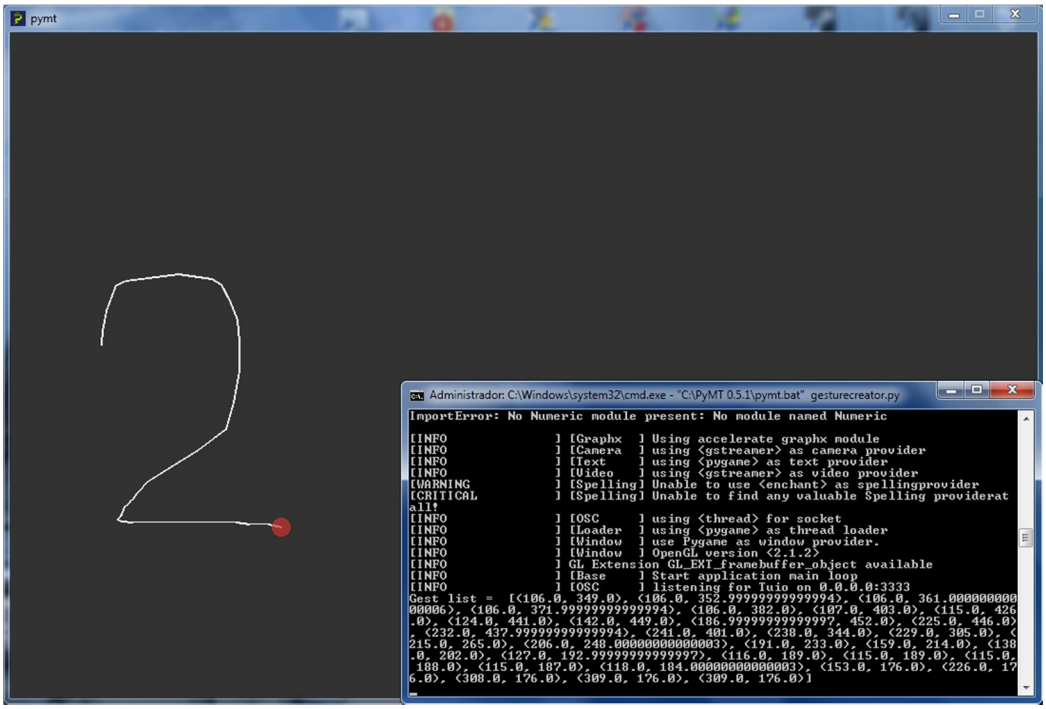


Figura 25 – Exemplo da criação do protótipo de um gesto.

Para o reconhecimento dos gestos, foi utilizado algoritmo *$1 Unistroke Recognizer* [Li10] [WWL10], que consegue reconhecer gestos independentemente da sua posição, escala ou rotação. É um algoritmo bastante eficiente, permitindo uma certeza de 97% nos resultados obtidos da comparação entre gestos e padrões de gestos guardados na sua base de dados [WWL07].

A aplicação da Fig.25 é simples de utilizar. Depois de executar o programa, temos uma janela onde podemos efetuar o gesto pretendido. Todos esses movimentos são desenhados para que o utilizador tenha uma ideia clara do que está a fazer. Os pontos desses gestos são guardados num ficheiro, criado no mesmo diretório do executável. Quando o utilizador obtém o gesto pretendido, pode encerrar o programa e aceder ao ficheiro, onde o último gesto desenhado surge como a última lista (padrão). Um padrão não é mais do que uma lista com a sequência dos pontos percorridos pelo gesto.

De seguida, é apresentado um excerto de código que define um padrão, que representa um gesto:

```
gesto_novo = [(268.0, 122.0), (273.0, 179.0), ..., (283.0, 219.0)]
```

Para utilizar este padrão, basta copiá-lo para o código da aplicação, adicionar esse padrão à instância do reconhecedor de gestos (*$1 Unistroke Recognizer*) com a função *addTemplate*. Com esta ação, o novo gesto está pronto a ser utilizado pela aplicação. Este processo é exemplificado de seguida:

```
# Inicializacao da lista que representa um gesto
gesto_novo = [(268.0, 122.0), (273.0, 179.0), ..., (283.0, 219.0)]
# Instancia o reconhecedor de gestos (Recognizer)
recon = Recognizer()
# Adiciona o padrao do gesto para ser reconhecido pelo Recognizer
recon.addTemplate('NOME_DO_GESTO', gesto_novo)
# Devolve o nome do gesto identificado, e a pontuacao do grau de confianca
(nome, pontuacao) = recon.recognize(GESTO_EFETUADO)
```

Esta aplicação permite uma forma fácil e rápida de criar novos padrões de gestos. É uma maneira de aumentar as capacidades da *framework*, pois permite diversificar a quantidade de gestos.

3.2.3 Criação de *widgets*

A utilização de *widgets* compostas pretende facilitar a vida ao programador. Estes elementos gráficos podem ser utilizados pelos programadores para começar a desenvolver aplicações, e possuem métodos que permitem a edição da informação que apresentam.

Cada *widget* foi desenvolvida tendo em conta a forma como se poderiam abstrair detalhes repetitivos da sua implementação. Um desses detalhes repetitivos é a necessidade de um botão para fechar cada *widget*, por isso, todas as *widgets* são instanciadas com este botão. Esta caraterística repete-se em todas as *widgets*, uma vez que há sempre necessidade de fechá-las. Outro pormenor útil que foi implementado é o redimensionamento automático das *widgets* compostas, e todos os seus elementos, sempre que uma delas é instanciada. Desta forma o programador poupa tempo, porque não tem de ajustar as *widgets* e todos os elementos que a compõem, sempre que redefine as suas dimensões.

Para a programação com *widgets*, procurou-se definir uma estrutura simples, que permitisse alguma flexibilidade. Para utilizar uma *widget* basta fazer a sua instanciação, e no caso de se querer utilizar dados provenientes de alguma fonte de informação, também se deve fazer a instanciação da *factory* correspondente. Também se devem inicializar os valores de posição e rotação, mas caso isso não aconteça, são invocados os valores *default* para os dois parâmetros. De seguida é apresentado um exemplo deste processo:

```
# Instanciacao da factory
factoryInstance = BaseFactory('http://www.fe.up.pt/si/')
# Instanciacao da widget, e inicializacao da posicao e rotacao
widgetInstance = BaseWidget(pos=(x,y), rot=(theta))
# Povoamento da widget com dados filtrados pela factory
# Funcao getPack() devolve o pacote com os dados filtrados
widgetInstance.addDataPack(factoryInstance.getPack())
# Adiciona a widget para visualizacao
self.add_widget(widgetInstance)
```

Esta é a estrutura básica de utilização de uma *widget*. É feita a instanciação da *factory*, utilizando o *link* da página com a informação a ser filtrada. Depois, instancia-se a *widget* com uma posição *(x, y)* e rotação *theta*. É feito o povoamento da *widget* com dados, através da função *addDadaPack*, que recebe como argumento um pacote de dados resultante da filtragem na *factory*. Por fim, a instância da *widget* é adicionada ao *desktop*. Alguns exemplos adicionais da utilização das *widgets* são apresentados no fim deste documento, no *Anexo E – Exemplos de Utilização da Framework*.

Esta sequência foi estabelecida, procurando fornecer uma metodologia intuitiva para construção de *widgets*. O objetivo é manter uma estrutura semelhante para operações semelhantes, por exemplo, o funcionamento para utilização de *widgets* é sempre igual:

1) Instanciação da *factory*.
2) Instanciação da *widget*.
3) Povoamento da *widget* com dados da *factory*.

Esta decisão vai ao encontro dos requisitos inicialmente identificados para a *framework*, modularidade e integração de elementos.

Para facilitar a utilização desta *framework*, foi elaborada uma API, que pode ser consultada no fim deste documento no *Anexo F – API*.

3.3 Caso de Estudo – SiFEUP MT

Como prova da utilidade da *framework*, foi desenvolvida uma aplicação sobre esta plataforma, baseada em algumas funcionalidades do SiFEUP, com o nome, SiFEUP MT.

O SiFEUP MT consiste numa interface, alternativa à existente, para acesso à informação disponível no site da FEUP. Esta aplicação fornece ao utilizador uma interação natural onde pode realizar pesquisas comuns no SiFEUP, como por exemplo, a ficha de aluno, o horário, ou as disciplinas que frequenta.

A aplicação foi desenvolvida para ser utilizada em estações MT de médias ou grandes dimensões, podendo ser utilizada por mais do que uma pessoa simultaneamente.

Como se pode observar na Fig.26, o SiFEUP possui muita informação acerca da faculdade, como por exemplo, alunos, docentes, notícias, etc. Por vezes esse excesso de informação confunde o utilizador, levando-o a perder demasiado tempo até encontrar a informação pretendida.

Na sua versão MT, só a informação crucial para os casos de utilização mais comuns é utilizada. A arquitetura geral desta aplicação pode ser observada na Fig.27.

Figura 26 – *Screenshot* da página principal do SiFEUP.

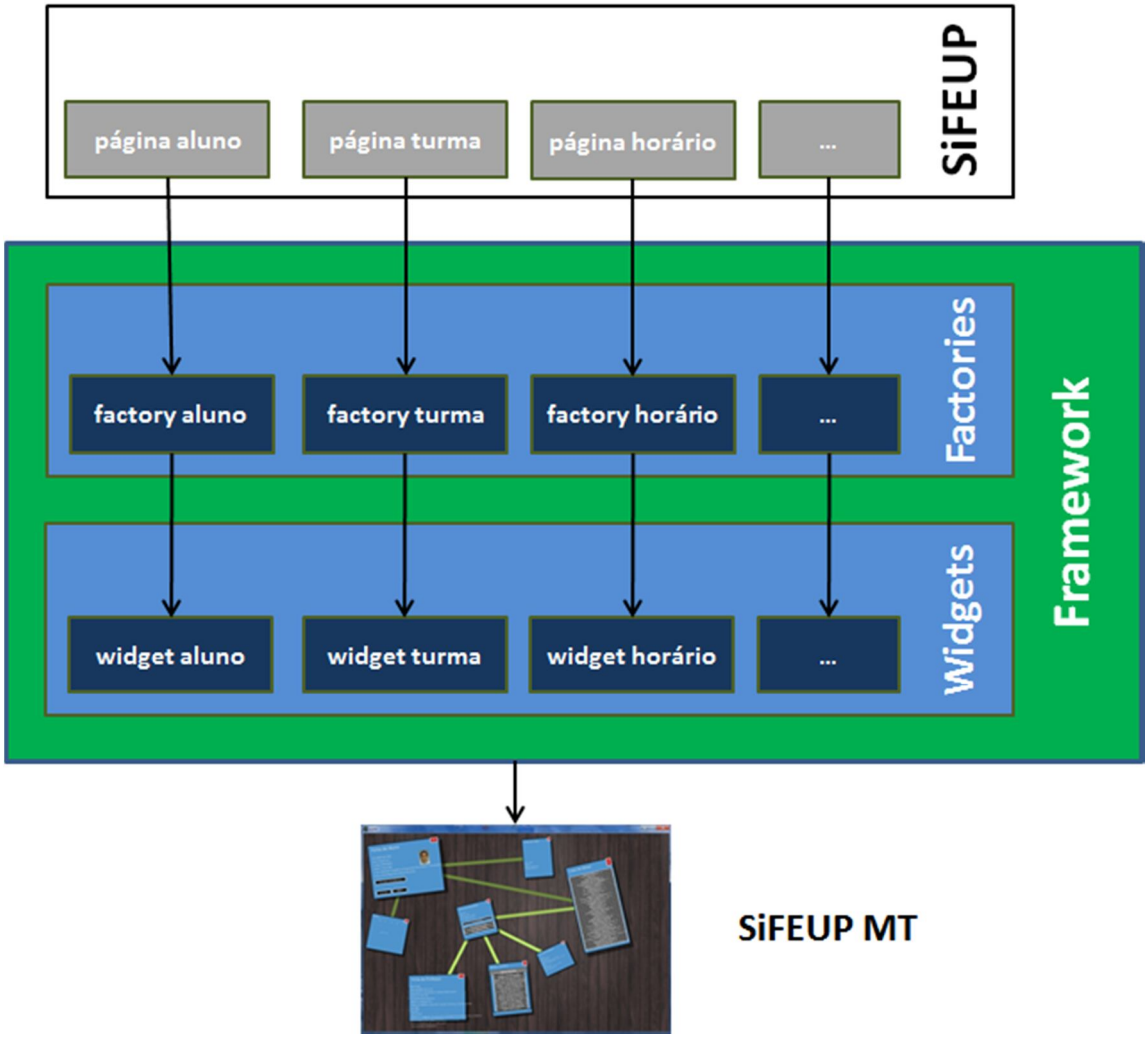


Figura 27 – Arquitetura da aplicação.

O código das páginas do SiFEUP é utilizado pelas *factories* para fazer o *parsing* de dados. Estes dados são depois enviados para as *widgets* correspondentes a cada uma das *factories*. Depois de todas as *widgets* construídas, e de programada a lógica da aplicação, temos o SiFEUP MT. Nas próximas duas páginas serão apresentadas *screenshots* do SiFEUP MT, representando alguns casos de utilização.

Como é mostrado na Fig.28 à esquerda, no ecrã inicial o utilizador deve efetuar o gesto relativo à instanciação da *widget* para autenticação (quadrado). Na mesma figura, mas à direita, podemos ver a *widget* de autenticação instanciada. Tanto a inclinação como a posição das *widgets*, são calculadas em função da posição e inclinação do gesto, de forma a orientar as *widgets* ao utilizador.

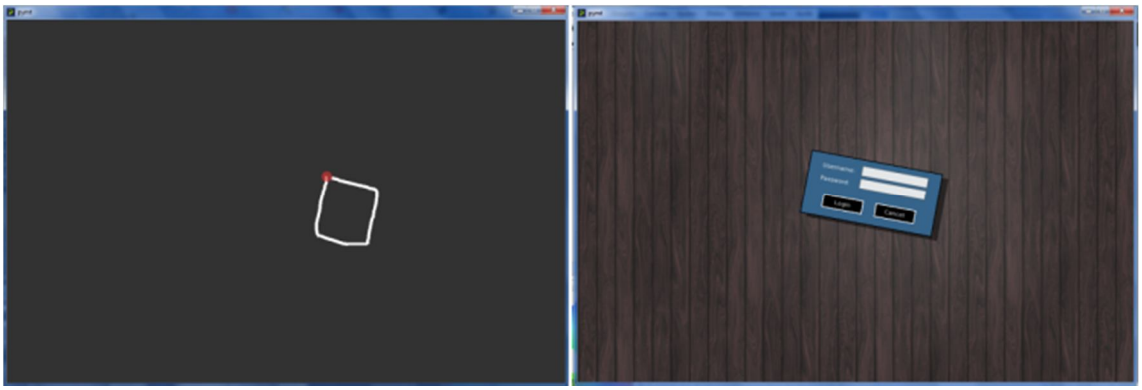


Figura 28 – Ecrã inicial do SiFEUP MT.

O processo de autenticação é mostrado na Fig.29, onde é necessário inserir o nome de utilizador e a palavra passe do estudante. Isso é conseguido com recurso à *widget* de teclado, que permite a inserção de texto.

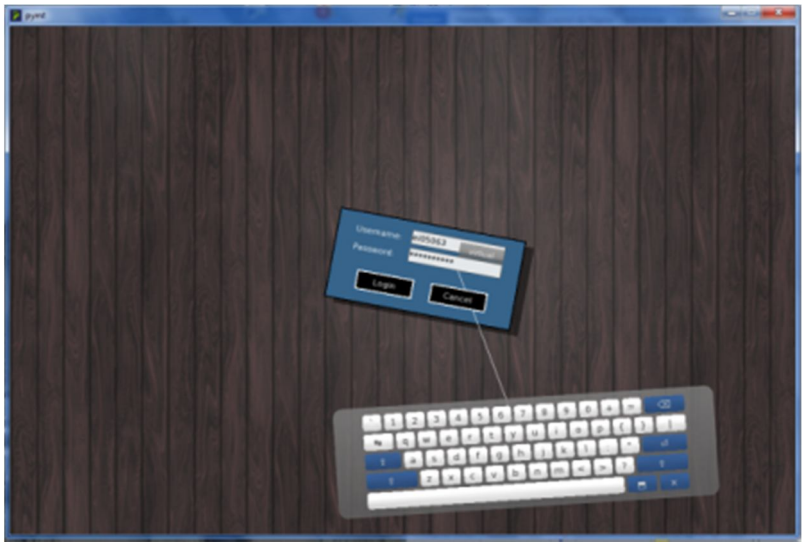


Figura 29 – Autenticação no SiFEUP MT.

Depois de feita a autenticação do estudante, surge a *widget* referente à sua ficha de aluno, Fig.30. Cada aluno poderá aceder à sua informação pessoal e consultar o horário, a ficha do curso ou o percurso académico. Dentro do percurso académico é possível conhecer as notas de cada unidade curricular e outras informações sobre as mesmas, nomeadamente, os docentes e alunos inscritos.

Figura 30 – Área pessoal do aluno.

Na Fig.31 está representado um caso de utilização mais avançado, onde o aluno pesquisa por várias informações em simultâneo. As *widgets* ligam-se entre si para representar um fluxo de utilização.

Figura 31 – SiFEUP MT.

3.3.1 *Widget* de Autenticação

A *widget* de autenticação surge assim que um utilizador realiza o gesto quadrado no ecrã da aplicação. Serve para autenticar um utilizador no sistema.

Figura 32 – Funcionamento da *widget* de autenticação.

Esta *widget* não recolhe dados do SiFEUP uma vez que a sua função é apenas autenticar um utilizador no sistema. Este procedimento tem de ser sempre realizado, uma vez que algumas informações só estão acessíveis a utilizadores da FEUP. A posição e rotação desta *widget*, tal como todas as outras, são obtidas a partir da posição e inclinação do gesto realizado. Desta forma a interface é adaptada à posição do utilizador ao redor da estação MT.

Figura 33 – Aspeto da *widget* de autenticação.

3.3.2 *Widget* de Aluno

A *widget* de aluno surge assim que um utilizador faz a sua autenticação no sistema, e disponibiliza a ficha de um dado aluno.

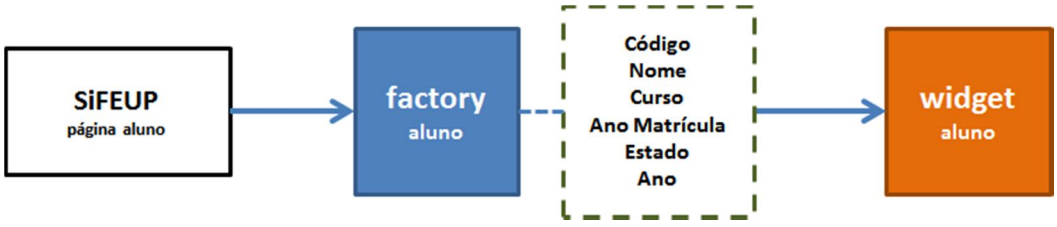


Figura 34 – Funcionamento da *widget* de aluno.

Na ficha de cada aluno é possível visualizar o código de aluno, o nome, curso, ano de matrícula, estado da inscrição e o ano que frequenta. Estas informações são construídas como *labels* e não permitem interação. No entanto, existem botões que ligam a outro tipo de informações, como o horário do aluno, ficha de curso, e ao percurso académico onde poderá consultar todas as notas.

Como já foi referido, tanto as ligações entre widgets, como o botão de fecho, encontram-se implementados em todas as *widgets* que vão ser apresentadas nas próximas páginas.

Figura 35 – Aspeto da *widget* de aluno.

3.3.3 *Widget* de Horário

A *widget* de horário surge a partir de um dos botões presentes na ficha de aluno, e fornece uma visualização do horário do aluno.

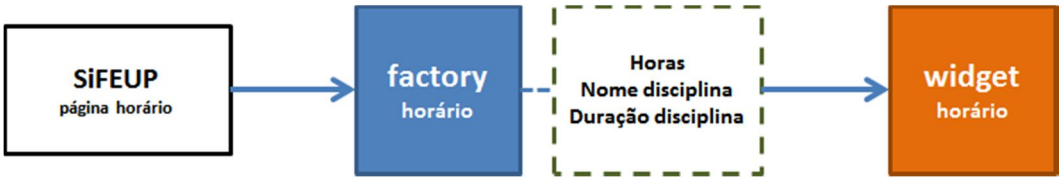


Figura 36 – Funcionamento da *widget* de horário.

Na visualização do horário, o estudante pode ver as aulas das disciplinas a que está inscrito, a hora de início e a hora a que terminam. Estas informações são construídas com *labels* e não permitem interação. No entanto, podem ser adicionadas funcionalidades como ligações às disciplinas, que pode ser feito substituindo as *labels* por botões.

Figura 37 – Aspeto da *widget* de horário.

3.3.4 *Widget* de Curso

A *widget* de curso surge a partir de um dos botões presente na ficha de aluno, e fornece uma visualização de informações úteis acerca do curso frequentado.

Figura 38 – Funcionamento da *widget* de curso.

Na visualização da ficha de um curso é possível saber a sigla do curso, o grau de formação, em que ano teve o seu inicio e também o diretor. Estas informações são construídas com *labels* e não permitem interação.

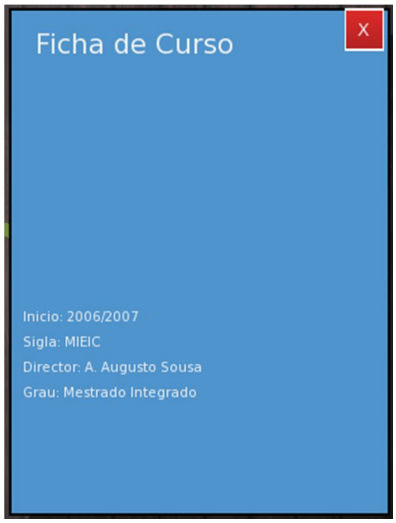


Figura 39 – Aspeto da *widget* de curso.

3.3.5 *Widget* de Percurso Académico

A *widget* de percurso académico surge a partir de um dos botões na ficha de aluno, e fornece uma listagem de todas as disciplinas em que o estudante já esteve inscrito e a nota final obtida.

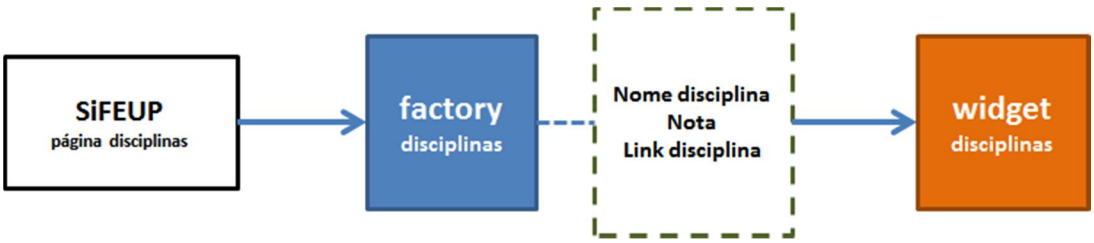


Figura 40 – Funcionamento da *widget* de percurso académico.

Na visualização do percurso académico é dada uma lista com todas as disciplinas e respetivas notas, além disso, é possível consultar cada uma dessas disciplinas. Os elementos da lista são construídos com botões que permitem instanciar uma *widget* de disciplina.

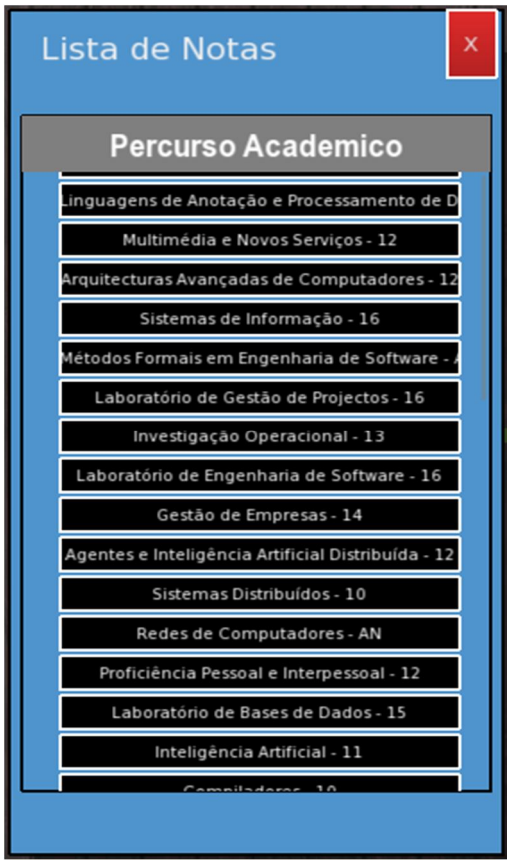


Figura 41 – Aspeto da *widget* de percurso académico.

3.3.6 *Widget* de Disciplina

A *widget* de disciplina surge a partir da *widget* percurso académico como resultado do toque numa disciplina.

Figura 42 – Funcionamento da *widget* de disciplina.

A visualização de uma disciplina disponibiliza informação sobre o nome dessa unidade curricular, o código, a sigla e a área científica onde se insere. Também é disponibilizado o conjunto dos docentes que lecionam a disciplina e a lista dos alunos inscritos.

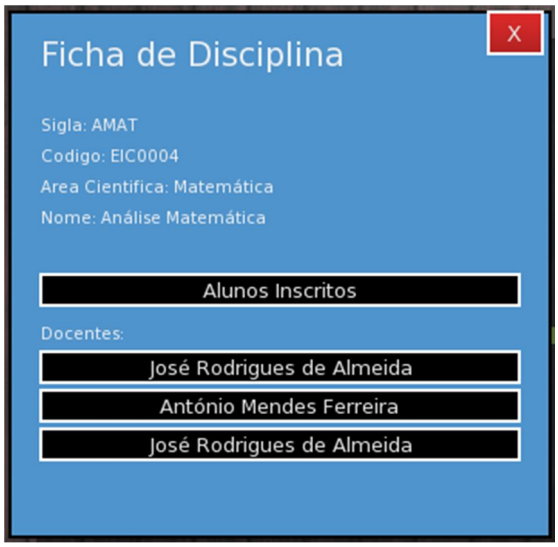


Figura 43 – Aspeto da *widget* de disciplina.

3.3.7 *Widget* de Docente

A *widget* docente surge a partir da visualização de uma disciplina quanto o utilizador toca no botão de um dos docentes.

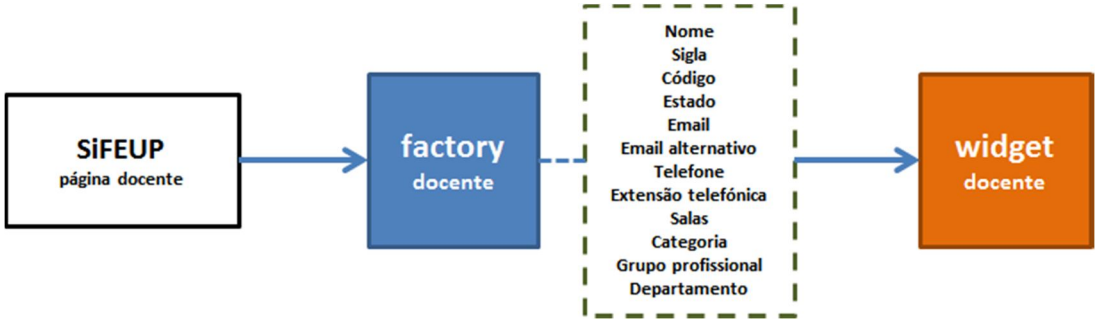


Figura 44 – Funcionamento da *widget* de docente.

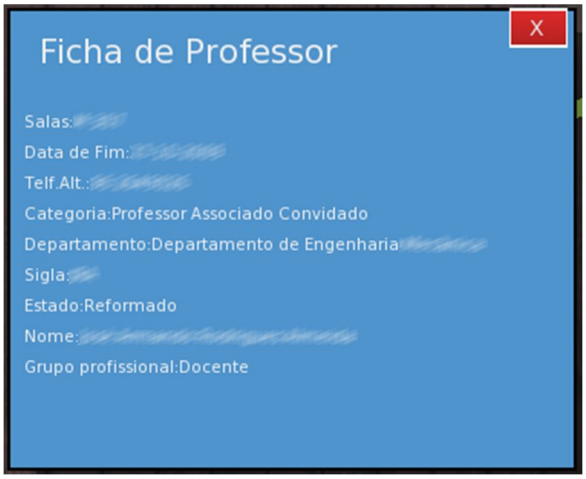


Figura 45 – Aspeto da *widget* de docente.

3.3.8 *Widget* de Lista de Alunos

A *widget* lista de alunos surge a partir da visualização de uma disciplina quando o utilizador carrega no botão que devolve a lista de alunos inscritos.

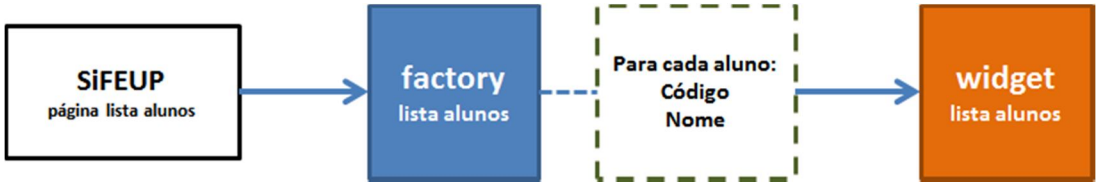


Figura 46 – Funcionamento da *widget* de lista de alunos.

Figura 47 – Aspeto da *widget* lista de alunos.

Nesta aplicação procurou-se implementar alguns dos casos de utilização mais comuns, como a consulta de horários, planos de estudo, percurso académico e unidades curriculares onde o aluno está ou esteve inscrito.

Durante o desenvolvimento foi dada grande importância à interface com o utilizador e consistência visual. Apesar das widgets serem diferentes umas das outras, foi mantido o mesmo estilo de cores e de botões, e a orientação das *widgets* segue a orientação da sua *widget* pai, mantendo o trabalho direcionado para o utilizador. A informação visualizada é realmente extraída do SiFEUP, permitindo testar casos de utilização reais.

No entanto, o objetivo principal nesta implementação era o de testar a capacidade da *framework* para auxiliar o desenvolvimento de aplicações baseadas em visualização de informação, num suporte de interação multitoque.

Como resultado, temos uma aplicação que pode ser totalmente utilizada com toques e gestos, e por vários utilizadores em simultâneo. Inclui gestos específicos para despoletar eventos, mas também permite que o programador possa criar os seus próprios protótipos de gestos e integrá-los nas aplicações.

Esta aplicação foi desenvolvida com uma ideia em mente, a possibilidade de utilizá-la no espaço público da FEUP para facilitar o acesso à informação de toda comunidade universitária. Esta capacidade foi avaliada com o recurso a testes e um inquérito, cujos resultados e conclusões vão ser apresentados no próximo capítulo.

Capítulo 4

Estudo de Aplicabilidade e Testes

Este capítulo apresenta um estudo de aplicabilidade sobre uma hipotética implementação de um conjunto de estações MT nas instalações da FEUP. O estudo foi realizado com utilizadores do SiFEUP, permitindo identificar que características mais seriam valorizadas num serviço deste tipo.

Neste capítulo também são identificados os testes realizados à aplicação desenvolvida com a *framework*, e apresenta o resultado dos mesmos, que serão discutidos. Os testes foram realizados com utilizadores finais da aplicação de forma a obter uma opinião fiável.

4.1 Estudo de aplicabilidade do SiFEUP MT

Uma vez que a aplicação desenvolvida com a *framework* tem como principais operações o acesso a informação de interesse à comunidade universitária da FEUP, convém perceber a sua opinião sobre uma eventual implementação desta tecnologia nas instalações da faculdade.

Para isso, foi enviado por *email*, um inquérito com questões relacionadas com o à vontade com as tecnologias multitoque, onde e com que serviços um sistema destes deveria ser implementado, e se haveria alguma vantagem desta tecnologia para melhorar o acesso à informação na faculdade. Foram obtidas 625 respostas ao inquérito, e os resultados são apresentados na Tab.4.

Tabela 2 – Resultados do questionário à comunidade FEUP.

P1. Já ouviu falar, ou experimentou algum equipamento informático com tecnologia multitoque?	
Sim, já experimentei.	65 %
Sim, ouvi falar, mas nunca experimentei.	29 %
Não, nunca ouvi falar.	6 %
P2. Se fossem instaladas na FEUP mesas multitoque, quais seriam os melhores locais?	
Entrada	76 %
Cantina	18 %
Biblioteca	69 %
Bar da biblioteca	11 %
Corredor do bloco B	72 %
Parque dos alunos	4 %
Bar da Associação de Estudantes	21 %

P3. Acha que poderia trazer algum benefício para a comunidade FEUP, no que toca à facilidade de acesso à informação?

Nenhum	1	2	3	4	5	Muitos
	4 %	6 %	27 %	42 %	20 %	

P4. Que tipo de conteúdos deveria oferecer?	
Acesso à ficha de aluno/docente/funcionário.	56 %
Acesso ao percurso académico.	24 %
Cursos, planos de estudos, unidades curriculares, etc.	60 %
Horários, salas, exames.	95 %
Acesso ao email.	45 %
Acesso às notícias da FEUP.	49 %
Pesquisa por alunos, docentes, unidades curriculares, etc.	47 %
P5. Que sistema acha que proporciona uma melhor experiência, uma mesa multitoque, ou um rato e teclado?	
Mesa multitoque.	48 %
Rato e teclado.	2 %
Depende da utilização.	47 %
São iguais.	2 %
P6. Preferia utilizar uma mesa multitoque (ecrã horizontal), ou um quadro multitoque (ecrã vertical)?	
Mesa (ecrã horizontal)	49 %
Quadro (ecrã vertical)	23 %
Indiferente	27 %

Pela análise dos resultados, é possível concluir que a implementação de uma estação MT na FEUP, com uma aplicação semelhante ao SiFEUP MT, seria bem acolhida. De referir que a maior parte (P1) possui algum tipo de conhecimento ou experiência com MT.

Podemos verificar que os locais mais desejados para instalação da(s) mesa(s) multitoque (P2), são também locais de grande movimento a acessibilidade, como a entrada, o corredor do bloco B ou a biblioteca, e existe uma a ideia de que esta poderia ser uma solução para melhorar o acesso à informação (P3), como se verifica pelas respostas à pergunta 3.

No que toca aos serviços disponibilizados (P4), as mais úteis do ponto de vista deste estudo são, a consulta de horários e salas, consulta de cursos e planos de estudos, e acesso às fichas de alunos e docentes. Algumas destas já estão integradas no protótipo, mas o estudo também permite conhecer as preferências dos utilizadores finais.

Também se verifica que existe a noção que a tecnologia multitoque oferece melhor usabilidade do que um rato e teclado (P5), mas também existe a noção que nem tudo deve ser integrado num sistema deste tipo, comprovado pelo número de respostas em que depende da utilização pretendida, 47% das respostas.

Para terminar, falta concluir sobre a inclinação da superfície MT (P6), onde se prefere a utilização de uma mesa, ao invés de um quadro. Este resultado pode estar relacionado com o desejo de privacidade dos utilizadores, a julgar pelas recomendações recolhidas nas respostas.

4.2 Testes

Os testes foram realizados sobre o SiFEUP MT, e tiveram como objetivo avaliar alguns parâmetros da aplicação:

- **Usabilidade –** Facilidade de utilização da aplicação.
- **Facilidade de aprendizagem da interface –** Avaliar a facilidade de aprendizagem percecionada pelos utilizadores.
- **Fiabilidade dos dados –** Determinar o grau de acerto dos dados extraídos do SiFEUP.
- **Aspeto gráfico –** O aspeto gráfico é importante nas aplicações baseadas em interfaces naturais, uma vez que a imagem é também a interface com o utilizador.

Para avaliar estes parâmetros, foi pedido a várias pessoas que utilizassem a aplicação, constituindo um total de 12 pessoas. No final da experimentação existia um questionário, onde os utilizadores identificavam a sua área de formação ou trabalho, os níveis de conhecimentos informáticos que possuem, e a avaliação que faziam da aplicação. Os resultados do questionário são apresentados na Tab.2.

Tabela 3 – Resultado do questionário aos utilizadores após os testes.

P1. Já tinha experimentado algum equipamento informático com tecnologia multitoque?						
Sim, tenho um.						50 %
Sim, mas não tenho nenhum.						50 %
Não.						0 %
P2. Achou a aplicação fácil de utilizar?						
Difícil	1	2	3	4	5	Fácil
	0 %	0 %	17 %	67 %	17 %	
P3. A aprendizagem de utilização da interface (gestos, toques, manipulação dos objetos, etc) foi rápida?						
Lenta	1	2	3	4	5	Rápida
	0 %	0 %	17 %	42 %	42 %	
P4. Gostou do aspeto gráfico da aplicação?						
Pouco atrativo	1	2	3	4	5	Muito atrativo
	0 %	0 %	17 %	67 %	17 %	
P5. Quanto à informação visualizada, era credível?						
Sim, os dados estavam corretos.						83 %
Mais ou menos, havia algumas inconsistências.						17 %
Não, havia dados errados.						0 %
P6. No geral, como foi a experiência de utilização?						
Má	1	2	3	4	5	Boa
	0 %	8 %	17 %	50 %	25 %	
P7. A instalação de uma mesa multitoque na FEUP, com uma aplicação deste tipo, teria alguma utilidade?						
Pouco	1	2	3	4	5	Muito
	0 %	0 %	0 %	67 %	33 %	

Pela análise dos resultados dos testes, pode-se concluir que o protótipo apresentado obteve bons resultados.

Todos os utilizadores já tinham experimentado algum equipamento com MT (P1), portanto, já possuíam alguma experiência de utilização. Os resultados também foram bastante satisfatórios ao nível da usabilidade (P2), da facilidade de aprendizagem da interface (P3) e do aspeto gráfico (P4), não havendo nenhum utilizador a responder negativamente a estas perguntas.

A pergunta relacionada com a informação visualizada (P5) permite aferir o trabalho realizado com as *factories*. Pelas respostas obtidas, grande parte da informação é credível, mas ainda seria preciso melhorar este aspeto, uma vez que a fiabilidade da informação é uma medida muito importante numa aplicação deste tipo.

No geral a experiência de utilização (P6) foi boa para a maioria dos utilizadores, e também se pode verificar que a grande maioria (P7) vê utilidade na instalação de uma estação MT na FEUP, com uma aplicação deste género.

Durante a fase de testes, foi pedido aos utilizadores que realizassem três casos de utilização típicos, começando com um acesso ao horário, seguido de um acesso à nota de uma disciplina, e por último, um acesso à lista de alunos inscritos nessa disciplina. Estes casos foram realizados no SiFEUP e no SiFEUP MT, onde foram registados os tempos de execução das tarefas. A perceção de melhoria foi avaliada com uma pergunta, por cada caso de utilização, onde foi perguntado, que sistema oferecia a melhor experiência para cada caso. Os resultados são apresentados na Tab.3.

Tabela 4 – Resultado das medições de tempo nos três casos de utilização.

C1. Relativamente ao primeiro caso, pesquisa do horário, qual acha que oferece a melhor utilização?		**Tempo médio no SiFEUP**
SiFEUP.	8 %	4,8 s
SiFEUP MT.	83 %	**Tempo médio no SiFEUP MT**
Indiferente.	8 %	2,3 s
C2. Relativamente ao segundo caso, procurar a nota de uma disciplina, qual acha que oferece a melhor utilização?		**Tempo médio no SiFEUP**
SiFEUP.	17 %	15,4
SiFEUP MT.	58 %	**Tempo médio no SiFEUP MT**
Indiferente.	25 %	15,5
C3. Relativamente ao terceiro caso, encontrar os alunos inscritos numa dada disciplina, qual acha que oferece a melhor utilização?		**Tempo médio no SiFEUP**
SiFEUP.	17 %	40,6 s
SiFEUP MT.	75 %	**Tempo médio no SiFEUP MT**
Indiferente.	8 %	32,8

Pela análise dos resultados pode-se concluir que o SiFEUP MT permite poupar alguns segundos na maioria dos casos testados (C1 e C3), mas também melhora a experiência dos utilizadores. Essa melhoria não se nota no segundo caso (C2), o que pode ser explicado pelo facto de que a

experiência levada a cabo foi o primeiro contacto dos utilizadores com o SiFEUP MT, deixando espaço para se deduzir que com alguma habituação, o resultado seria melhor.

Estes resultados ajudam a sustentar o facto lançado anteriormente neste documento, que o SiFEUP apresenta alguma saturação de informação, que tende a prejudicar a procura de algumas informações. O SiFEUP MT apresenta uma interface mais limpa e utilitária, onde só a informação mais vezes utilizada é apresentada.

Capítulo 5

Conclusões

A tecnologia MT oferece um paradigma de interação diferente do que é oferecido pelos sistemas habituais com rato e teclado. Existe mais interação e a aprendizagem é natural, alargando o número de utilizadores que pode utilizar o computador. Desta forma melhora-se o acesso à tecnologia, facilitando a sua utilização a infoexcluídos, idosos, e a alguns tipos de deficiência, que podem usufruir de uma forma mais natural de manipular a informação.

Com a mudança de paradigma na interação, surge também uma mudança de paradigma nas aplicações, que são mais orientadas às expetativas dos utilizadores.

Com o sucesso das interfaces de toque nos dispositivos móveis a cativar milhões de utilizadores, houve um aumento do interesse por este tipo de interfaces. Os ganhos com a interatividade são claros, e a própria experiência é visualmente mais apelativa. A satisfação dos utilizadores é patente nas exibições públicas deste tipo de tecnologia, onde desperta a curiosidade das pessoas, conseguindo arrancar sorrisos a quem experimenta. Em última análise, esta é uma ótima métrica para o sucesso.

5.1 Apreciação do Trabalho Realizado

Uma das componentes desta dissertação foi investigar as tecnologias mais utilizadas na área das interfaces naturais. Esta investigação abrangeu os equipamentos de *hardware* mais comuns na captação de toques e gestos, as técnicas mais utilizadas no processamento de imagem para a produção de eventos gestuais, mas também algumas ferramentas de *software* utilizadas para desenvolver aplicações. A investigação permitiu obter uma visão abrangente do estado da arte desta tecnologia, das suas aplicações e vantagens em algumas situações, e do tremendo crescimento da indústria multitoque nos últimos anos, com tendência para continuar nos próximos anos.

Outra componente da dissertação é a implementação de uma *framework* que permita o desenvolvimento de aplicações MT, baseadas em serviços fornecidos pelo SiFEUP. A *framework* é composta por um conjunto de *widgets* que permitem diferentes representações da informação. A proveniência da informação está orientada para fontes HTML, existindo uma *factory* de dados que extrai essa informação e a passa, segundo um formato pré-definido, para as *widgets*. As tarefas relacionadas com a representação visual e relação entre *widgets* estão contempladas na plataforma de modo que seja possível flexibilizar o desenvolvimento das aplicações. Com esta plataforma de desenvolvimento o programador tem acesso a uma base que pode aproveitar para começar um novo projeto.

Depois existe a componente de teste, cujo principal objetivo passa por testar a aplicação desenvolvida como prova de conceito da utilidade da *framework*. Neste caso, foi desenvolvido um portal alternativo à página da FEUP na *internet*. Este portal apresenta uma interface totalmente orientada ao toque, sem necessidade de utilização do rato ou teclado. O SiFEUP MT permite a um aluno da FEUP um acesso rápido a informação como horários, professores, percurso académico, etc. Os testes foram realizados junto de alunos da FEUP, permitindo avaliar as qualidade da aplicação, e identificar que funcionalidades devem ser melhoradas ou adicionadas.

Uma aplicação deste tipo seria uma solução interessante para implementar na FEUP (ou outra instituição) sobre um suporte físico como uma estação multitoque de médias ou grandes dimensões. Para avaliar essa hipótese, realizei um estudo junto da comunidade universitária da FEUP, que permitiu conhecer o grau de disponibilidade para a adoção de uma solução deste género, que tipo de serviços deveriam ser disponibilizados, e quais os locais onde seria mais útil. Estes dados podem ser utilizados num um eventual trabalho futuro, para estudar a viabilidade de um projeto deste tipo.

5.2 Trabalho Futuro

Esta área tem sido alvo de uma atenção crescente por parte da indústria informática e meio académico, que apostam mais na investigação em novos paradigmas de interação.

A aplicação criada para prova da utilidade da *framework* é sem dúvida interessante, e demonstra que é possível desenvolver aplicações deste tipo com esta plataforma, mas existe espaço para crescimento.

A ferramenta de prototipagem de gestos poderia ser melhorada ao nível do número de toques utilizados, uma vez que só permite um toque de cada vez, estando a prototipagem limitada a gestos simples. A prototipagem de gestos complexos envolvendo mais do que um toque aumentaria as possibilidades de interação das aplicações. A interface para a prototipagem também poderia ser melhorada, nomeadamente, pela disponibilização de uma interface gráfica que guie o utilizador através de todo o processo.

O leque de *widgets* compostas é interessante no contexto em que foi utilizado, mas é algo limitado. Esta limitação poderia ser reduzida alargando a variedade de *widgets* de forma a cobrir ainda mais casos de representação de informação.

A juntar a estas ideias, a inclusão de um motor de animações. Não são fundamentais para representar informação, mas melhoram a experiência de utilização.

Para além do desenvolvimento de aplicações MT para *desktops* de médias e grandes dimensões, este trabalho também permitiu desenvolver interesse na área de desenvolvimento para dispositivos móveis. Nos dispositivos móveis a colaboração entre utilizadores num mesmo aparelho é muito baixa ou mesmo inexistente, e o espaço limitado exige outras decisões no posicionamento dos elementos gráficos, mas por outro lado, as aplicações têm um caráter mais utilitário, fruto da mobilidade oferecida e maior frequência de utilização.

Como referi anteriormente, o mercado dos dispositivos móveis mantém um crescimento forte em número de vendas, principalmente impulsionado pelas vendas de *smartphones* que começam a substituir os telemóveis como *standard* da mobilidade [MFF08]. Portanto, cada vez mais pessoas, e naturalmente alunos universitários, possuem *smartphones*, o que faz deles um potencial grupo de utilizadores de um hipotético projeto semelhante ao SiFEUP MT, mas orientado a dispositivos móveis. Na hipótese de coexistirem as duas versões, faria sentido pensar numa interligação dos dois sistemas, em que, por exemplo, diversas informações poderiam ser consultadas no terminal de grandes dimensões por conveniência de espaço, e enviadas diretamente para o dispositivo móvel para aproveitar a mobilidade e privacidade.

Esta é sem dúvida uma tecnologia de futuro, onde ainda poderá haver mais avanços à medida que a comunidade de programadores desenvolve novas ideias, e novas formas de tirar partido das interações naturais aplicadas à tecnologia. O segredo está no pensar simples, pensar no nosso dia a dia, e aí veremos que existem inúmeras aplicações para a tecnologia multitoque, à espera de serem descobertas. Seja o que for que o futuro reserva a esta tecnologia, parece prometedor.

Referências

[AKA08] Alikutty K. A. *Multitouch, A Seminar Report*. School Of Engineering Cochin University Of Science & Technology. Novembro 2008

[AlR08] Hajad Mohammed Alamri, Asma Saeed Al Raiza. *Seminar of Multitouch Screen Technology*. King Khalid University. 2008

[BIH08] Alex Butler, Shahram Izadi, Steve Hodges. *SideSight: Multi-"touch" Interaction Around Small Devices*. UIST'08 (ACM Symposium on User Interface Software and Technology). Outubro 2008

[Bro08] Stuart F. Brown. *Hands-on Computing*. Scientific American. Julho 2008

[BUX09] Bill Buxton. *Multi-Touch Systems that I Have Known and Loved*. Bill Buxton Website. Outubro 2009. http://www.billbuxton.com/multitouchOverview.html

[CDG98] Chi-Cheng P. Chu, Tushar H. Dani e RajitGadh. *Evaluation of virtual reality interface for product shape designs*. IIE Transactions (1998). Fevereiro 1998

[Cro07] Jeff Croft. *Frameworks for Designers*. A List Apart: CSS, HTML and XHTML, Project Management and Workflow. Junho 2007. http://www.alistapart.com/articles/frameworksfordesigners

[ELI08] ELI 7 Things You Should Know. *7 Things you Should Know About Multi-touch Interfaces*. Educause Learning Initiative. Maio 2008

[GeW10] *Gesture Works, True Multitouch for Flash and Flex*. Gestureworks. Dezembro 2010. http://gestureworks.com/features/open-source-gestures/

[Gro10] NUI Group. *NUI Group Page*. Novembro 2010. http://nuigroup.com/go/

[HaB01] Christian von Hardenberg, François Bérard. *Bare-Hand Human-Computer Interaction*. Proceedings of the ACM Workshop on Perceptive User Interfaces, Orlando, Florida, USA. Novembro 2001

[HDV10] Thomas Hansen, Christopher Denter, Mathieu Virbel. *Using the PyMT toolkit for HCI Research*. Forum on Tactile and Gestural Interaction. Junho 2010

[IHT08] Shahram Izadi, Steve Hodges, Stuart Taylor, Dan Rosenfeld, Nicolas Villar, Alex Butler, Jonathan Westhues. *Going Beyond the Display: A Surface Technology with an Electronically Switchable Diffuser*. UIST'08 (ACM Symposium on User Interface Software and Technology). Outubro 2008

[JKGB07] Sergi Jordà, Martin Kaltenbrunner, Gunter Geiger, Ross Bencina. *The Reactable**. TEI'07 (Tangible, Embedded and embodied Interaction). Fevereiro 2007

[Jor03] Sergi Jordà. *Sonigraphical Instruments: From FMOL to the reacTable*. NIME'03 (Proceedings of the 2003 Conference on New Interfaces for Musical Expression). Maio 2003

[Lee11] Duke Lee. *The State of the Touch-Screen Panel Market in 2011*. SID'11 (Society for Information Display). Maio 2011

[Li10] Yang Li. *Protractor: A Fast and Accurate Gesture Recognizer*. CHI'10 (ACM Conference on Human Factors in Computing Systems). Abril 2010

[Liu10] Qian Liu. *TUIO, Touchlib, reacTIVision and Community Core Vision*. University of California, Santa Barbara. 2010

[MFF08] Troy Mastin, Meggan Friedman, Brian Freiwald. *Interactive Marketing Survey*. William Blair & Company, L.L.C. Industry Report. Julho 2008

[MFW10] Meredith Ringel Morris, Danyel Fisher, Daniel Wigdor. *Search on surfaces: Exploring the potential of interactive tabletops for collaborative search tasks*. Information Processing and Management Journal. Novembro 2010.

[MoN03] Thomas B. Moeslund, Lau Nørgaar. *A Brief Overview of Hand Gestures used in Wearable Human Computer Interfaces*. Laboratory of Computer Vision and Media Technology of Aalborg University, Denmark. 2003

[NUI09] NUI Group Authors. *Multi-Touch Technologies (1st edition)*. NUI Group. Maio 2009

[Ove07] *Touch Technology Overview*. NCR Corporation. Abril 2007

[Set08] Seth (cerupcat). *Getting Started WithMultiTouch*. NUI Group Community Forums. Maio de 2008. http://nuigroup.com/forums/viewthread/1982/

[ShH10] Orit Shaer, Eva Hornecker. *Tangible User Interfaces: Past, Present and Future Directions*. Foundations and Trends® in Human-Computer Interactions. 2010

[SKO09] Johannes Schöning, Antonio Krüger, Patrick Olivier. *Multi-touch is Dead, Long live Multi-touch*. CHI'09 (ACM Conference on Human Factors in Computing Systems). Abril 2009

[Tho07] Michael Thornlund. *Gesture Analyzing for Multi-Touch Screen Interfaces*. Lulea University of Technology. 2007

[WHF10] Annette Wiedemann, Vojtech Horna, Pedro Fernandes. *DISPLAX unveils Multitouch 'Skin' that Transforms Surfaces into Interactive Screens*. ISE'10 (Integrated Systems Europe). Fevereiro 2010

[WWL07] Jacob O. Wobbrock. Andrew D. Wilson. Yang Li. *Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes*. UIST'07 (ACM Symposium on User Interface Software and Technology). Outubro 2007

[WWL10] Jacob O. Wobbrock, Andrew D Wilson, e Yang Li. *$1 Unistroke Recognizer in Java Script*. University of Washington and Microsoft Research. Abril 2010 http://depts.washington.edu/aimgroup/proj/dollar/

Anexo A

Instalação do Multitoque em Windows 7

O Windows 7 traz de raiz algumas funcionalidades de MT, mas para utilizá-las é necessária uma instalação da *driver*. O processo é simples.

1. Fazer o download da *driver* para ativar o multitoque no Windows 7, em `http://multitouchvista.codeplex.com/releases/28979/download/72207`, e extrair os ficheiros do arquivo comprimido.
2. Depois da extração dos ficheiros, navegar até ao diretório **Driver\x32** (**x64**, se o processador do computador utilizar instruções de 64 bits), e executar o *script* **Install driver.cmd**. Esta operação vai instalar os ficheiros necessários.
3. Depois da instalação, voltar à raiz da pasta descomprimida e correr o executável **Multitouch.Service.Console.exe** para iniciar o serviço MT. A janela de comandos aberta deve permanecer aberta para não encerrar o serviço.
4. De seguida, é necessário correr o **Multitouch.Driver.Console.exe**, e também esta janela de comandos deve permanecer aberta.
5. Poderá ser necessário configurar o tipo de *input* para os toques, executando o ficheiro **Multitouch.Configuration.WPF.exe**. Há duas opções para o tipo de *input*, **MultipleMice** e **TUIO**. A primeira deve ser selecionada se desejarmos utilizar múltiplos cursores do rato para simular vários toques, caso não exista *hardware* adequado. A opção TUIO deve ser selecionada quando na presença de *hardware* MT.
6. Neste momento as funcionalidades de toque já devem estar disponíveis. Agora é possível executar as funcionalidades deste sistema operativo com o toque dos dedos.

Anexo B

Instalação do Multitoque em Linux

A instalação do multitoque em Linux pode ser realizada recorrendo a uma máquina virtual, na eventualidade de não haver a possibilidade de uma instalação Linux de raiz. Durante esta dissertação foi utilizada uma máquina virtual criada com o Virtual Box.

De seguida são explicados os processos de instalação e configuração de duas tecnologias MT testadas na plataforma Linux.

B.1 – Processing

Estando a máquina virtual criada e o Ubuntu devidamente instalado, é necessário proceder a uma alteração nas definições da rede para nos certificarmos que as mensagens TUIO, provenientes do CCV, são enviadas para a máquina virtual e não para o sistema operativo hospedeiro. Para isso devem ser seguidos os seguintes passos:

1. Aceder às definições da máquina virtual onde está instalado o Ubuntu, e no separador **Rede**, ativar um segundo adaptador que esteja associado à opção **Host-only Adapter**. Clicar em OK e sair.
2. Arrancar o Ubuntu instalado na máquina virtual, e na linha de comandos inserir o comando **ifconfig** para obter o endereço IP que está a ser utilizado pela máquina virtual. Essa informação pode ser obtida no grupo **eth1**, referente à configuração feita no passo anterior.
3. Em Windows, ir à pasta onde está instalado o CCV, abrir o ficheiro **\data\config.xml** com um editor de texto (por ex: Notepad++), e na tag **LOCALHOST**, inserir o endereço IP da máquina virtual anteriormente obtido. Gravar as alterações e fechar o ficheiro. Executando agora o CCV, é possível verificar que os dados de toques recolhidos estão a ser enviados através de mensagens TUIO para o endereço IP da máquina virtual. A porta de comunicação deve permanecer inalterada.

4. No Ubuntu, abrir um exemplo em Processing que faça uso de mensagens de *input* no protocolo TUIO. Se todos os passos tiverem sido bem executados, já deverá estar a funcionar.

B.2 – Multi-Pointer X (MPX)

A utilização de múltiplos cursores foi configurada na máquina virtual com a instalação do Ubuntu. O processo para ativar e controlar um segundo cursor é simples:

1. Arrancar o Ubuntu na máquina virtual e conectar um segundo rato no computador (neste caso foi utilizada uma porta USB).
2. No separador **Dispositivos** do Virtual Box, ir até aos dispositivos USB e selecionar o 2º rato introduzido.
3. No terminal do Ubuntu, introduzir o comando **xinput list** dá-nos uma lista dos dispositivos de input presentes na máquina virtual, com os ID's correspondentes. É possível identificar os dois ratos existentes, bem como o teclado. O que pretendemos é associar o segundo rato como dispositivo master de input.
4. Para criar um novo cursor, basta introduzir o comando **xinput create-master [nome para o dispositivo]**. É criado um cursor, mas sem nenhum diapositivo de input associado.
5. Para associar um dispositivo de input é necessário introduzir o comando **xinput reattach [ID do dispositivo slave] [ID do dispositivo master]**. Agora já deve ser possível utilizar dois cursores em simultâneo no Ubuntu.

É possível obter informações adicionais sobre esta configuração em `https://wiki.ubuntu.com/X/MPX`.

Anexo C

Estudo das Tecnologias

Compilação das características de cada uma das plataformas de desenvolvimento de aplicações multitoque estudadas durante a dissertação.

C.1 – PyMT

Linguagem de programação: Python
Desenvolvimento: Notepad++ (livre)
É uma biblioteca *open source* desenvolvida para facilitar o desenvolvimento de aplicações multitoque altamente interativas, e pode ser utilizada em múltiplas plataformas (Linux/OSX/Win).

Características:

- Controlo sobre o fluxo de execução das aplicações, gestão manual da cache (apesar de também ser automática), e tem uma *framework* que permite a manipulação de dispachers de eventos, tratamento de exceções, facilita os cálculos geométricos, permite criação de novos gestos e comparação entre eles, carregamento assíncrono que permite utilizar dados mesmo que ainda não estejam disponíveis, formato de ficheiro OBJ para manipulação 3D, abstração para tratar de texturas OpenGL, e operações com vetores.
- Funções para desenhar em OpenGL.
- Permite a inclusão de som/música, utilização de uma câmara, utilização de imagens, utilização de vídeo, permite Scalable Vetorial Graphics (SVG), e desenho de texto.
- Desenho de curvas de Bezier, manipulação de cores, ignora toques em áreas específicas do ecrã, e *inputs* TUIO.
- Manipulação dos atributos de cada toque, e listagem dos *input providers*.
- Possui uma biblioteca para cálculos de matrizes 4×4.

- Interação cinética, *popup's*, tabs em *widgets*, input de texto multilinha, e teclado virtual com vários *layouts* com *spell checker*. Funciona a base de *widgets*, existinto vários *layouts*, ajustar as bordas, e dispor *widgets* por grelha. Vários tipos de botões, conversão de símbolos em *widgets*, *widgets* movíveis, *widgets* com dois lados, labels, *widgets* para objetos fiduciais, manipulação de vetores, e *widgets* com imagens SVG.
- Animação de objetos e CSS, e vários temas.

Documentação adicional:

- Página oficial - `http://pymt.eu/`
- API - `http://pymt.eu/docs/api/`
- Wiki - `http://pymt.eu/wiki/`

C.2 – Multitouch Vista

Linguagem de programação: C#
Desenvolvimento: Microsoft Visual Studio 2008/2010 (pago)
Esta *framework* permite desenvolver aplicações para Windows 7. O desenvolvimento para esta plataforma é atrativo uma vez que este é o sistema operativo mais amplamente utilizado a nível mundial pelos utilizadores comuns, e como tal, é garantia de um enorme público a utilizar as aplicações. Outra vantagem é a possibilidade de utilizar integralmente este sistema operativo através de toques.

Características:

- Suporte para manipulação e inércia nas aplicações.
- Suporta vários gestos como *tap / double tap, zoom, rotate, press and tap, selection / drag, panning with inertia, two-finger tap, press and hold, flicks*.
- Gestão e interpretação das mensagens dos gestos Windows Touch.

Documentação adicional:

- Guia de programação e exemplos de código - `http://msdn.microsoft.com/en-us/library/dd562197(v=VS.85).aspx`
- Tutorial - `http://www.dotnetspark.com/kb/1666-windows-7-multitouch-application-development.aspx`

C.3 – Flash/Flex

Linguagem de programação: Action Script 3.0
Desenvolvimento: CS5 (pago)

A *framework* Flash para MT permite o desenvolvimento de aplicações altamente interativas e visualmente muito interessantes. Permite o desenvolvimento de aplicações para desktops e para plataformas móveis.

Características:

- Possibilidade de criar os próprios gestos e lançar eventos.
- Possui um conjunto de classes para tratamento de eventos de gestos
- É possível criar aplicações que respondem a eventos de toque básicos ou criar *event listeners* utilizando os tipos de eventos da classe.
- Possibilidade de controlar o processamento de *event listeners*.
- O número de pontos de toque é apenas limitado pelo *hardware* utilizado.
- A API de gestos funciona nos *browsers* em Windows, mas não em Mac.

Possui uma API para o tratamento da interação do utilizador através de gestos, que possui as seguintes classes:

- **flash.events.TouchEvent** – Permite o desenvolvimento de aplicações onde o utilizador utilize eventos de toque básico (um toque único de cada vez).
- **flash.events.GestureEvent** – Permite o desenvolvimento de aplicações onde o utilizador utilize eventos de gestos (dois ou mais toques de cada vez).
- **flash.events.TransformGestureEvent** – Permite o controlo de eventos de gestos complexos.
- **flash.events.GesturePhase** – É uma classe de enumeração de constantes para utilizar com as classes GestureEvent, PressAndTapGestureEvent e TransformGestureEvent. Estes valores servem para mapear o início, progresso e terminação de um evento de gesto.
- **flash.events.PressAndTapGestureEvent** – Permite o controlo sobre eventos do tipo *press-and-tap* (geralmente para abrir um *popup* de contexto)
- **flash.events.EventDispatcher.addEventListener** – Esta classe permite que qualquer objeto de uma *display list* possa ser utilizado com um evento, de forma a tirar partido da interface IEventDispatcher.

Documentação adicional:

- Página oficial - `http://gestureworks.com/`
- Manual de referência - `http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/ui/Multitouch.html`
- Documentação Flash/Flex - `http://www.adobe.com/devnet/flex/documentation.html`

C.4 – Processing

Linguagem de programação: Processing
Desenvolvimento: Processing
Processing é uma linguagem de programação *open source*, que pode ser utilizada em Windows, Linux e Mac.

Caraterísticas:

- Facilita o processo de criação de aplicações visualmente sofisticadas.
- Possui implementação para controlar cores, luzes, câmaras, imagens, *rendering*, tipografia e cálculos matemáticos.
- Possibilidade de definir várias formas como primitivas 2D/3D, curvas, vértices, etc.

Documentação:

- Página oficial - `http://processing.org/`
- Processing API - `http://processing.org/reference/`

Anexo D

Preparação do Hardware/Software

Este anexo resume o processo de montagem e calibração do equipamento utilizado durante a dissertação.

Também é explicado o processo de calibração do CCV, e é introduzida a programação da aplicações com o PyMT.

D.1 – Montagem do equipamento

Para a configuração da mesa foi utilizada a técnica FTIR. Um *setup* habitual que faça uso desta tecnologia utiliza um projetor para produzir a imagem visível ao utilizador, uma câmara com a capacidade de captar luz IV, uma placa retangular de acrílico que funciona como tela da projeção, e um conjunto de leds para emitir luz IV.

No *setup* deste projeto utilizou-se uma câmara Playstation Eye, a mesma da consola Playstation 3. A decisão sobre esta câmara deve-se à sua ótima relação qualidade/preço. Apenas foi necessário substituir o filtro IV que acompanha a câmara, e instalar no computador as *drives* necessárias para o controlo do acessório.

Os *leds* foram acoplados a duas calhas que foram colocadas ao comprimento de um acrílico retangular. É neste que acontece o fenómeno FTIR captado pela câmara. Os *leds* foram colocados sobre o vidro para permitir uma melhor distribuição da luz IV por toda a superfície. Sendo assim, a imagem é projetada para a tela de acrílico. Por cima desta tela está a placa de vidro com os leds. A câmara ligada ao computador capta a imagem iluminada com luz IV, que corresponde ao efeito FTIR causado pelos toques dos dedos do utilizador na camada de vidro.

Depois de termos o hardware devidamente preparado, é tempo de passar para o software que também necessita de algumas afinações para obter o melhor resultado possível.

D.2 – Calibração do CCV

Como qualquer sistema que utilize câmaras, é preciso proceder a alguma calibração da imagem. A calibração é fundamental para a precisão dos toques, pois certifica que estes estão alinhados com os elementos gráficos no ecrã.

A solução de software adotada para a camada de *input* foi o Community Core Vision (CCV). Uma das vantagens desta solução é a ferramenta de calibração embebida. Esta ferramenta permite fazer o alinhamento da *bounding box* (janela na qual os toques serão processados) com a superfície de toque, e a adição de pontos de calibração de modo a aumentar a precisão. A dimensão da *bounding box* vai depender do tamanho da superfície de interação que se pretende. Normalmente estica-se para acomodar toda a superfície, mas em alguns casos, pode-se pretender interagir com uma pequena área. O número de pontos de calibração depende da precisão de toque desejada, ou então da dimensão da *bounding box*. Quanto maior a concentração de pontos maior é a precisão.

Este processo pode ser observado na Fig.48. Neste caso são utilizados 20 pontos. O utilizador tem de pressionar durante alguns segundos o ponto assinalado a vermelho, repetindo o mesmo para todos os pontos. A *bounding box* define a área que será considerada para identificação dos toques. Neste exemplo são deixados de fora os cantos da superfície.

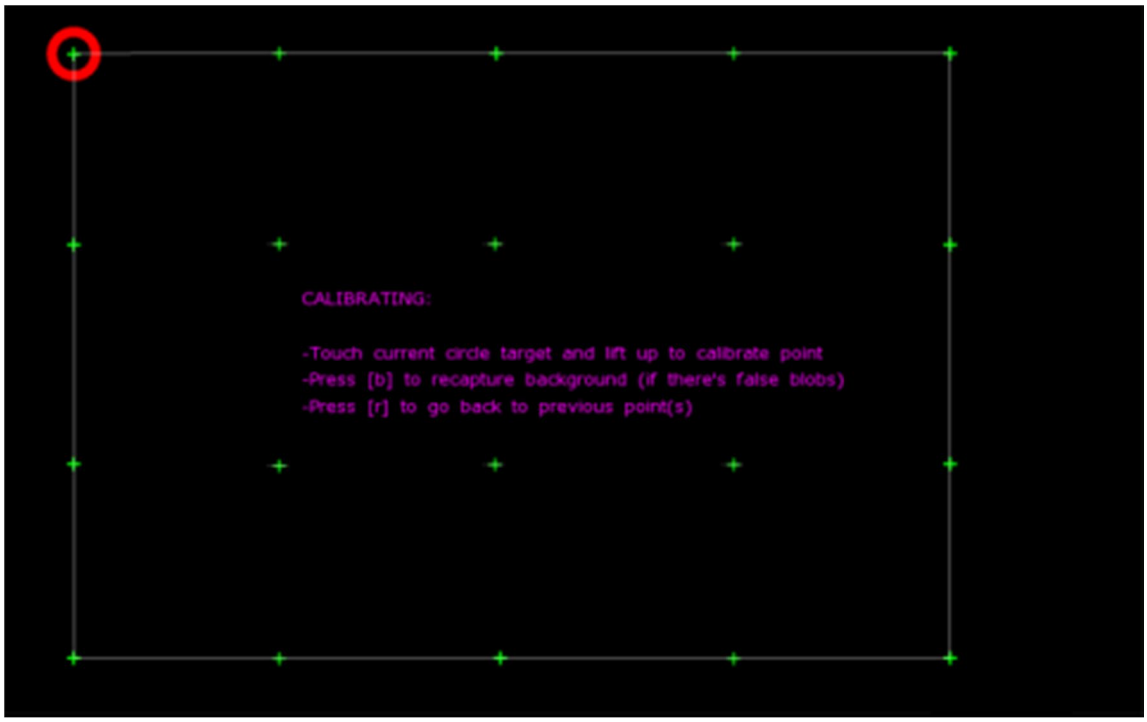


Figura 48 – Ferramenta de calibração do CCV.

A calibração pode ser demorada. Baseia-se num processo de tentativa e erro, mas é fundamental para conseguir extrair o máximo do hardware. O CCV possui ainda uma série de parâmetros editáveis ao nível do processamento da imagem. Essa informação está visível na Fig.49, e explicada em detalhe no texto que lhe segue.

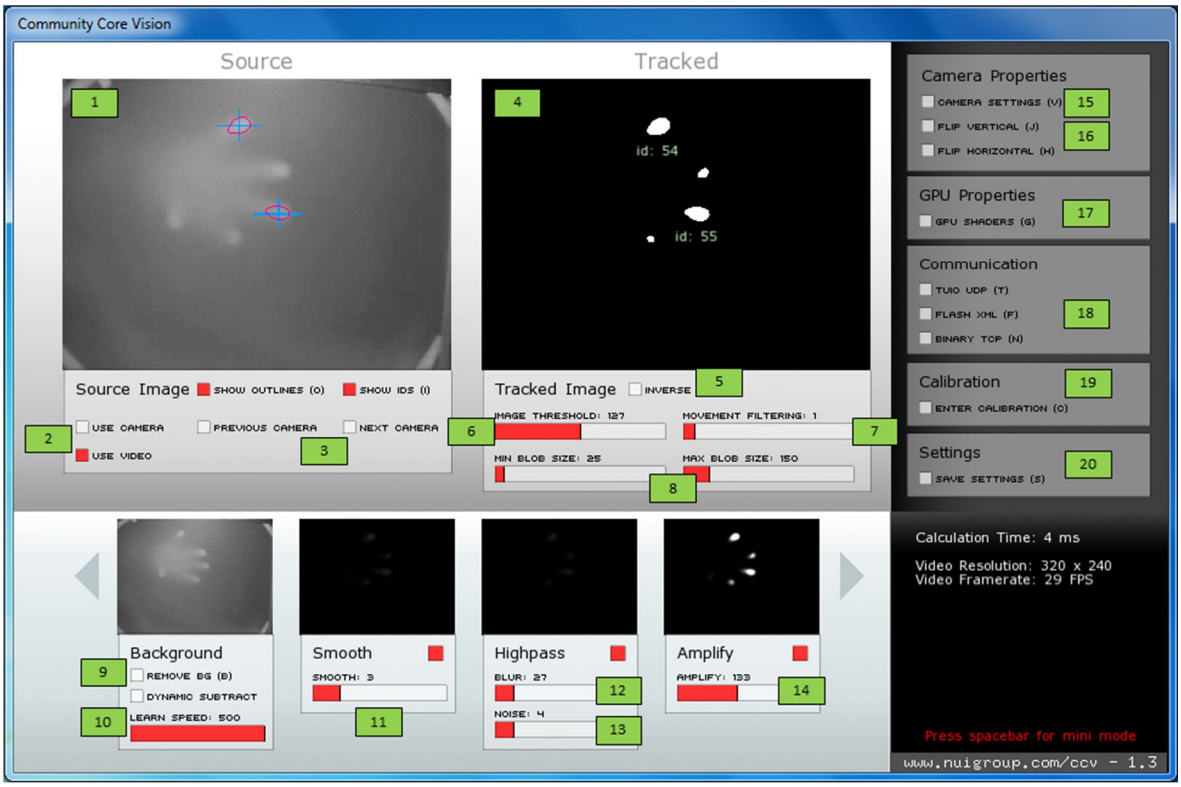
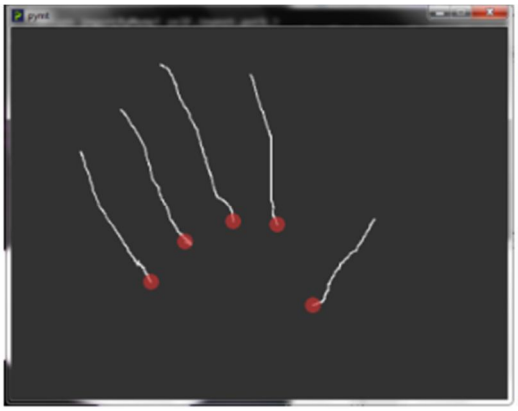


Figura 49 – *Screenshot* do CCV.

1. Imagem fonte – Mostra a imagem obtida pela câmara ou ficheiro de vídeo. **2. Selecionar *input* –** Seleciona como *input* a câmara ou um vídeo. **3. Seleção de câmaras –** Se houver mais do que uma câmara ligada, é possível selecionar cada uma delas. **4. Imagem detetada –** Mostra a imagem final depois de filtrada. **5. Inverso –** Deteta *blobs* pretos em vez de *blobs* brancos. **6. *Slider* do *threshold* –** Ajusta o nível de aceitação dos pixéis detetados. Quanto maior o valor, mais os *blobs* têm de ser convertidos em *blobs* detetados. **7. Filtragem de movimento –** Ajusta o nível de distância (em pixéis) antes de um *blob* ser detetado. Quanto maior o valor, mais será necessário deslocar os dedos para o CCV detetar o movimento. **8. Tamanho mínimo e máximo do *blob* –** Ajusta o tamanho mínimo e máximo para aceitação do *blob*. Quanto maior o valor, maior tem de ser o *blob* para ser aceite. Se for muito baixo, mais facilmente perderá o seu ID. **9. Botão para remoção do fundo –** Captura uma imagem para usar como fundo estático, a ser subtraído à imagem ativa. **10. Subtração dinâmica –** Ajusta dinamicamente a imagem de fundo. Deve-se ligar quando a luz ambiente varia constantemente ou quando surgem falsos *blobs* resultantes dessa variação. O *slider* determina a velocidade de "aprendizagem" do fundo. **11. *Slider* de *smooth* –** Alisa a imagem e filtra o ruído. **12. *Slider* de *highpass blur* –** Remove o efeito *blurry*. **13. *Highpass noise* –** Filtra o ruído. **14. *Slider* de amplificação –** Clarifica pontos fracos, fortificando os *blobs*. **15. Opções de câmara –** Abre as opções da câmara. Se estiver a ser utilizada uma camara PS3 Eye, ficarão disponíveis configurações adicionais. **16. Flip Vertical e Horizontal –** Inverte horizontal e verticalmente a imagem. **17. Modo GPU –** Ativa a aceleração por *hardware* utilizando o GPU. Aconselhável apenas para placas gráficas recentes. **18. TUIO UDP, Flash XML e Binary TCP –** Ativa o envio de mensagens TUIO, Flash XML ou RAW (coordenadas xy). **19. Calibração –** Carrega o ecrã de calibração. **20. Opções para guardar –** Guarda as definições atuais num ficheiro XML.

D.3 – Utilização

Depois de todo o processo de preparação, interessa começar a experimentar, e uma boa forma de fazê-lo é extrair o arquivo PyMT 0.5.1 (http://pymt.eu/wiki/DevGuide/Installation). Este arquivo contém vários exemplos de aplicações (código aberto) que podemos testar, ou então criar as nossas próprias aplicações. Na Fig.50 podemos ver uma pequena aplicação, em Python, que desenha as *tracks* dos toques.



```
from pymt import *

class Tracer(MTWidget):

    def on_touch_down(self, touch):
        touch.userdata['trace'] = list(touch.pos)

    def on_touch_move(self, touch):
        touch.userdata['trace'].extend(touch.pos)

    def draw(self):
        set_color(1)
        for touch in getCurrentTouches():
            drawLine(touch.userdata['trace'])

runTouchApp(Tracer())
```

Figura 50 – Programação de aplicações multitoque com o PyMT.

Com a linguagem de programação Python, e uma biblioteca PyMT repleta de funcionalidades, é fácil para um programador desenvolver aplicações. Neste exemplo temos uma aplicação capaz de desenhar linhas coincidentes com os movimentos dos nossos dedos, e o código correspondente no lado direito da imagem.

Outra forma de entrar do mundo da programação MT é com o Visual Studio. Se o objetivo for utilizar o protocolo TUIO, é necessário instalar a *driver* multitoque do Windows 7 (http://multitouchvista.codeplex.com/) para tirar partido das funcionalidades MT, e o Windows 7 Multitouch API (http://www.dotnetspark.com/kb/1666-windows7-multitouch-application-development.aspx) para começar a programar com o Visual Studio.

Anexo E

Exemplos de Utilização da *Framework*

Neste anexo estão presentes alguns exemplos de código, com o objetivo de demonstrar algumas capacidades da *framework*.

E.1 – Instanciação de uma *widget*

Este exemplo demonstra como criar uma *widget* e povoá-la com dados provenientes de um *site*.

```
# Instanciacao de uma factory de dados
factoryInstance = BaseFactory(DataIdentification)
# Instanciacao de uma widget e inicializacao da posicao e rotacao
widgetInstance = BaseWidget(pos=(x,y), rot=(theta))
# Povoamento da widget com dados da factory
widgetInstance.addDataPack(factoryInstance.getPack())
# Adiciona a widget ao seu pai (desktop ou widget)
self.add_widget(widgetInstance)
```

E.2 – Prototipagem de um gesto

Este exemplo demonstra a criação de protótipos para gestos simples.

```
# Executar o ficheiro 'gestureCreator.py' e efetuar o gesto pretendido
# Gesto efetuado fica registado no ficheiro 'gestfile.txt'
gesto_novo = [(365.0, 472.0), (372.0, 475.9), (386.0, 484.0), (396.0, 489.0), (404.0,
489.9), (411.0, 489.9), (419.0, 489.9), (428.0, 487.9), (436.0, 481.0), (442.0, 475.0),
```

```
(454.0, 461.0), (460.0, 448.0), (468.0, 427.0), (469.0, 397.9), (463.0, 384.0), (445.0,
380.0), (389.0, 382.0), (376.0, 384.0), (371.0, 385.9), (361.0, 395.0), (358.0, 400.0),
(355.0, 415.0), (355.0, 427.0), (358.0, 442.9), (362.0, 456.0), (364.0, 465.0), (365.0,
468.9), (366.0, 470.0), (367.0, 472.0), (368.0, 473.0), (368.0, 473.0)]
# Esta lista pode ser copiada para uma aplicacao para utilizar o gesto
# A utilizacao e explicada em D.3
```

E.3 – Utilização de um gesto prototipado

Este exemplo demonstra a utilização do prototipado de um gesto numa aplicação.

```
# Inicializacao da janela onde serao efetuados os gestos
window = MTWindow()
# Inicializacao da template que representa um gesto
gesto_novo = [(365.0, 472.0), ..., (368.0, 473.0)]

# Adiciona o padrao do gesto para ser reconhecido pelo Recognizer
Recon = Recognizer()
recon.addTemplate('NOME_DO_GESTO', gesto_novo)

# Instancia widget onde se vai efetuar gestos
gestWid = MTGestureWidget()
# Associa evento quando existe gesto
@gestWid.event
def on_gesture(gest, touch):
      # Reconhece o gesto realizado
      (name,score,theta) = recon.recognize(gest.strokes[0].screenpoints)
      # Se o gesto coincide com uma template, inicia acao
      if name == 'NOME_DO_GESTO':
            print 'Gesto encontrado!'
```

E.4 – *Parsing* de dados de um site

Este exemplo demonstra a utilização do *parser Beautiful Soup* para extrair informação de uma página.

```
# Filtro para diminuir tempo de pesquisa na arvore html (+desempenho)
cut = SoupStrainer(atributo='VALOR_ATRIBUTO')
# Cria uma arvore atraves do WEBLINK e restricao fornecidas
soup = BeautifulSoup(urllib2.urlopen('WEBLINK'), parseOnlyThese=cut)
# Procura na arvore todas as tags 'td'
allTD = soup.findAll('td')
# Procura na árvore a primeira tag 'title'
titulo = soup.find('title')
# Procura todas as tags cujo valor href se inicie com 'alunos'
allStudents = soup.findAll('a', href=re.compile('^alunos'))
```

E.5 – *Widget* de autenticação

Este exemplo demonstra a utilização da *widget* de autenticação para se poder extrair dados que só estão disponíveis após autenticação. Neste processo é utilizado o *Mechanize* para estabelecer uma sessão autenticada.

```
br = mechanize.Browser()
cj = mechanize.LWPCookieJar()
br.set_cookiejar(cj)
# Se ja existirem cookies, carrega-as
cj.revert('cookies.txt', True, True)
# Cria uma arvore para o parsing dos dados

if (‘NAO_AUTENTICADO’):
        br.select_form(nr=0)
        br.form['p_user'] = username
        br.form['p_pass'] = password
        # Submete o formulario atual
        response = br.submit()
        # Guarda os dados da sessao atual em ‘cookies.txt’
        cj.save('cookies.txt', True, True)
        # Le a pagina em sessao autenticada
        page = response.read()
        # Cria uma arvore da pagina em sessao autenticada
        soup = BeautifulSoup(page)
        print ‘SESSAO AUTENTICADA’
else:
        # Utiliza informacao armazenada
        print ‘SESSAO AUTENTICADA’
```

Anexo F

API

Neste anexo está especificada a API da *framework*, que deve ser consultada sempre que se utilize esta plataforma para desenvolvimento.

Widget de estudante

Widget para representação alunos.

Classe: `StudentWidget(**kwargs)`
Base: `MTWidget`

Parâmetros:

```
pPos: default (0,0)
scatWidget.pos: default (50, 50)
scatWidget.size: default (400, 300)
scatWidget.rotation: default 0
closeBtn.size: default (10% do scatWidget.size.x, 8% do scatWidget.size.y)
closeBtn.pos: default (scatWidget.size.x - 1.15 * closeBtn.size.x, scatWidget.size.y - closeBtn.size.y)
```

Métodos:

`addLabel(labl, position, fontSize):`
Adiciona uma label à *widget*.

`addButton(widget, labl, position, dimension):`
Adiciona um botão à *widget*.

`addImage(imgID):`
Adiciona uma imagem à *widget*.

`addDataPack(pack):`

Recebe um pacote de dados para povoar a *widget* com essa informação.

`dataFromID(id):`

Extrai dados de uma *factory*, através de um ID de aluno.

`centerPosition():`

Retorna a posição do centro da *widget*.

`on_btnX_touch_down(touch)`

Evento de toque para fechar a *widget*.

Widget de horário

Widget para representação de um horário.

Classe: `ScheduleWidget(**kwargs)`
Base: `MTWidget`

Parâmetros:

```
pPos: default (0,0)
scatWidget.pos: default (50, 50)
scatWidget.size: default (500, 470)
scatWidget.rotation: default 0
closeBtn.size: default (10% do scatWidget.size.x, 8% do
scatWidget.size.y)
closeBtn.pos: default (scatWidget.size.x - 1.15 * closeBtn.size.x,
scatWidget.size.y - closeBtn.size.y)
```

Métodos:

`addLabel(labl, position, fontSize):`

Adiciona uma label à *widget*.

`addDataPack(horas, horario):`

Recebe um pacote de dados para povoar a *widget* com essa informação.

`dataFromID(id):`

Extrai dados de uma *factory*, através de um ID de horário.

`centerPosition():`

Retorna a posição do centro da *widget*.

`draw():`

Desenha uma ligação com a sua *widget* pai.

`on_btnX_touch_down(touch):`

Evento de toque para fechar a *widget*.

Widget de curso

Widget para representação da ficha de um curso.

Classe: `CourseWidget(**kwargs)`
Base: `MTWidget`

Parâmetros:

```
pPos: default (0,0)
scatWidget.pos: default (50, 50)
scatWidget.size: default (300, 400)
scatWidget.rotation: default 0
closeBtn.size: default (10% do scatWidget.size.x, 8% do
scatWidget.size.y)
closeBtn.pos: default (scatWidget.size.x - 1.15 * closeBtn.size.x,
scatWidget.size.y - closeBtn.size.y)
```

Métodos:

`addLabel(labl, position, fontSize):`

Adiciona uma label à *widget*.

`addDataPack(pack):`

Recebe um pacote de dados para povoar a *widget* com essa informação.

`dataFromSigla(sigla):`

Extrai dados de uma *factory*, através de uma sigla de curso.

`centerPosition():`

Retorna a posição do centro da *widget*.

`draw():`

Desenha uma ligação com a sua *widget* pai.

`on_btnX_touch_down(touch):`

Evento de toque para fechar a *widget*.

Widget de disciplina

Widget para representação da ficha de uma disciplina.

Classe: `DisciplineWidget(**kwargs)`
Base: `MTWidget`

Parâmetros:

```
pPos: default (0,0)
scatWidget.pos: default (50, 50)
scatWidget.size: default (350, 340)
scatWidget.rotation: default 0
closeBtn.size: default (10% do scatWidget.size.x, 8% do
scatWidget.size.y)
closeBtn.pos: default (scatWidget.size.x - 1.15 * closeBtn.size.x,
scatWidget.size.y - closeBtn.size.y)
```

Métodos:

`addLabel(labl, position, fontSize):`

Adiciona uma label à *widget*.

`addButton(widget, labl, position, dimension):`

Adiciona um botão à *widget*.

`addDataPack(pack):`

Recebe um pacote de dados para povoar a *widget* com essa informação.

`setProfs(pack):`

Extrai dados de uma *factory*, e cria botões com ligações.

`setClass(data):`

Extrai dados de uma *factory*, e cria botões com ligações.

`dataFromID(id):`

Extrai dados de uma *factory*, através de um ID de disciplina.

`centerPosition():`

Retorna a posição do centro da *widget*.

`draw():`

Desenha uma ligação com a sua *widget* pai.

`on_btnX_touch_down(touch):`

Evento de toque para fechar a *widget*.

Widget de classe _____

Widget para representação de uma turma.

Classe: `ClassWidget(**kwargs)`
Base: `MTWidget`

Parâmetros:

```
pPos: default (0,0)
scatWidget.pos: default (50, 50)
scatWidget.size: default (380, 500)
scatWidget.rotation: default 0
kinetList.size:   default   (95%   do   scatWidget.size.x,   80%   do
scatWidget.size.y)
kinetList.pos:
closeBtn.size:   default   (10%   do   scatWidget.size.x,   8%   do
scatWidget.size.y)
closeBtn.pos: default (scatWidget.size.x - 1.15 * closeBtn.size.x,
scatWidget.size.y - closeBtn.size.y)
```

Métodos:

`center():`

Retorna a posição do centro da lista nesta *widget*.

`addLabel(labl, position, fontSize):`

Adiciona uma label à *widget*.

`addButton(id, labl, dimension):`

Adiciona um botão à *widget*.

`addDataPack(pack):`

Recebe um pacote de dados para povoar a *widget* com essa informação.

`centerPosition():`

Retorna a posição do centro da *widget*.

`draw():`

Desenha uma ligação com a sua *widget* pai.

`on_btnX_touch_down(touch):`

Evento de toque para fechar a *widget*.

Widget de professor

Widget para representação de um professor.

Classe: `ProfWidget(**kwargs)`
Base: `MTWidget`

Parâmetros:

```
pPos: default (0,0)
scatWidget.pos: default (50, 50)
scatWidget.size: default (370, 300)
scatWidget.rotation: default 0
closeBtn.size: default (10% do scatWidget.size.x, 8% do
scatWidget.size.y)
closeBtn.pos: default (scatWidget.size.x - 1.15 * closeBtn.size.x,
scatWidget.size.y - closeBtn.size.y)
```

Métodos:

`addLabel(labl, position, fontSize):`

Adiciona uma label à *widget*.

`addButton(widget, labl, position, dimension):`

Adiciona um botão à *widget*.

`addDataPack(pack):`

Recebe um pacote de dados para povoar a *widget* com essa informação.

`dataFromID(id):`

Extrai dados de uma *factory*, através de um ID de professor.

`centerPosition():`

Retorna a posição do centro da *widget*.

`draw():`

Desenha uma ligação com a sua *widget* pai.

`on_btnX_touch_down(touch):`

Evento de toque para fechar a *widget*.

Widget de percurso académico

Widget para representação de um percurso académico.

Classe: `GradesWidget(**kwargs)`
Base: `MTWidget`

Parâmetros:

```
pPos: default (0, 0)
scatWidget.pos: default (50, 50)
scatWidget.size: default (320, 600)
scatWidget.rotation: default 0
closeBtn.size: default (10% do scatWidget.size.x, 8% do
scatWidget.size.y)
closeBtn.pos: default (scatWidget.size.x - 1.15 * closeBtn.size.x,
scatWidget.size.y - closeBtn.size.y)
```

Métodos:

`center():`

Retorna a posição do centro da lista nesta *widget*.

`addLabel(labl, position, fontSize):`

Adiciona uma label à *widget*.

`addButton(url, labl, position, dimension):`

Adiciona um botão à *widget*.

`addDataPack(pack):`

Recebe um pacote de dados para povoar a *widget* com essa informação.

`dataFromID(id):`

Extrai dados de uma *factory*, através de um ID de percurso académico.

`centerPosition():`

Retorna a posição do centro da *widget*.

`draw():`

Desenha uma ligação com a sua *widget* pai.

`on_btnX_touch_down(touch):`

Evento de toque para fechar a *widget*.