FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Framework para a Integração de Dispositivos Móveis na Interação Pessoa-Computador

Wilson da Silva Oliveira

Mestrado Integrado em Engenharia Informática e Computação

Orientador: António Augusto de Sousa

Co-orientador: Diego Jesus

24 de julho de 2016

Framework para a Integração de Dispositivos Móveis na Interação Pessoa-Computador

Wilson da Silva Oliveira

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: António Coelho

Arguente: Luis Magalhães

Vogal: António Augusto de Sousa

24 de julho de 2016

Resumo

A utilização do rato e teclado para interagir com computadores é bastante comum atualmente. No entanto, estes periféricos limitam bastante o utilizador no modo como interagem com o computador, especialmente em tarefas mais complexas. Assim a procura por métodos mais naturais de interação persiste, existindo vários trabalhos e soluções que abordam desde as mesas interativas ao uso e reutilização de comandos de consola. Possíveis respostas para essa limitação poderão vir a criar novas formas de interagir com informação digital, seja em casos de uso específicos ou modificando o modo como as pessoas utilizam os seus computadores diariamente. É frequente encontrar utilizadores que possuam pelo menos um dispositivo móvel devido à proliferação destes nos últimos anos. Atualmente estes dispositivos são mais do que meros aparelhos para comunicação, estando eles próprios repletos de métodos de interação baseados em vários sensores como superfícies multi-toque, acelerómetro e outros.

Numa tentativa de propor um método de interação diferente, procura-se explorar a capacidade de integrar estes dispositivos como ferramentas de interação, substituindo ou complementando o rato e teclado de hoje. Fazendo uso da capacidade de processamento destes aparelhos e dos seus diferentes meios de interação, será possível interagir de formas diferentes com o computador, utilizando as superfícies multi-toque e os dados fornecidos pelos diferentes sensores (p.e. giroscópio). Para além dos dispositivos móveis poderem ser explorados para introduzir informação em computadores, a presença de meios de *output* - como o ecrã ou altifalante - permite a utilização dos mesmos como formas adicionais de exposição de informação. Para atingir este objetivo foi desenvolvida uma *framework* que procura simplificar a integração destes dispositivos na interação pessoa-computador. O desenvolvimento desta *framework* implicou a definição das arquiteturas a utilizar, tanto por uma aplicação móvel, como por um servidor. Este servidor é um componente de uma aplicação de computador e permite efetuar a comunicação com a aplicação móvel utilizando um protocolo definido pela *framework*. A aplicação móvel desenvolvida é capaz de interagir com qualquer servidor que implemente a *framework*. O servidor fica responsável por definir a interface da aplicação móvel, assim como implementar a lógica necessária para interagir com a aplicação do computador a partir dos dados transmitidos pela aplicação móvel. No contexto desta dissertação, o servidor faz parte de uma extensão para a aplicação *Blender*. Esta extensão permite utilizar dados recebidos da aplicação móvel (ecrã e acelerómetro) para interagir diretamente com o *Blender*.

Para avaliar a utilidade da solução encontrada foi realizado um caso de estudo, utilizando a aplicação referida no parágrafo anterior. Este caso de estudo permitiu analisar a reação e eficiência dos utilizadores que experimentaram a solução proposta, assim como avaliar o potencial da *framework* em aplicações futuras. Nas condições em que esta avaliação foi realizada, foi possível concluir que a metodologia proposta com a interação por toque apresenta alguma afinidade por parte dos utilizadores, mas que o uso do rato e teclado continua a ser mais eficiente. No entanto, foi possível identificar alguns melhoramentos facilmente integráveis na *framework*, que podem melhorar o seu desempenho e aceitação por parte dos utilizadores, nomeadamente no que respeita à interação por toque.

Abstract

Computer interaction nowadays is commonly done resorting to a keyboard and mouse. Although these methods are common, they present limitations in certain, more complex tasks. These difficulties motivate the search for new interaction methods, with various projects and solutions ranging from interaction tables to repurposing home console controllers. New methods that possibly solve these problems have the possibility to completely change the way people interact with their computers.

With the introduction of mobile devices, specifically smartphones, in people's daily routines it's common to find users with one or more of these devices. Today these devices are more than simple communication devices, supporting multiple interaction methods resorting to many sensors like accelerometers or multi-touch screens.

In an effort to propose a different solution for this problem, we seek to look into the potential that these devices have to be used as interaction methods complementing the common methods used today. By exploring the processing power and different interaction methods of these devices it will be possible to interact with computers in different ways. Although the primary focus is to explore these devices as input methods when using computers, the presence of output methods like the screen and the speaker allows their use as additional ways of exposing information and giving feedback to the users.

In order to achieve this goal, a framework was developed which aims to facilitate the integration of mobile devices in computer interaction. The development of this framework led to the specification of an architecture to be implemented both by a mobile application and a server. This server is one component of a computer application and is responsible to communicate with the mobile device using the protocol specified by the framework. The developed mobile application can interact with any server that implements this protocol. The server is responsible to specify both the interface and logic necessary for the mobile application to interact with the computer application.

For the context of this dissertation the server belongs to an add-on developed for the modeling program Blender. The add-on developed allows the use of the mobile application to interact with Blender.

In order to evaluate the proposed solution a case study was performed using the application mentioned in the last paragraph. In the evaluated scenario it was possible to conclude that the users showed some preference for the touch method but the use of mouse and keyboard was the most effective. However, it was possible to identify some improvements that can be easily integrated in the framework and improve the performance and acceptance by the users.

Agradecimentos

Gostaria de agradecer ao meu orientador e professor António Augusto de Sousa, assim como ao meu co-orientador Diego Jesus, por toda a ajuda e acompanhamento que disponibilizaram durante a realização da dissertação.

Wilson Oliveira

"In my experience there is no such thing as luck"

Master Jedi Obi-Wan Kenobi

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

OSC	Open Sound Control
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
SDCI	Smart Device Computer Integration
HTML	Hyper Text Markup Language
XML	eXtensible Markup Language
API	Application Programming Interface
JSON	JavaScript Object Notation

Capítulo 1

Introdução

A procura por diferentes e melhores métodos de interação com computadores e informação digital está presente desde há muito tempo [ZSS97]. Com o aumento da capacidade e complexidade dos sistemas, existe uma grande necessidade de explorar novas mecânicas que facilitem esta utilização [Zab11].

1.1 Contexto

A utilização do rato e teclado para interagir com computadores é bastante comum atualmente. No entanto, apesar desta utilização ser tão comum, nem sempre o teclado e o rato são os métodos mais indicados, como por exemplo na seleção de áreas complexas em imagens [KKY+15]. A procura por métodos mais naturais de interação e manipulação de informação persiste, existindo trabalhos que abordam a integração de mesas interativas [HTB14] e até a reutilização de comandos de consola [FPT12]. Possíveis respostas para esta limitação poderão criar novas formas de interagir com informação digital, seja em casos de uso específicos ou modificando o modo como as pessoas utilizam os seus computadores diariamente.

1.2 Motivação

É frequente encontrar utilizadores que possuam pelo menos um dispositivo móvel, devido à proliferação destes nos últimos anos. Atualmente estes dispositivos são mais do que um mero aparelho para comunicação estando eles próprios repletos de métodos de interação baseados em vários sensores como superfícies multi-toque, giroscópio e outros. Procurando explorar todo o potencial fornecido por estes dispositivos e continuar a procura por métodos naturais de interação surge o propósito desta dissertação.

1.3 Problema

Numa tentativa de explorar a capacidade de processamento destes dispositivos e dos seus diferentes meios de interação, surge a ideia de os utilizar como dispositivos de interação com computadores tradicionais.
Assim surgem as questões:

- Como interpretar a informação recolhida pelos sensores para controlar ações no computador?
- Como explorar a adaptabilidade da interface do dispositivo reconfigurando-a por via automática a partir do computador?
- Como transmitir a informação recolhida dos dispositivos para os computadores?
- Quais as vantagens ou desvantagens da integração destes dispositivos na interação com computadores?

1.4 Objetivos

Para tentar encontrar uma resposta para as perguntas na secção anterior, esta dissertação tem como objetivo principal o desenvolvimento de uma *framework*, denominada *Smart Device Computer Integration (SDCI)*, que permita a utilização de dispositivos móveis como meios de interação com computadores tradicionais.

Assim, com o estudo, desenvolvimento e avaliação dessa *framework*, pretende-se atingir os seguintes objetivos:

- Reconhecer e interpretar gestos e eventos de vários sensores
- Possibilitar, à aplicação no computador, a manipulação/configuração da interface presente no dispositivo
- Desenvolver um protocolo de comunicação entre dispositivo e computador
- Desenvolver uma aplicação protótipo
- Avaliar a usabilidade

Fazendo uso da capacidade de processamento dos dispositivos móveis e dos seus diferentes meios de interação, a *framework Smart Device Computer Integration* deve proporcionar métodos para interpretar os *inputs* dos diferentes sensores disponíveis e transmitir a informação necessária para que o computador efetue a ação pretendida. Para além disso, a *framework* deve possibilitar ao computador a manipulação da interface do dispositivo.

Para possibilitar a avaliação da usabilidade deste método de interação, a aplicação desenvolvida deve implementar todas as funcionalidades fornecidas pela *framework Smart Device Computer Integration*, para efeitos de teste e demonstração. Uma vez que a *SDCI* abrange tanto o dispositivo móvel como o computador, a aplicação vai interagir com uma extensão para o *Blender*, que siga as regras definidas pela *Smart Device Computer Integration*.

1.5 Estrutura da Dissertação

Para além da introdução presente neste capítulo, esta dissertação contém mais 5 capítulos. No capítulo 2, é descrito o estado da arte e são apresentados trabalhos relacionados assim como algumas bases teóricas. No capítulo 3, apresenta-se a solução proposta, a *framework SDCI*. Neste capítulo é apresentada a arquitetura proposta para a *framework Smart Device Computer Integration*, assim como o protocolo de comunicação especificado pela mesma, desde as mensagens definidas, até à interação entre os dispositivos. O capítulo 4 descreve a implementação da *framework* efetuada durante esta dissertação. Inclui as arquiteturas utilizadas, assim como as funcionalidades implementadas pela aplicação. No capítulo 5 é descrito o caso de estudo efetuado para avaliar a *SDCI* e os métodos de interação propostos. São expostos e analisados os resultados obtidos durante os testes.
Finalmente o capítulo 6 apresenta as conclusões retiradas desta dissertação, assim como indicações trabalho futuro e outras aplicações possiveis da *Smart Device Computer Integration*.

Capítulo 2

Estado da Arte

Como descrito no capitulo 1, a pesquisa por novos meios de interação com computadores é um processo contínuo e com uma história considerável. Esta pesquisa recai sobre a área principal de Interação Pessoa-Computador, assim este capitulo tem como objetivo a introdução aos princípios desta área assim como a exposição de outros trabalhos desenvolvidos.

2.1 Métodos de Interação

Nesta secção vão ser referidos alguns métodos de interação com informação, desde dispositivos móveis, que no âmbito desta dissertação se referem a *smartphones* e *tablets*, a mesas multitoque.

De seguida, apresenta-se o projeto *WallShare* [VGT+10] que procura explorar as capacidades dos dispositivos móveis como métodos de interação. Este projeto implementa um sistema de partilha de recursos especialmente orientado a equipas.

O *WallShare* utiliza um computador e um projetor para criar um ecrã com um ambiente de trabalho partilhado entre todos os utilizadores. Para se ligarem ao sistema os utilizadores têm que se conectar através do seu dispositivo móvel utilizando uma aplicação. O sistema, após reconhecer o utilizador, coloca um novo cursor na área partilhada que pode ser controlado por gestos realizados pelo utilizador e capturados pelo dispositivo. Os utilizadores podem assim disponibilizar recursos do seu telemóvel para a área partilhada. Estes recursos podem ser visualizados por todos os utilizadores, e podem ser adquiridos para os seus dispositivos.

De acordo com os autores [VGT+10], o *WallShare* traz alguns benefícios, nomeadamente:

- Estende as capacidades dos dispositivos móveis
- Proporciona um método de interação natural
- Permite a partilha de documentos entre utilizadores

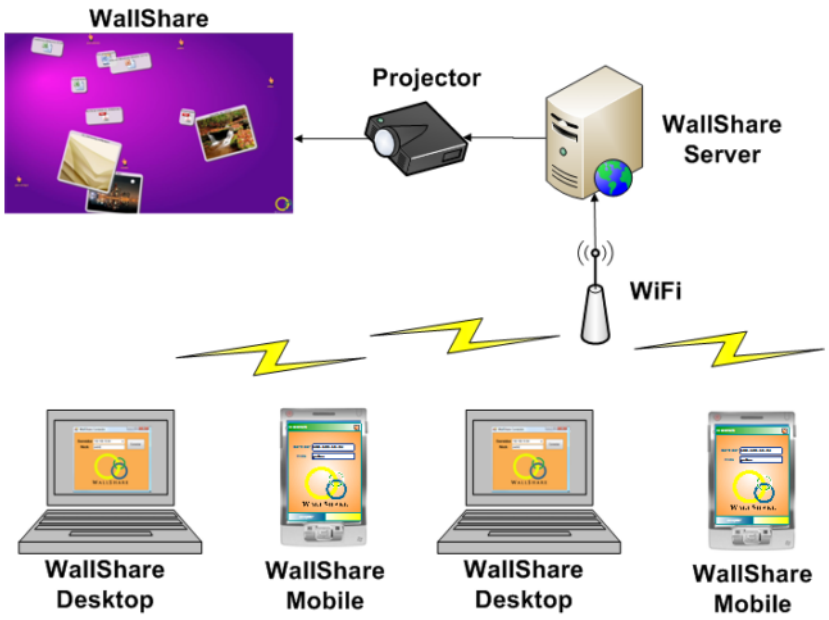


Figura 2.1: Sistema *WallShare* [VGT+10]

Apesar da sua implementação não parecer complexa nem exigir muitos custos, exige pelo menos um servidor e um ecrã externo. Também não proporciona, neste momento, muita funcionalidade para além da partilha de ficheiros.

Quase como uma resposta à introdução da tecnologia multi-toque e à popularização dos *smartphones*, começam a surgir monitores capazes de suportar a tecnologia. Atualmente existem vários computadores no mercado que possuem ecrãs com superfícies multi-toque. Apesar da popularização destes computadores hibridos a utilização de superficies verticais pode não ser a mais adequada, uma vez que esta utilização tem tendência para criar mais fadiga e desconforto [BGMF11]. De acordo com os autores, o rato parece ser mais eficaz em tarefas simples e que exigem algum nível de precisão [FWSB07]. Assim, uma possível configuração para computadores pessoais pode passar pela utilização de múltiplos métodos de interação [BGMF11] combinando superfícies multi-toque, rato e teclado proporcionando métodos de escrita [HHCC07] e interação familiares complementados com outros paradigmas de interação.

No âmbito das mesas de interação, também conhecidas por *tabletops*, começamos por analisar o *ActivitySpace* [HTB14].

O *ActivitySpace* é um projeto que tenta unir e simplificar a utilização de vários dispositivos através do uso de uma mesa multi-toque interativa.

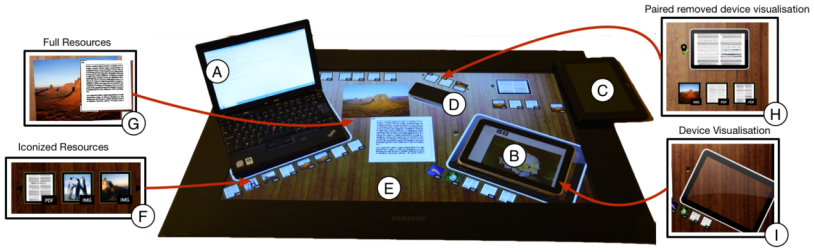


Figura 2.2: *ActivitySpace Desk* [HTB14]

Para possibilitar a sua implementação, o *ActivitySpace* faz uso de uma mesa de interação para ligar vários dispositivos, e cria uma área representativa no seu ecrã. Permite assim a partilha de recursos entre os dispositivos e a própria mesa assim como a sua visualização.

Magic Desk [BGMF11] assume uma abordagem diferente. Em vez de criar um sistema com regras de interação novas, explora a integração da tecnologia multi-toque na superfície das mesas de trabalho atuais.

Figura 2.3: *Magic Desk* a ser utilizada com um computador [BGMF11]

O *Magic Desk* propõe a utilização de uma superfície multi-toque sob o teclado e o rato. Esta superfície tem como principal objetivo aumentar as possibilidades de interação com o computador, desde a manipulação de janelas até à interação com menus de contexto, introduzindo ainda uma *taskbar* melhorada.

Na avaliação do *Magic Desk* os autores utilizaram várias métricas. Avaliaram o tempo necessário para completar uma tarefa, contabilizaram o número de vezes que o utilizador se reposiciona durante a tarefa e até a fadiga causada pela utilização. Com a sua análise destes dados foi possível concluir que os utilizadores utilizavam a superfície abaixo do teclado mais eficientemente do que

o resto da superfície, tanto para operações que exigiam uma ou mesmo duas mãos. Assim decidiram colocar a *taskbar* melhorada e as funções de redimensionamento de janelas na parte inferior da superfície. As funções que requerem interação mais simples foram desviadas para as zonas laterais ao teclado e ao rato.

Este sistema está limitado pela utilização de *hardware* próprio, apesar de ser teoricamente possível a sua adaptação a *tablets*. No entanto isto reduziria consideravelmente a superfície de toque disponível para a integração completa deste sistema.

Existe também alguma investigação sobre a utilização de *smartphones* como controlos de videojogos. *Baldauf* e outros implementaram, no dispositivo móvel, quatro *layouts* possíveis para um controlador de jogos, fazendo uso do ecrã multi-toque e do acelerómetro[BFAS15]. Os *layouts* implementados foram baseados em esquemas utilizados em jogos de *smartphones* e em comandos físicos existentes.

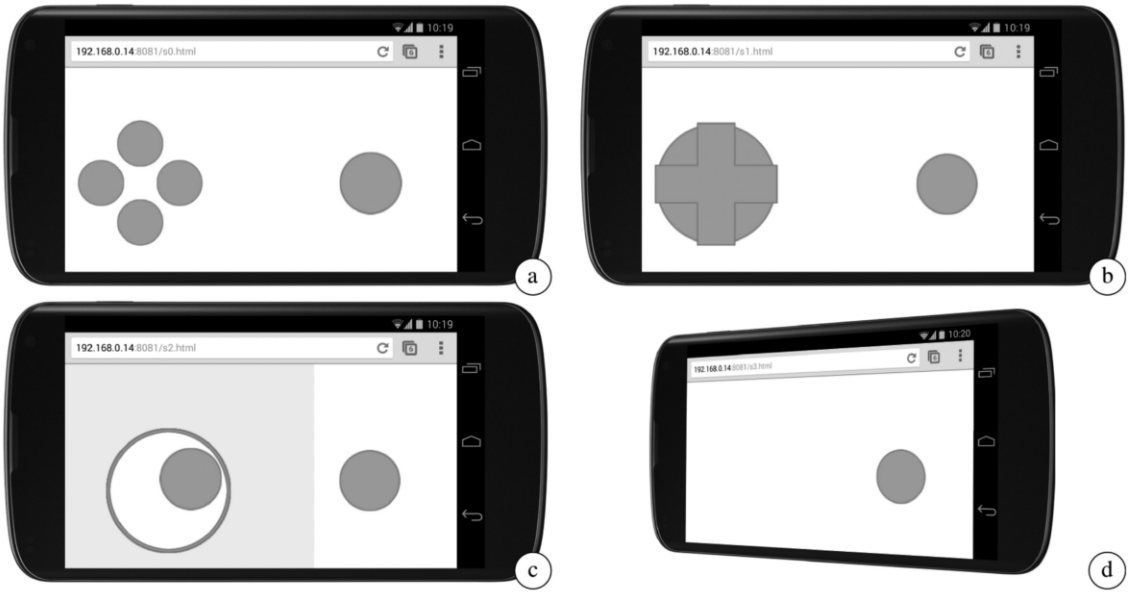


Figura 2.4: Esquemas de controlo implementados [BFAS15]

A figura 2.4 representa os quatro *layouts* implementados, por ordem: botões direcionais; *direction pad*; *joystick* flutuante; e finalmente controlo baseado na inclinação.

Para avaliar os diferentes métodos implementados, os autores realizaram um caso de estudo, onde propuseram a utilização destes métodos em duas tarefas diferentes. A primeira tarefa, designada pelos autores como uma tarefa de controlo, consistiu em posicionar um circulo branco sobre um circulo azul, que se deslocava ao longo do tempo. A segunda tarefa foi a utilização dos métodos no controlo de dois jogos bastante conhecidos, *Super Mario* e *Pac-Man*.
Para retirar conclusões deste caso de estudo, os autores recolheram vários dados, desde a opinião dos utilizadores a dados mais concretos, como tempo de execução das tarefas e o número de vezes

que os utilizadores desviaram o olhar do ecrã para o dispositivo.
Os autores tinham como objetivo analisar a possibilidade de utilizar os dispositivos móveis como comandos, em situações em que um comando físico não está disponível, como por exemplo, campanhas publicitárias em ecrãs públicos.
Assim, os autores concluíram que o maior problema dos métodos implementados é a falta de um *feedback* físico, o que leva a que os utilizadores desviem a sua atenção do ecrã principal para o dispositivo. Um método que se revelou promissor foi o *joystick* flutuante, apresentando um bom desempenho e reduzindo o número de vezes que os utilizadores desviaram o olhar do ecrã principal. O esquema de controlos baseado na inclinação do dispositivo apresentou os piores resultados, incluindo a rejeição dos utilizadores. No entanto, os autores ponderam que a sua utilização pode ser eficaz em tarefas diferentes das avaliadas, como por exemplo algo que exija movimento continuo.

Numa outra vertente de interação, alguns trabalhos de investigação exploram técnicas de captura de movimentos.
A consola *Nintendo Wii* destacou-se pela introdução de um comando capaz de realizar captura de movimentos, o chamado *Wiimote*. Apesar de não impressionar através do seu poder de processamento teve bastante sucesso na indústria, em parte pela introdução de novos meios de interação orientados para os jogos. Este comando permite aos utilizadores movimentarem-se para executar ações ao contrário de apenas pressionarem botões. Utiliza *Bluetooth* para comunicar com a consola e é capaz de detetar movimento ao longo de três eixos. Inclui ainda um sensor infra-vermelho para determinar a direção do comando.
Seguindo o exemplo e sucesso da *Nintendo*, a *Sony* lançou o seu próprio comando capaz de captar movimentos e a *Microsoft* deu um passo mais longe com o lançamento do *Kinect*. Este introduziu captura de movimentos através de câmaras de vídeo e sensores infra-vermelhos. Permite também a utilização de comandos de voz. Surgiu assim um novo paradigma de interação a ser pensado na criação de jogos, que poderia trazer muitas novidades ao setor.

Após a introdução de comandos para consolas capazes de realizar captura de movimentos, foi possível adaptar estes comandos para interação com computadores através da utilização de *SDKs* proprietários ou de terceiros [FPT12].

Exemplo desta adaptação foi o desenvolvimento de duas aplicações por Francese e outros [FPT12], utilizando o comando *Wiimote* e o sistema *Kinect*. Fazendo uso da captura de movimentos, os autores desenvolveram uma aplicação para a exploração da aplicação de mapas do Bing. No caso do comando *Wiimote*, os controlos utilizados para controlar a exploração do mapa são baseados nos três eixos de rotação capturados pelo comando, como pode ser visto na figura 2.5.

Na utilização do *Kinect* o controlo do movimento é feito de uma maneira similar aos de um avião em voo como pode ser visto na figura 2.6.

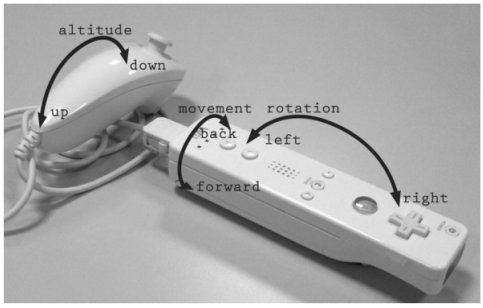


Figura 2.5: Esquema de controlos utilizado definido para o *Wiimote* [FPT12]

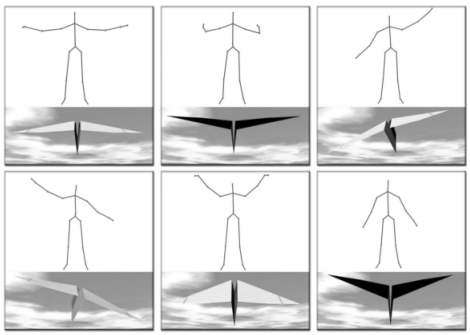


Figura 2.6: Movimentos capturados pelo *Kinect* para a exploração do mapa [FPT12]

A metáfora entre os gestos e os movimentos de um avião levou a uma fácil aprendizagem dos controlos e rapidamente os utilizadores executaram movimentos mais complexos [FPT12].

A avaliação subjetiva deste projeto foi feita através de dois questionários, um para analisar a satisfação dos utilizadores a facilidade com que concluíram as tarefas, e outro focado em quatro áreas especificas, avaliação geral, utilidade do sistema, qualidade da informação e da interface [FPT12].

Ambas as aplicações tiveram bons resultados a nível de usabilidade, mas houve uma clara preferência pelo sistema *Kinect* que obteve resultados ligeiramente superiores.
A novidade do sistema *Kinect* pode ter influenciado a preferência dos utilizadores [FPT12] mas foi possível concluir que a utilização deste sistema neste tipo de aplicações é preferível. Apesar disso, o mesmo pode não acontecer noutras aplicações que necessitem de gestos ou operações mais complexas. O uso prolongado do sistema também pode aumentar a fadiga dos utilizadores devido ao esforço físico necessário. A integração deste sistema no uso tradicional também se complica pela necessidade de adquirir o *hardware* necessário para a sua utilização.

Esta secção descreveu alguns projetos atuais que procuram implementar novos métodos de interação, fazendo uso de superfícies multi-toque e captura de movimentos. A utilização de dispositivos móveis como métodos de interação permite a utilização conjunta destas duas tecnologias, uma vez que os dispositivos atuais possuem ecrãs multi-toque e sensores capazes de detetar o

movimento do dispositivo.

2.2 Base Tecnológica

O crescimento e proliferação da utilização dos computadores tanto em ambientes profissionais como pessoais fizeram da área de Interação Pessoa-Computador uma área de constante investigação. A necessidade de fornecer aos utilizadores sistemas simples de usar, com curvas rápidas de aprendizagem, torna-se muito importante, de forma a permitir o aumento da produtividade e a redução da frustração dos utilizadores [BJJ98]. O desenho de uma interface é uma tarefa complexa que reutiliza informação de várias áreas cientificas, incluindo da psicologia. Tem como objetivo não só apresentar e recolher informação do utilizador de uma forma simples, mas também simular a manipulação de objetos [BJJ98]. A evolução desta área procura tornar as interfaces mais simples de aprender e fáceis de utilizar. Isto tem como consequencia o aumento da complexidade e dificuldade da sua implementação, uma vez que o número de opções e possibilidades disponibilizadas para desempenhar uma tarefa aumenta [BJJ98].

Surgem mais recentemente as interfaces naturais, que se distinguem das interfaces comuns pela utilização de meios de interação inerentes aos humanos, como toques, gestos e fala [SAL14].

Uma interface gráfica tradicional tem em mente a utilização de um rato e teclado e assenta no paradigma *WIMP - Windows, icons, menus, pointer*, que como o nome pode indicar, introduz o conceito de janelas e ícones, com os quais o utilizador pode interagir diretamente ou indiretamente através de menus, tipicamente utilizando o rato [Liu10]. Com a introdução das superfícies multi-toque, este tipo de interfaces continua a ser funcional, mas pode não ser o mais adequado para a introdução de gestos. Existem até algumas aplicações que já adaptam a sua interface, como a suite *Office* da *Microsoft* que, nas suas últimas versões, apresenta uma interface mais compatível com interação por toque[MN13].
A introdução recente de dispositivos capazes de captura de movimentos levou à pesquisa de interfaces de interação natural. Este tipo de interfaces usa como principal meio de interação gestos, movimentos e comandos de voz, permitindo uma interface gráfica simplificada mas com toda a funcionalidade [FPT12].
Em alguns casos, as interfaces naturais exigem uma fase algo prolongada de adaptação inicial. No entanto isto nem sempre sucede, pois, dependendo da tarefa a ser executada, a sua aprendizagem pode ser bastante simples como se mostra em [FPT12].

Foi identificado por Foley e outros [FWC84] um conjunto de tarefas simples que os utilizadores podem efetuar em aplicações gráficas [BBRS06]:

- Posicionar
- Orientar

- Selecionar
- Especificar um movimento (exemplo, desenhar uma linha)
- Quantificar
- Introduzir texto

Existem algumas aplicações que tentam usar dispositivos móveis para efetuar estas tarefas em computadores, como podemos ver em [JHKB06]. Mas como vimos em [FWSB07], o rato pode ser mais eficiente em tarefas simples como a seleção de objetos. Assim porque não manter a utilização do rato para estas tarefas de precisão, enquanto se utiliza a tecnologia multi-toque para tarefas mais complexas de efetuar com o rato, como posicionar objetos ou desenhar uma linha.

2.2.1 TUIO

TUIO [KBBC05] é um protocolo criado para ligar mesas de interação entre si, com o objetivo de manter a posição de objetos e gestos sincronizados entre diferentes mesas de interação ou ecrãs. O protocolo define dois tipos principais de mensagens: *Set* e *Alive*. Mensagens *Set* são utilizadas quando um novo objeto ou toque é detetado e contêm a informação da sua posição e orientação. Mensagens *Alive* são mensagens periódicas que contêm informação dos objetos detetados num determinado momento na superfície. Não é definido nenhum tipo de mensagens para a remoção de objetos, sendo o cliente responsável por detetar a sua remoção através das mensagens *Alive*. Para obter uma menor latência no uso do protocolo, este utiliza o protocolo de transporte UDP. No entanto, o protocolo UDP não previne contra a perda de informação na rede, pelo que para contrariar a possível perda de mensagens e garantir fiabilidade na informação o *TUIO* tem alguma redundância nas suas mensagens.
O TUIO é implementado utilizando o *Open Sound Control* e segue a sintaxe proposta pelo mesmo. O *Open Sound Control - OSC* define um protocolo de comunicação entre dispositivos baseado em mensagens que seguem uma estrutura definida pelo *OSC* e são independentes do tipo de transporte utilizado. Assim, a implementação do protocolo TUIO está dependente de uma biblioteca capaz de interpretar e manipular os pacotes definidos pelo *OSC*. O *TUIO* pode ser implementado em dispositivos com ecrãs multi-toque e transmitir para outros dispositivos a informação detetada pelo ecrã como coordenadas e orientação do objeto, assim como outros parâmetros calculados pelo *TUIO* como aceleração, rotação e o vetor de movimento.
Apesar do *TUIO* ser um protocolo bastante completo, apenas define o transporte dos dados básicos de um movimento, como posição e orientação. Isto deixa para o cliente a interpretação dos gestos. Seria interessante que o protocolo permitisse o envio de gestos já processados como a identificação de um *swipe* ou de um *pinch*.

2.2.2 Análise de tecnologias de desenvolvimento

Esta secção refere algumas das tecnologias que podem ser utilizadas para auxiliar o desenvolvimento da aplicação proposta.

A manipulação da interface do dispositivo por parte do servidor é uma funcionalidade importante que se pretende explorar durante o desenvolvimento desta dissertação. Para cumprir este objetivo, foram analisadas algumas *frameworks* de desenvolvimento de aplicações móveis que permitissem acesso a funções nativas disponibilizadas pelos sistemas do dispositivo móvel, mas que ao mesmo tempo disponibilizassem um maior controlo durante a execução sobre a interface apresentada no dispositivo. A possibilidade da aplicação poder ser compilada para várias plataformas ao mesmo tempo também é um fator de interesse, permitindo possuir rapidamente uma aplicação multi plataforma sem necessidade de efetuar duas implementações.

Por estes dois motivos a implementação da aplicação utilizando as ferramentas nativas dos sistemas operativos móveis foi excluída. Para além de ser necessário implementações diferentes para obter uma aplicação multi plataforma a manipulação da interface é também mais difícil uma vez que exige que os elementos de interface sejam desenhados antes da compilação da aplicação.

Uma das ferramentas analisadas foi o *Unity - Game Engine*. O *Unity* é uma ferramenta de desenvolvimento de jogos, que permite exportar as aplicações desenvolvidas para vinte e uma plataformas diferentes, incluindo *android* e *iOS* [Tec16].

A utilização do *unity* permite criar interfaces a partir de imagens e objetos criados na ferramenta de desenvolvimento. O desenvolvimento com *unity* suporta também o uso de toque e do acelerómetro como métodos de interação com a aplicação[Tec16]. Para possibilitar a manipulação da interface utilizando o *unity* podem ser utilizadas imagens para criar a interface, e seriam especificadas no protocolo as coordenadas sobre as quais os eventos eram registados.

Este processo traria complicações, principalmente na criação de interfaces. A utilização de imagens tornaria mais complicada a implementação de interfaces responsivas para dispositivos de diferentes tamanhos.

A utilização do *unity* permite a implementação do protocolo de comunicação com *TCP* e *UDP*.

Para além do *unity*, uma das *frameworks* de desenvolvimento testadas foi *Ionic*. A utilização de *Ionic* permite o desenvolvimento de aplicações *web* multi-plataforma. É construida sobre a *framework Angular* e permite o desenvolvimento de aplicações utilizando *HTML*, *Javascript* e *CSS*. As aplicações desenvolvidas com *Ionic* permitem a exportação para *web* e dispositivos móveis

com *android* e *iOS*[Co16].

A utilização de *ionic* obriga à implementação do protocolo de comunicação com *websockets*, uma vez que este é o único protocolo suportado por *javascript* na web.

Para permitir a manipulação da interface a utilização de *html* e *javascript* fornece algumas facilidades, uma vez que *javascript* permite a manipulação direta da interface. Assim, é possível, para modificar a interface presente no dispositivo, receber uma *string html*, converte-la para um elemento válido de *html* e adicionar diretamente este elemento.

A utilização de *html* e *javascript* também facilita a subscrição de eventos dos sensores, uma vez que é possível associar um evento a um elemento, independentemente das suas coordenadas no ecrã.

Existem mais algumas *frameworks*, como *Xamarin*[1] ou *libGDX*[2], que poderiam ser utilizadas durante a realização desta dissertação, mas devido a restrições de tempo não puderam ser analisadas.

Uma vez que a utilização de *html* e *javascript* parece fornecer algumas vantagens face à utilização de *unity*, como uma manipulação mais simples da interface, um melhor processo para implementar a subscrição de eventos e ainda ser uma tecnologia mais familiar, optou-se pela utilização de *Ionic* no processo de desenvolvimento desta dissertação.

2.2.3 Reconhecimento de gestos

Devido à decisão de utilizar *Ionic* para o desenvolvimento da aplicação móvel, nesta secção são apresentadas bibliotecas que auxiliem no reconhecimento de gestos sobre o ecrã multi-toque e que possuam implementações na linguagem *javascript*.

Existem dois tipos de gestos que interessam no desenvolvimento da *framework* proposta. Em primeiro lugar, temos o reconhecimento de gestos típicos de ecrãs multi-toque, como *swipe*, *pinch* ou *rotate*, que serão identificados daqui para a frente como eventos multi-toque. Em segundo lugar, temos o reconhecimento de gestos livres sobre uma área do ecrã. Esta funcionalidade permite a especificação de gestos compostos por um ou mais traços e, posteriormente, o seu reconhecimento quando efetuados sobre o ecrã para despoletarem ações.

[1] https://www.xamarin.com/
[2] https://libgdx.badlogicgames.com/

2.2.3.1 Reconhecimento de Eventos Multi-toque

Primeiro temos o *Touchy.js*, uma biblioteca de *javascript* que permite algum controlo sobre os eventos de toque sobre o ecrã. O *Touchy.js* fornece algumas funções úteis, como definir eventos com um número mínimo de toques sobre a superfície [Set16]. No entanto não suporta o reconhecimento de eventos como *swipe* ou *pinch* e portanto não pode ser utilizada no desenvolvimento da *framework*.

De seguida temos também o *Zepto.js*, outra biblioteca de *javascript* que tem como objetivo fornecer várias funcionalidades aos programadores, como funções para realizar pedidos ou manipular formulários de páginas *web*. Possui ainda um módulo capaz de reconhecer eventos de toque. Este módulo permite o reconhecimento de eventos como duplo toque ou *swipe*. No entanto não tem suporte para eventos de multi-toque como *pinch* ou *rotate* [Fuc16].

Finalmente, a biblioteca *Hammer.js* disponibiliza funções para subscrever vários eventos multi-toque, como *swipe* e *pinch* [AS16], e retorna vários parâmetros sobre os eventos, como por exemplo a quantidade de *zoom* aplicada num gesto *pinch*. Permite ainda uma serie de configurações, como tempo mínimo e distância mínima para o reconhecimento de um evento, assim como a definição de novos eventos a serem processados pela biblioteca. Ou seja, para além dos gestos suportados, seria possível definir novos tipos de eventos a serem reconhecidos. Para além disso é uma biblioteca *open-source* bastante simples, sem dependências extra.

Após a análise destas três bibliotecas apresentadas, decidiu-se utilizar o *Hammer.js* na implementação da *framework* por duas razões. Primeiro, suporta, sem configurações adicionais, todos os eventos de interesse para a *framework*. Segundo, permite uma grande quantidade de configurações sobre a biblioteca e os eventos que podem ser reconhecidos.

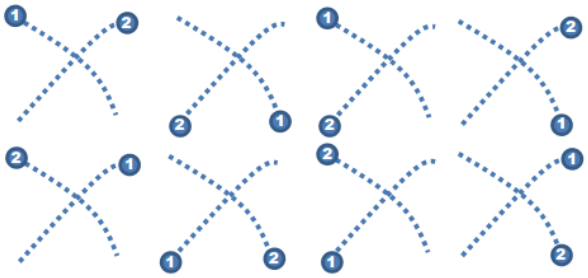
2.2.3.2 Reconhecimento de Gestos livres

O *$N Multistroke Recognizer* é uma biblioteca simples e eficiente, que utiliza geometria e trigonometria básica para efetuar o reconhecimento de desenhos sobre um ecrã multi-toque. O *$N* tem como objetivo permitir o desenho de interfaces baseadas em gestos. Esta biblioteca é a evolução do trabalho apresentado pela biblioteca *$1 Unistroke Recognizer*[AW10].

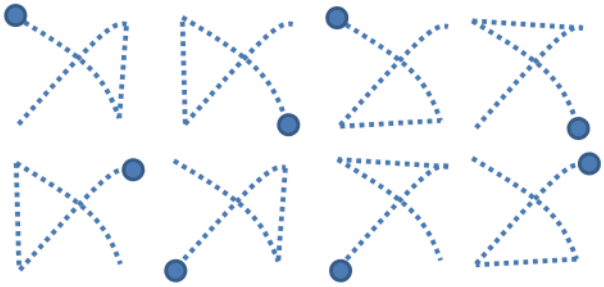
A biblioteca *$1* permite o reconhecimento de gestos com um único traço e efetuados na mesma direção que o gesto base especificado. Por outro lado, o *$N* permite a especificação de gestos com múltiplos traços. Assim, o *$N*, é capaz de reconhecer desenhos com múltiplos traços, independentemente da ordem e direção em que são desenhados pelo utilizador no ecrã.

Tanto o *$N*[3] como o *$1*[4] fornecem uma aplicação de teste na sua página *web*.

Um gesto definido com múltiplos traços pode ser desenhado pelo utilizador de várias maneiras diferentes, uma vez que a ordem de desenho deste gesto pode sofrer permutações ao nível da ordem dos traços, como na direção dos mesmos. Assim para permitir o reconhecimento de gestos com múltiplos traços, o *$N* cria e guarda todas as permutações possíveis do gesto, para poderem mais tarde ser comparadas com o desenho efetuado. Isto permite ao *$N* utilizar algum do trabalho efetuado para o *$1*, uma vez que na realidade compara gestos com um único traço, obtidos a partir da permutação do gesto definido[AW10].



(a) Diferentes permutações para um gesto "X"de dois traços



(b) Gestos com um traço criados pela biblioteca *$N*

Figura 2.7: Diferentes permutações para um gesto, calculadas pelo *$N*

Como é possivel observar na figura 2.7 [AW10], para um gesto "X" definido com dois traços são possíveis oito permutações possíveis do gesto. A figura 2.7b representa os oito gestos guardados pela biblioteca, que vão ser comparados com o gesto desenhado pelo utilizador, quando for efetuado o reconhecimento.

Devido à simplicidade da implementação do *$N*, existem algumas limitações. Devido à utilização de permutações com um único traço para efetuar o reconhecimento, o *$N* permite o reconhecimento de gestos efetuados com menos traços do que o original, mas pode falhar quando o gesto do utilizador utilizar mais traços do que a especificação original[AW10].

Também foi analisado o *GestRec*[uwd16], uma implementação *javascript* do algoritmo de reconhecimento de gestos *Protractor*[LV10] desenvolvido por *Yang Li*. Este algoritmo procura

[3]Demo: http://depts.washington.edu/aimgroup/proj/dollar/ndollar.html
[4]Demo: http://depts.washington.edu/aimgroup/proj/dollar/index.html

atingir os mesmos objetivos que o *$N*.

O *GestRec* disponibiliza uma aplicação para testes[5]. Após experimentar com as demos disponibilizadas por ambas as bibliotecas, optou-se por utilizar o *$N* na implementação da *framework*. Esta biblioteca parece apresentar melhor eficiência no reconhecimento de gestos, principalmente quando estes são desenhados utilizando diferentes permutações do original.

2.2.4 Dispositivos Móveis

Os dispositivos móveis são bastante comuns hoje em dia. Possuem ecrãs multi-toque que permitem não só *input* de gestos, mas também *output* de informação. Para além disto possuem vários sensores como acelerómetro, gps, camera e permitem a ligação a outros aparelhos através de antenas *Wifi* e *Bluetooth* [BTE11]. Todos eles integram um tipo de sistema operativo sendo um dos mais comuns o sistema *Android*. Os sistemas operativos atuais permitem várias manipulações destes sensores desde a sua leitura até à combinação das suas medições para a criação de novos sensores virtuais [Gui16].

2.3 Conclusões

A tabela 2.1 apresenta uma breve comparação dos diferentes métodos apresentados neste capítulo sobre as propriedades mais relevantes para o desenvolvimento desta dissertação.

	ActivitySpace	WallShare	Magic Desk	Wiimote and Kinect
Suporte Multi-toque	Sim	Sim	Sim	Não
Suporte Captura Movimento	Não	Não	Não	Sim
Suporte Output	Sim	Sim	Sim	Sim
Multiplos Dispositivos	Sim	Sim	Não	Não
Hardware Especifico	Sim	Sim	Sim	Sim
Integração com Smartphones	Sim	Sim	Não	Não
Colaborativo	Sim	Sim	Não	Sim

Tabela 2.1: Comparação dos diferentes métodos disponíveis atualmente

Observando a tabela é possível verificar que o multi-toque é bem suportado por estes métodos, uma vez que o único método sem suporte é a implementação do *Wiimote* e do *Kinect*. Pelo contrário, estes destacam-se por serem os únicos capazes de utilizar captura de movimento como método de interação.
O suporte a *output* de informação está relacionado com a capacidade do método de controlo transmitir *feedback* ao utilizador, seja por imagem ou por áudio. Neste caso, o *Wiimote* é capaz de fornecer algum *feedback* ao utilizador, através de um motor de vibração e de uma pequena coluna, para além de possuir quatro *leds* que podem ser controlados. Por outro lado o *Kinect* necessita da

[5]Demo: http://uwdata.github.io/gestrec/

utilização do ecrã para transmitir informação ao utilizador.
O suporte à utilização de múltiplos dispositivos é apenas suportado pelo *ActivitySpace* e pelo *WallShare*. Contudo, o *Magic Desk* propõe que a utilização de dispositivos móveis pode ser integrada no seu sistema.
Todos os métodos requerem *hardware* específico, como mesas de interação ou comandos, mas é necessário mencionar que os requisitos de *hardware* do *WallShare* são os mais simples, necessitando apenas de um servidor e de um ecrã externo. O *ActivitySpace* proporciona a integração de dispositivos móveis, mas apenas para efetuar partilha de documentos. Pelo contrário, o *WallShare* integra a sua utilização como método de interação e controlo.

Finalmente, todos estes métodos, exceto o *Magic Desk*, têm algum tipo de suporte para trabalho colaborativo.
O *TUIO* é excluído da tabela 2.1, uma vez que se enquadra numa área ligeiramente diferente dos restantes. O *TUIO* não é em si um método de interação, mas a sua implementação possibilita a ligação e interação entre vários ecrãs multi-toque e dispositivos, incluindo *smartphones*. Por este motivo, é talvez a tecnologia analisada até ao momento que mais se relaciona com os objetivos desta dissertação.

Como tecnologias possíveis de serem utilizadas durante o desenvolvimento da *framework* foram analisadas o *Unity* e o *Ionic*. O *Unity* permite o desenvolvimento de aplicações multi-plataforma, com suporte para multi-toque e o acelerómetro, especialmente orientado para jogos. O *Ionic* permite o desenvolvimento de aplicações multi-plataforma, utilizando tecnologias *web*. A utilização de *Ionic* foi a escolhida para a implementação da *framework*, por ser a tecnologia mais familiar e também por permitir algumas simplificações na manipulação da interface, assim como na subscrição de eventos.

Escolhida a tecnologia a ser utilizada no desenvolvimento da *framework*, foram procuradas bibliotecas que auxiliassem o tratamento de eventos multi-toque e o reconhecimento de gestos. Para o reconhecimento de eventos multi-toque, como *pinch* e *swipe*, foram analisadas várias bibliotecas. De entre estas bibliotecas foi escolhida o *Hammer.js*, que possui suporte para este tipo de eventos, como permite ainda a configuração de novos eventos.
Para possibilitar o reconhecimento de desenhos no ecrã foi analisado o *$N Multistroke Recognizer* e o *Gestrec*. De entre estes dois optou-se pela utilização do *$N* para a implementação da *framework*. O *$N* permite o reconhecimento de desenhos, previamente especificados, sobre o ecrã multi-toque. A vantagem do *$N* é a possibilidade de especificar e reconhecer gestos com múltiplos traços, incluindo as diferentes permutações que estes desenhos podem assumir quando efetuados pelo utilizador.

Com o objetivo de distinguir o trabalho desta dissertação, dos trabalhos apresentados neste capítulo, o foco da implementação da *framework* incide sobre a utilização dos vários sensores

disponíveis nos dispositivos móveis, assim como a possibilidade de controlar a interface do dispositivo.
A utilização conjunta do ecrã multi-toque e dos sensores, como o acelerómetro, permite a inclusão de novas funcionalidades em aplicações, assim como uma interação possivelmente mais imersiva.

Capítulo 3

Smart Device Computer Integration

Neste capítulo são apresentados os conceitos principais da solução proposta por esta dissertação para o problema exposto na secção 1.3. Este capítulo descreve a *framework Smart Device Computer Integration* (*SDCI*). São abordadas as componentes principais da arquitetura assim como o protocolo de comunicação entre o servidor e o dispositivo móvel.

3.1 Descrição

Os dispositivos móveis atuais fornecem aos utilizadores vários métodos de interação com o próprio dispositivo, através dos vários sensores que contêm. Estes dispositivos contém vários sensores, como acelerómetro/giroscópio ou sensor de luz ambiente, que estão constantemente a recolher uma vasta quantidade de informação. O acelerómetro, por exemplo, permite a recolha de informação como a orientação do dispositivo, aceleração linear do dispositivo e até a quantidade de rotação efetuada num determinado momento. Para além disso, estes dispositivos, tipicamente possuem um ecrã multi-toque. Estes ecrãs permitem a interação com o dispositivo, através da seleção direta de elementos da interface, ou através de operações mais complexas, como um movimento *pinch* de dois dedos para ampliar uma fotografia.

Uma aplicação típica de computador poderia explorar estes sensores e o ecrã multi-toque para possibilitar a interação com a mesma. Por exemplo, uma utilização comum do acelerómetro em jogos de dispositivos móveis é a utilização da rotação do dispositivo para deslocar um objeto. Assim, uma aplicação de modelação 3D, poderia utilizar o acelerómetro para mover ou rodar um objeto selecionado. Seguindo a mesma linha de pensamento, porque não a mesma aplicação utilizar um gesto de *pinch* para ampliar o objeto, ou ainda, um *browser web* do computador utilizar *swipes* para navegar por uma página, explorando assim o potencial do ecrã multi-toque.

Para além de explorar o dispositivo móvel como um meio de introdução de dados na aplicação, também é possível utilizar o mesmo como um meio de exposição de informação ao utilizador. O ecrã pode ser utilizado para apresentar informação contextual da aplicação do computador, ou até

uma interface mais apropriada à tarefa que estiver a ser efetuada no momento. Por exemplo, uma aplicação como o *Paint* poderia apresentar a tabela de cores no dispositivo, quando uma ferramenta com essa possibilidade está a ser utilizada.

Para tentar explorar este tipo de interações surge a ideia de desenvolver uma *framework*, designada por *Smart Device Computer Integration*, que permita facilmente a exploração dos meios de interação dos dispositivos móveis por parte de um computador. A *SDCI* é uma *framework* que tem como objetivo principal facilitar a integração dos dispositivos móveis como meio de interação com computadores.
Assim, a *SDCI* disponibiliza uma aplicação móvel capaz de interagir com qualquer aplicação de computador que implemente as regras especificadas pela *framework*. Uma aplicação de computador que utilize a aplicação móvel tem acesso a várias funcionalidades disponibilizadas pela *SDCI*:

- Utilização do acelerómetro/giroscópio como meio de *input*
- Deteção de vários eventos comuns em ecrãs multi-toque (*pinch*, *rotate* entre outros)
- Reconhecimento de gestos sobre o ecrã multi-toque
- Manipulação da interface do dispositivo

A utilização do acelerómetro como método de *input* permite a utilização dos eventos gerados por este sensor pela aplicação de computador. Os valores gerados por este sensor permitem implementar funcionalidades em aplicações do computador como aumento de valores ou deslocar objetos. A deteção de eventos como *pinch* permite a introdução de funções típicas de dispositivos móveis, como ampliar uma fotografia, em computadores que não possuam um ecrã tátil. O reconhecimento de gestos livres sobre o ecrã multi-toque permite ao computador definir gestos para serem posteriormente reconhecidos pela aplicação móvel. Esta funcionalidade permite ao computador introduzir novos métodos de interação, como utilizar gestos para atalhos na aplicação ou até possibilitar desenho livre sobre um *canvas*, sem a necessidade de dispositivos específicos como um *tablet* de desenho ou uma caneta. A manipulação da interface do dispositivo permite a um servidor externo fornecer e modificar a interface presente na aplicação do dispositivo. Isto permite à aplicação do computador controlar os elementos que são apresentados no dispositivo assim como o tipo de eventos que subscreve, como toques num botão ou gestos sobre o ecrã.

Estas funcionalidades foram implementadas durante a realização desta dissertação. Poderiam ser desenvolvidas outras funcionalidades, como o controlo do sensor de luz ambiente, ou até investigar a utilização do acelerómetro do dispositivo como um comando similar ao *Wiimote*, no entanto, optou-se por estas funcionalidades por permitirem criar uma base sólida para a *SDCI* assim como possibilitar a criação de uma aplicação de teste mais interativa e com mais funcionalidades.

Para possibilitar toda esta interação entre o dispositivo móvel e o computador é necessário manter um canal de comunicação constante entre ambos os dispositivos. Este canal de comunicação vai possibilitar o envio, para o computador, da informação recolhida pelos sensores do dispositivo, assim como o envio da interface especificada pelo computador para o dispositivo. Assim, a *SDCI* define uma arquitetura e um protocolo de comunicação a ser implementado por uma aplicação móvel e por uma aplicação de computador que pretenda explorar as funcionalidades apresentadas nesta secção.

Nas secções seguintes é explicado como se pode tirar partido destas funcionalidades, assim como são especificadas componentes essenciais como o protocolo de comunicação, o reconhecimento de gestos e a arquitetura proposta.

3.2 Arquitetura

A arquitetura especificada pela *SDCI* estende-se aos dois dispositivos necessários, o computador e o dispositivo móvel.

Na figura 3.1 é possível visualizar uma representação dos principais componentes da arquitetura especificada.

Ambas as partes estão dividas em vários componentes, alguns similares nas tarefas que executam.

O componente *Communication* tem como tarefa principal a receção e envio de mensagens entre os dois dispositivos. É um componente essencial para manter a transmissão de informação entre os dois dispositivos durante a utilização do dispositivo móvel como meio de interação.

O componente *Parser* é responsável pela validação das mensagens recebidas, incluindo a verificação de parâmetros obrigatórios, para permitir que os restantes componentes façam uso da informação transmitida com garantias da sua regularidade e sem necessidade de verificações adicionais.

No dispositivo móvel, a componente *EventHandler* é responsável por subscrever os eventos dos sensores suportados pela *framework*, ecrã multi-toque e acelerómetro. Os *handlers* registados para estes eventos devem processar os dados recebidos pelo sensor e criar as mensagens especificadas pelo protocolo para enviar para o servidor. Estas mensagens identificam no seu conteúdo o evento detetado e os elementos que despoletaram o mesmo.

Por outro lado, o *EventHandler* da aplicação do computador deve registar *handlers* para os diferentes elementos que subscrevem eventos no dispositivo. Estes *handlers* são executados a partir da informação contida na mensagem e devem invocar o *ApplicationHandler* assim como responder ao dispositivo móvel, seguindo as especificações do protocolo.

O componente *ApplicationHandler* deve implementar a lógica necessária para interagir com a aplicação do computador.

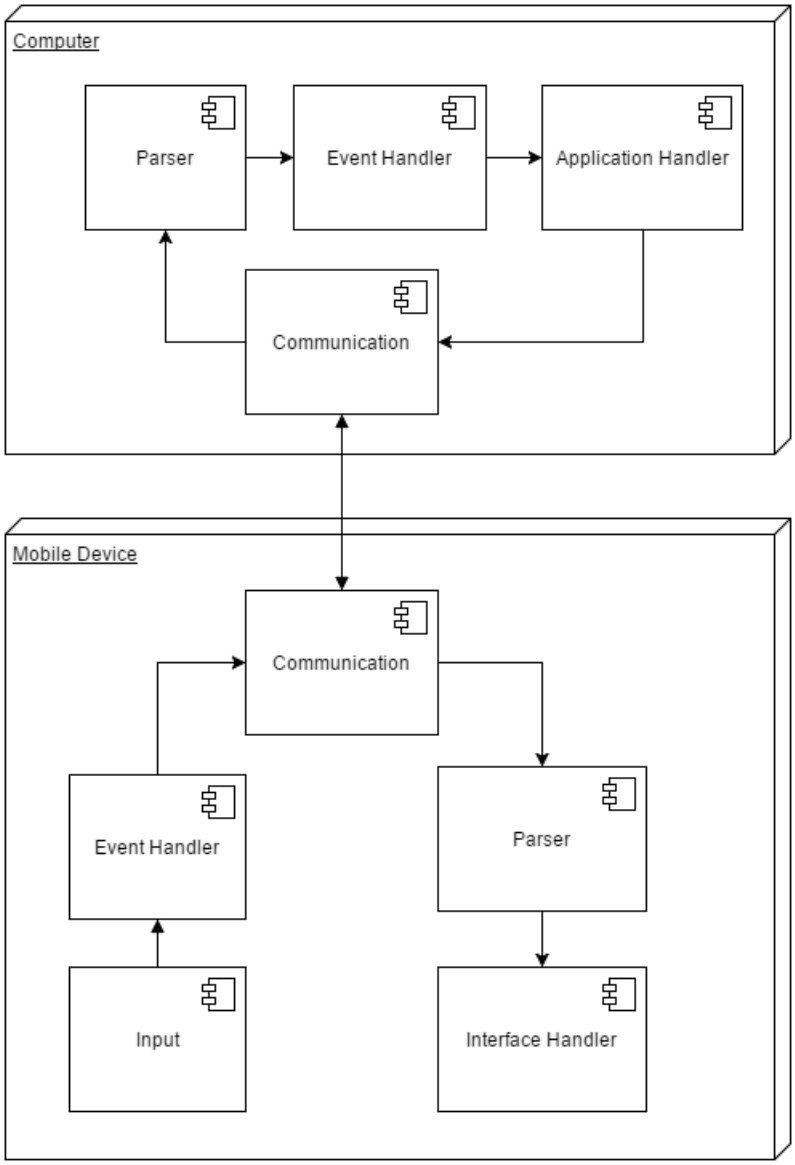


Figura 3.1: Representação da arquitetura implementada

3.3 Protocolo

Esta secção descreve o protocolo de comunicação a utilizar na implementação da *SDCI*. Apresenta as diferentes mensagens utilizadas pela *framework* assim como as palavras chave utilizadas nas mensagens, e a ordem pela qual as mesmas devem ser transmitidas entre o servidor e o dispositivo móvel.

As mensagens definidas pela *SDCI* utilizam uma estrutura baseada em *JSON*[1]. A utilização de *JSON* deve-se à simplicidade e acessibilidade que este tipo de estrutura permite tanto na leitura,

[1] *JSON* (*Javascript Object Notation*) é um formato simples de troca de dados [eI16]

como na manipulação da informação transmitida [eI16]. É também uma estrutura de dados bastante comum e implementada em várias linguagens de programação.

Analisando as funcionalidades propostas pela *framework SDCI* na secção 3.1 foram definidos vários tipos de mensagens que possibilitem a sua implementação. Assim, foram definidos três tipos principais de mensagens: mensagens de interface, mensagens de eventos e finalmente mensagens de informação.

As mensagens de interface são utilizadas pelo computador para permitir a manipulação da interface do dispositivo. Permitem a definição de interfaces novas, a adição de elementos a uma interface previamente definida e ainda a modificação de propriedades de elementos da interface atual. Para além disso, permitem ainda ao computador, subscrever eventos sobre os elementos da interface. Esta subscrição de eventos é o que permite à aplicação funcionar. Subscrever um evento sobre um elemento da interface faz com que o dispositivo móvel envie para o computador a informação retornada pelo evento para o computador. Este tipo de mensagens é descrito em mais detalhe na secção 3.3.1.

Para possibilitar o envio das informações retornadas pelos eventos dos sensores para o servidor, foram definidas as mensagens de eventos. Este tipo de mensagens identificam o tipo de evento detetado pelo dispositivo móvel, assim como outras informações relevantes que permitem ao computador efetuar ações sobre a aplicação.Este tipo de mensagens é descrito em mais detalhe na secção 3.3.2.

Finalmente foram definidas as mensagens de informação. Estas mensagens servem para transmitir informação entre os dispositivos, como informação de sucesso ou de erros durante a execução de operações. Este tipo de mensagens é descrito em mais detalhe na secção 3.3.3.

3.3.1 Mensagens de Interface

Existem três tarefas principais na manipulação de uma interface. Para manipular corretamente uma interface, deve ser possível definir uma interface nova, adicionar elementos a uma interface criada, e modificar elementos de uma interface definida. Assim, para suportar estas três ações, foram definidos três tipos de mensagens que constituem as mensagens de interface. Estas mensagens são do tipo apresentado na tabela 3.1.

Tabela 3.1: Tipos de mensagens de manipulação do ecrã

Tipo	Descrição
newScreen	Este valor representa uma mensagem que transmite uma interface nova para o dispositivo
addScreen	Este valor representa uma mensagem que transmite elementos a adicionar à interface atual do dispositivo
modifyScreen	Este valor representa uma mensagem que transmite um conjunto de regras para modificar a interface presente no dispositivo

As mensagens *newScreen* permitem a criação de interfaces no dispositivo. Estas mensagens fazem com que o dispositivo móvel apresente apenas a interface recebida na mensagem, eliminando primeiro qualquer interface que esteja a ser apresentada naquele momento. Para possibilitar o uso destas mensagens, estas devem conter a informação da interface a ser criada, assim como um nome que identifique a interface.

As mensagens *addScreen* permitem adicionar elementos a uma interface previamente definida, isto é, permitem a criação de interfaces no dispositivo, que têm como base interfaces criadas previamente. Assim, para possibilitar o uso destas mensagens, estas devem conter a informação dos elementos que vão ser adicionados, assim como um nome que identifique a nova interface. Uma vez que esta interface é construida tendo outra interface como base, este nome deve ser um subnome de uma interface criada. Isto é, se uma mensagem *addScreen* tem como objetivo adicionar elementos a uma interface identificada por "*MainInterface*", o identificador utilizado para a nova interface deve ser algo tipo "*MainInterface.AddedInterface*". Isto permite à aplicação identificar qual a interface base a que devem ser adicionados os novos elementos recebidos.

De seguida, temos as mensagens *modifyScreen* que permitem modificar atributos dos elementos da interface atual. Os elementos de uma interface possuem vários atributos que podem ser configurados após a sua criação, como cores ou texto apresentado. Na versão atual da *SDCI* é possível configurar os atributos presentes na tabela 3.2. Assim, é possível, utilizando a *framework*, esconder e desativar elementos, assim como modificar a cor de fundo e o texto apresentado. Para possibilitar o uso destas mensagens, estas mensagens devem conter uma lista com os identificadores dos elementos que vão ser modificados, assim como as propriedades que vão ser modificadas como apresentado na tabela 3.2.

Tabela 3.2: Propriedades implementadas nos elementos

Propriedade	Valor	Descrição
hidden	true/false	Propriedade utilizada para esconder um elemento
disabled	true/false	Propriedade utilizada para desativar um elemento
background	#555555	Propriedade utilizada para modificar a cor de um elemento
text	"texto"	Propriedade utilizada para modificar o texto contido num elemento

Para além destes tipos de mensagens de interface existe ainda um outro tipo, mensagem reduzida de interface. Estas mensagens reduzidas, permitem a transição entre interfaces previamente criadas no dispositivo, sem a necessidade de enviar novamente toda a interface. Por exemplo, após a criação de uma interface identificada por "*MainInterface*" com uma mensagem *newScreen*, o computador pode indicar à aplicação móvel para apresentar esta interface com uma mensagem reduzida. Esta mensagem reduzida contém apenas o identificador do ecrã que deve ser apresentado. Estas mensagens têm como objetivo principal reduzir a quantidade de informação transmitida entre os dispositivos, assim como melhorar a eficiência da aplicação.

Para ajudar a distinguir a funcionalidade da mensagem *modifyScreen* e *addScreen* podemos analisar a figura presente em 3.2.

Iniciando a interação a partir do primeiro ecrã apresentado na figura 3.2, após o utilizador clicar no botão *Scale*, o servidor envia uma mensagem *addScreen*. Esta mensagem mantém a interface do ecrã anterior e adiciona os controlos necessários para efetuar a operação de escalamento. Após o utilizador clicar no botão *Start using accelerometer*, o servidor responde com uma mensagem de *modifyScreen*, mantendo a interface com os mesmos componentes, mas modificando a cor e texto do botão clicado.

Para além da especificação de ecrãs, as mensagens de interface permitem também realizar a subscrição de eventos no dispositivo móvel. Para poder subscrever ou cancelar a subscrição de um evento, as mensagens de interface devem conter uma lista com os identificadores dos elementos que vão subscrever os eventos, assim como os eventos que cada elemento deve subscrever. Esta subscrição de eventos vai despoletar no dispositivo móvel o envio de mensagens de eventos, quando estes forem detetados.

3.3.2 Mensagens de Eventos

Quando um evento, do ecrã ou dos sensores, é detetado pelo dispositivo móvel, a informação obtida deve ser enviada para o computador. Assim, foram definidas as mensagens de eventos.

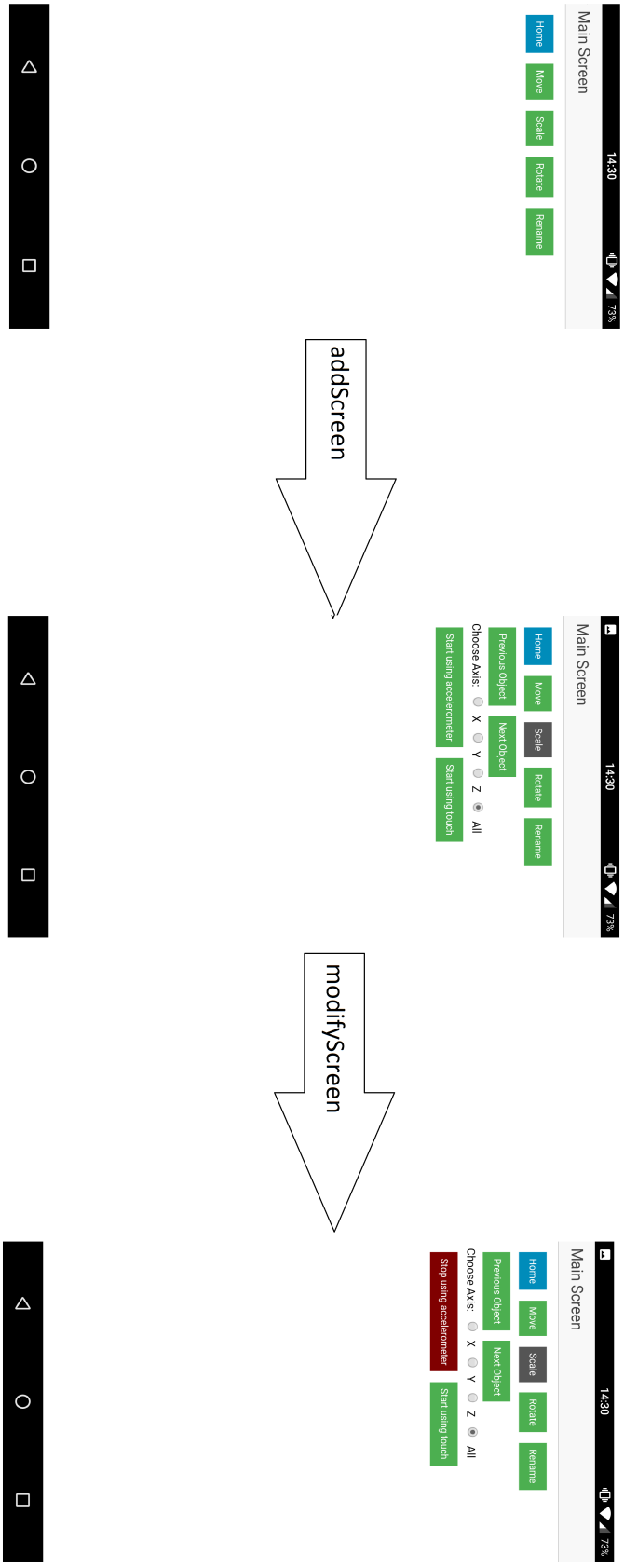


Figura 3.2: Diferenças entre a mensagem *addScreen* e *modifyScreen*

Estas mensagens contêm a informação relevante de um evento detetado e são transmitidas do dispositivo móvel para o computador.

Para possibilitar ao computador efetuar ações com os eventos detetados, este necessita de duas informações importantes. Primeiro, é necessário identificar o tipo de evento que ocorreu. O tratamento de uma mensagem pode ser diferente consoante o evento ocorrido. Por exemplo, a ação para um toque sobre um elemento é, provavelmente, diferente para a ação de um *swipe* sobre o mesmo elemento. Segundo, a mensagem deve identificar o elemento que despoletou o evento. Isto porque, a ação para um toque sobre um elemento, não é a mesma que a ação para um toque sobre outro elemento diferente. Assim todas as mensagens de eventos contêm a identificação do evento ocorrido e do elemento que o despoletou.

A versão atual da *framework SDCI* suporta vários tipos de eventos, do ecrã multi-toque e do acelerómetro. Os valores definidos para identificar estes eventos são apresentados na tabela 3.3. Os valores desta tabela, para além de utilizados nas mensagens de eventos, são utilizados pelas mensagens de interface para efetuar a subscrição dos eventos sobre um elemento.

Tabela 3.3: Valores assumidos pela chave *type* nas mensagens relacionadas com eventos do dispositivo

Valor	Descrição
tap	Utilizado para subscrever/representar um evento de toque num elemento
swipe	Utilizado para subscrever/representar um evento de arrastar sobre um elemento
pinch	Utilizado para subscrever/representar um evento multi-toque de "pinch" com dois dedos
rotate	Utilizado para subscrever/representar um evento multi-toque de rotação com dois dedos
free-draw	Utilizado para subscrever/representar um evento de toque livre no elemento
draw	Utilizado para subscrever/representar a funcionalidade de reconhecimento de gestos sobre um elemento
acceleration	Utilizado para subscrever/representar um evento do acelerómetro
stop-acceleration	Utilizado para remover a subscrição de eventos do acelerómetro
alert-text	Utilizado para mostrar, após um toque, um alerta com uma caixa de texto

Tabela 3.4: Chaves possiveis numa mensagem de evento

Chave	Exemplo	Tipo de eventos	Descrição
totalTouches	1	tap, swipe, pinch, rotate, free-draw, pan	Numero de toques detetado no ecrã
targetIDs	"idElemento"	Presente em todas as mensagens	Array com os ids que despoletaram os eventos
coordinate	[{x: 100, y: 150}]	tap, swipe, pinch, rotate, pan	Array com as coordenadas dos toques detetados
direction	2	swipe	Direção do movimento no caso de um evento *swipe*
scale	1.5	pinch, rotate, pan	Quantidade de ampliação efetuada
rotation	100	pinch, rotate, pan	Rotação efetuada num evento multi-toque em graus
deltaTime	113	swipe, pinch, rotate, acceleration, pan	Tempo em ms desde que o movimento foi iniciado
gestureName	"square"	draw	Nome do gesto reconhecido
gestureScore	0.88	draw	Grau de confiança no gesto reconhecido 0-1
acceleration	[0.3, 0.1, -0.5]	acceleration	Array com os valores de aceleração recolhidos do sensor
rotationVelocity	[3, 1, 6]	acceleration	Array com os valores da velocidade de rotação
rotationRate	[48, 16, 96]	acceleration	Array com a distancia da rotação efetuada
velocity	[4.8, 1.6, -8]	acceleration, pan	Array com a velocidade do movimento efetuado
distance	[76.8, 25.6, 128]	acceleration, pan	Array com a distancia do movimento efetuado
textValue	"Texto"	alert-text	Texto introduzido na caixa de texto do alerta
path	"[{x: 100, y: 150}...{x: 200, y: 280}]"	free-draw	Trajeto efetuado pelo utilizador sobre o ecrã

As mensagens de eventos, para além da identificação do evento e do elemento possuem outras informações. Estas informações variam consoante o evento representado pela mensagem. Por exemplo, uma mensagem de *pinch* possui informação sobre o número de toques detetados no ecrã assim como a quantidade de escala efetuada durante o movimento de *pinch*. No entanto,

uma mensagem de toque (representada por *tap* na mensagem) possui apenas a informação do número de toques detetados e das coordenadas no ecrã do toque, uma vez que num evento de *tap* não é possível calcular um valor para a escala efetuada. Os diferentes valores encontrados numa mensagem, para cada evento, podem ser vistos na tabela 3.4. Esta tabela identifica o nome utilizado para descrever a informação na mensagem, assim como exemplos do conteúdo e o tipo de eventos em que pode ser encontrada.

3.3.3 Mensagens de Informação

Para além das mensagens definidas nas secções anteriores, foram definidas ainda mensagens de informação. As mensagens de informação surgem da necessidade de transmitir entre os dispositivos informação sobre o sucesso ou insucesso das operações efetuadas, assim como confirmar a receção de mensagens.

Estas mensagens podem ser utilizadas como simples mensagens de confirmação de uma operação, ou então para apresentar um erro ao utilizador como vemos na figura 3.3.

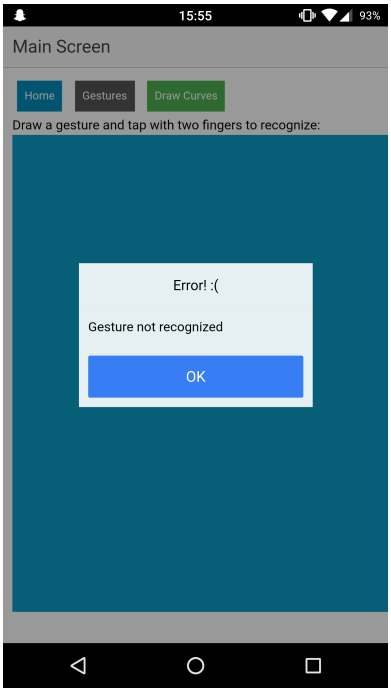


Figura 3.3: Alerta apresentado após uma mensagem de informação com um erro

Na figura 3.3, vemos a mensagem de erro "*Gesture not recognized*". Esta mensagem é transmitida através de uma mensagem de informação. O dispositivo quando deteta uma mensagem de informação que contém um erro, alerta o utilizador mostrando um texto contido na mensagem.

3.3.4 Sequência de Mensagens

Estabelecida a conexão entre o servidor e o dispositivo móvel, o envio de mensagens é sempre iniciado pelo dispositivo móvel.

A primeira mensagem a ser enviada é uma mensagem de informação especial, identificada por *FirstConfiguration*. Esta mensagem é utilizada pelo dispositivo para receber a sua primeira interface, disponibilizada pelo computador. Assim, a resposta é uma mensagem de interface, do tipo *newScreen*, enviada pelo computador. Esta resposta deve ser obrigatoriamente deste tipo, uma vez que define o primeiro ecrã base da aplicação.

Após esta primeira interação, a aplicação do dispositivo móvel está pronta a ser utilizada. A mensagem de interface recebida, para além de definir a interface do dispositivo, define os eventos que são detetados pelo dispositivo quando o utilizador interage com a aplicação. Assim, quando um evento é detetado, uma mensagem de evento é criada e enviada para o servidor.

Uma vez recebida e processada no servidor uma mensagem de evento, este responde ao dispositivo. Esta resposta pode assumir dois tipos de mensagem, uma mensagem de informação a confirmar a ação efetuada ou uma mensagem de interface, para modificar o ecrã apresentado no dispositivo.
A troca de mensagens entre os dois dispositivos deve terminar sempre com uma mensagem de informação, que garante que ambos os intervenientes receberam todas as mensagens e efetuaram as ações pedidas.

Esta é a interação principal entre as aplicações dos dois dispositivos. No entanto, a utilização de mensagens reduzidas de interface exige o tratamento de uma mensagem de informação especial.
Quando um ecrã, requisitado por uma mensagem reduzida de interface, não é encontrado no dispositivo, este envia ao servidor uma mensagem de informação a requisitar a interface completa do ecrã. O servidor deve estar preparado, para quando receber esta mensagem, reenviar para o dispositivo o ecrã completo requisitado.

3.4 Conclusão

Neste capitulo foram apresentadas os conceitos principais da *framework SDCI* desenvolvida ao longo da dissertação.

Iniciou-se por apresentar uma descrição das funcionalidades principais da *framework* seguida de uma visão das componentes principais da arquitetura da *SDCI*.

Foi também especificado o protocolo desenvolvido durante esta dissertação. O protocolo utiliza mensagens com uma estrutura de dados similar a um dicionário. Foram apresentados exemplos das várias mensagens definidas pelo protocolo, desde mensagens simples de *acknowledgement* a mensagens mais complexas como a manipulação da interface do dispositivo.

Para além disto foram exemplificadas as mensagens que devem ser enviadas pelo dispositivo, após eventos dos sensores ou ecrã multi-toque serem despoletados pelo utilizador. São também apresentados os vários parâmetros que devem ser incluídos nestas mensagens, desde a identificação do elemento subscrito ao evento até informação específica retornada pelo sensor. Estes parâmetros devem ser incluídos nestas mensagens para possibilitar ao servidor o seu tratamento e despoletar ações na aplicação que está a beneficiar deste método de interação.

Capítulo 4

Implementação

Neste capitulo é detalhadamente apresentada uma implementação da *framework SDCI* especificada no capítulo 3. Para tornar possível a realização de um caso de estudo com o objetivo de avaliar a usabilidade e viabilidade da *framework* sugerida, foi desenvolvida uma aplicação cliente-servidor, implementando a *framework*.

4.1 Tecnologia

Esta secção apresenta as tecnologias utilizadas na implementação da *SDCI* com o objetivo de implementar a estrutura especificada no capítulo 3.

4.1.1 Dispositivo móvel

A aplicação para o dispositivo móvel foi desenvolvida utilizando a *framework Ionic*, uma *framework* que permite o desenvolvimento de aplicações *web* multi-plataforma. É construida sobre a *framework Angular* e permite o desenvolvimento de aplicações utilizando *HTML*, *Javascript* e *CSS* [Co16]. As aplicações desenvolvidas sobre esta *framework* podem ser utilizadas num navegador *web* ou exportadas para os sistemas móveis nativos, *Android* e *iOS*.

Devido à utilização de *Javascript* na implementação da aplicação móvel, a tecnologia utilizada para implementar o protocolo de comunicação foi a *Websocket*. Esta tecnologia corresponde a um protocolo de comunicação por *sockets*, baseado em *TCP* e desenvolvido especificamente para a *web* [IF16]. A utilização de *websockets* deve-se a este ser o único protocolo de comunicação por *sockets* suportado pelos *browsers* ou por *webviews*, sendo estas últimas utilizadas pelo *Ionic* na exportação da aplicação móvel.

Para tornar possível o reconhecimento de gestos ao nível do dispositivo móvel, é utilizada a biblioteca *$N Multistroke Recognizer*. Esta biblioteca permite o reconhecimento de gestos com

múltiplos traços. Para além disso possui implementações em várias linguagens, incluindo *javascript* [LA16]. Para auxiliar na identificação e manipulação de eventos multi-toque, como *pan* ou *rotate*, foi utilizada a biblioteca de *javascript Hammer.js*, que fornece métodos para subscrever este tipo de eventos, sem ser necessário processar diretamente os dados dos sensores para cada evento [AS16].

4.1.2 Computador

Para possibilitar a utilização da *framework SDCI* é necessário uma aplicação de computador que tire partido das funcionalidades disponibilizadas pela *framework*. Esta aplicação deve implementar um servidor, seguindo a arquitetura especificada pela *SDCI*, e deve suportar o protocolo *Websocket* para funcionar em conjunto com a aplicação do dispositivo móvel. Assim, para evitar o desenvolvimento de raiz de uma aplicação de computador, procurou-se uma aplicação que permitisse a extensão das suas funcionalidades, assim como beneficiar das funcionalidades da *Smart Device Computer Integration*.

Por este motivo, foi decidido utilizar a aplicação *Blender*. O *Blender* é uma aplicação de modelação 3D *open source* que permite a extensão das suas funcionalidades através da criação de *add-ons* desenvolvidos em *Python* [Fou16]. Sendo o *Blender* uma aplicação de modelação 3D, fornece várias funções, como deslocar ou escalar objetos, que podem beneficiar das funcionalidades da *framework* de várias maneiras. Por exemplo, o acelerómetro pode ser utilizado para mover um objeto, ou ainda, utilizar um gesto multi-toque de *pinch* para escalar um objeto.

A extensão desenvolvida implementa a estrutura da *SDCI*, e interage diretamente com a *API* do *Blender*. Para ser possível a extensão suportar *websockets*, foi utilizada uma biblioteca externa de *Python* [Aug16].

4.2 Especificação das Mensagens

Esta secção apresenta a especificação utilizada para o protocolo apresentado na secção 3.3, assim como uma especificação para os gestos utilizados pela *framework* para o reconhecimento de gestos.

4.2.1 Protocolo

O protocolo proposto na secção 3.3 utiliza uma estrutura de dicionário chave-valor para as suas mensagens, assim optou-se pela utilização de *json* para a especificação do protocolo. Assim, todas as mensagens definidas pelo protocolo possuem uma chave *type*. Esta chave permite identificar rapidamente o tipo de mensagem transmitido assim como os parâmetros que são esperados nessa

mensagem. Esta chave pode assumir os valores apresentados nas tabelas 3.1 e 3.3 assim como o valor **info** e **FirstConfiguration**.

Na secção seguinte são especificadas as mensagens apresentadas no capítulo anterior (mensagens de interface, de eventos e de informação), assim como é explicada a utilização das mensagens para efetuar a subscrição de eventos.

4.2.1.1 Mensagens de interface

Para permitir à aplicação do computador manipular a interface do dispositivo, são utilizados os três tipos de mensagem presentes na tabela 3.1.

As mensagens do tipo *newScreen* e *addScreen* apresentam uma estrutura semelhante, exemplificada na listagem 4.1.

A chave *html* contém a interface a ser criada ou adicionada ao dispositivo. A interface transmitida consiste numa *string* de *html* válido, que permita a sua reconstrução na aplicação móvel.

```
{
   type: "newScreen",
   mode: "MainMode",
   html: "<div>...</div>",
   ids: ["rotateButton", "transformButton", "scaleButton"],
   events: {
               rotateButton: ["swipe", "pinch"]
            },
   property: {
            "transformButton": {
                "hidden": "true"
            },
            "scaleButton": {
                "disabled": "true",
                "background": "#555555"
            }
       }
}
```

Listagem 4.1: Exemplo de mensagem de novo ecrã

A chave *ids* contém um *array*, cujos objetos identificam elementos de interesse para efetuar a manipulação da interface. Os valores presentes nesta chave são utilizados para aceder aos elementos contidos nos dicionários encontrados na chave *events* e *property*. Esta redundância de informação existe para simplificar o acesso aos valores contidos nos dicionários de *events* e *property* reduzindo o número de ciclos necessários para processar ambos os dicionários, assim como permitir um processamento ordenado dos valores dos dicionários, uma vez que não existe garantias de um dicionário manter a ordem das suas chaves.

A chave *events* é utilizada para efetuar a subscrição de eventos, associando os elementos da interface transmitida pela mensagem aos eventos que devem despoletar no dispositivo. A utilização desta chave é explicada em detalhe na secção 4.2.1.4.

A chave *property* possui um dicionário com um subconjunto dos valores contidos no *array* de *ids*. Cada uma destas chaves possui um dicionário com propriedades que podem ser manipuladas pela aplicação no dispositivo. A tabela 3.2 contém o exemplo dos valores implementados no desenvolvimento desta dissertação.

A chave *mode* pode assumir qualquer valor e serve para identificar o ecrã transmitido para o dispositivo móvel. A utilização deste valor permite efetuar a *cache* dos ecrãs transmitidos. Isto é, após a primeira transmissão de um ecrã, é possível enviar uma mensagem reduzida de interface, apenas com o modo e tipo de mensagem (listagem 4.2), uma vez que o dispositivo já tem guardada a interface a adicionar. Isto torna a comunicação mais eficiente, uma vez que reduz a quantidade de informação transmitida nas mensagens de manipulação da interface.

```
{
   type: "addScreen",
   mode: "MainMode.SubMode1"
}
```

Listagem 4.2: Exemplo de mensagem reduzida de interface

Para além destas duas mensagens, o servidor tem ainda acesso a mensagens do tipo *modifyScreen*, exemplificada na listagem 4.3.

```
{
   type: "modifyScreen",
   ids: ["rotateButton", "transformButton", "scaleButton"],
   events: {
               rotateButton: ["swipe"]
           },
   property: {
           "transformButton": {
               "hidden": "false"
           },
           "scaleButton": {
               "disabled": "false",
               "background": "#555555"
           }
       }
}
```

Listagem 4.3: Exemplo de mensagem utilizada para modificar o ecrã

Esta mensagem, para além da chave *type* e *ids*, define as chaves *events* e *property* explicadas nos parágrafos anteriores.

4.2.1.2 Mensagens de eventos

Após um elemento subscrever um evento, a aplicação móvel informa o servidor quando o evento é despoletado. Estas mensagens assumem a estrutura apresentada na listagem 4.4.
Nestas mensagens, a chave *type* assume um dos valores apresentados na tabela 3.3 para identificar o evento em questão. Para além disso é enviado no corpo da mensagem o id que despoletou o evento, assim como a informação relevante que permite ao servidor executar a função de processamento correta.

```
{
   type: "tap",
   totalTouches: 1,
   targetIDs: ["elementID"],
   coordinate: [{x: 100, y: 150}],
   direction: 2,
   scale: 1.5,
   rotation: 100,
   deltaTime: 100
}
```

Listagem 4.4: Exemplo de mensagem de um evento tap

4.2.1.3 Mensagens de informação

As mensagens de informação, especificadas na listagem 4.5 garantem o sucesso ou insucesso da transmissão de informação entre o servidor e o dispositivo móvel. Esta mensagem é utilizada por ambos os participantes, seja como confirmação de receção ou para transmitir informação sobre erros encontrados durante o processamento da ultima mensagem recebida.

```
{
  type : "info",
  error : "true", //ou "false"
  errorDescription : "Mensagem Invalida"
}
```

Listagem 4.5: Exemplo de mensagem de confirmação

4.2.1.4 Subscrição de eventos

A subscrição de eventos é possível utilizando a chave *events* das mensagens de manipulação de interface especificadas na secção 4.2.1.1.

A chave *events* contém um dicionário cujas chaves são os ids dos elementos que vão subscrever os eventos. Cada uma destas chaves contém um *array* com o tipo de eventos a que o objeto deve subscrever. Os valores que podem ser utilizados para subscrever eventos são os valores apresentados na tabela 3.3.

Para desativar a subscrição de eventos existem dois métodos diferentes. No caso do acelerómetro, uma vez que o sensor está continuamente a enviar informação para a aplicação e exige um maior processamento de dados[1], foi definido o evento *stop-aceleration*. Se a aplicação encontrar numa mensagem este evento associado a um elemento da interface, remove a subscrição de eventos do acelerómetro.

Para os eventos do ecrã multi-toque, a anulação da subscrição do evento é feita a partir da *property disabled*. Quando um elemento é colocado como *disabled* o servidor deixa de ser notificado sobre eventos de toque. Isto torna a aplicação mais rápida, não necessitando de estar constantemente a subscrever (ou cancelar a subscrição) o *handler* do evento.

4.2.2 Reconhecimento de Gestos

A *SDCI* é capaz de efetuar reconhecimento de gestos efetuados sobre o ecrã multi-toque. Para ser capaz de executar esta funcionalidade, foi utilizado o *$N Multistroke Recognizer* (*$N*) [LA16]. O *$N* permite o reconhecimento de gestos com múltiplos traços desenhados numa sequência arbitrária.

Para o *$N* poder efetuar o reconhecimento de gestos, a especificação destes deve ser transmitida para o dispositivo móvel utilizando o protocolo de comunicação. Assim, quando o servidor pede o reconhecimento de gestos sobre um objeto, deve acrescentar na mensagem enviada uma chave *gestures* (exemplificado na listagem 4.6), que contém a especificação dos gestos a serem reconhecidos.

[1] É necessário, a partir dos dados do sensor, calcular velocidades, distâncias entre outros

```
{
   type: "modifyScreen",
   ids: ["gestureDiv"],
   gestures: <GestureGraph>...</GestureGraph>,
   events: {
               rotateButton: ["draw"]
           }
}
```

Listagem 4.6: Exemplo de mensagem de novo ecrã

Esta secção propõe uma especificação em *XML* para efetuar a definição dos gestos.

```
<?xml version="1.0"?>
<GestureGraph>
  <Gestures>
    <Gesture name="Cube">
      <Stroke>
        <Point x="86" y="348" />
        <Point x="86" y="347" />
        <Point x="86" y="345" />
        <Point x="86" y="343" />
        <Point x="86" y="340" />
        <Point x="86" y="337" />
        <Point x="86" y="334" />
        <Point x="86" y="332" />
        <Point x="86" y="331" />
      </Stroke>
      <Stroke>
        <Point x="86" y="332" />
        <Point x="86" y="331" />
        <Point x="86" y="329" />
        <Point x="86" y="349" />
        <Point x="86" y="348" />
        <Point x="87" y="347" />
        <Point x="87" y="346" />
        <Point x="87" y="346" />
        <Point x="87" y="345" />
        <Point x="87" y="344" />
      </Stroke>
    </Gesture>
  </Gestures>
</GestureGraph>
```

Listagem 4.7: Exemplo reduzido do xml utilizado na implementação da framework

Como é possível identificar na listagem 4.7 um gesto é identificado por um elemento *Gesture* com o atributo *name*, obrigatório para ser possível a identificação do gesto após o seu reconhe-

cimento. Este elemento contém um ou mais elementos *Stroke* que correspondem aos diferentes traços que compõem o gesto. Por sua vez, os elementos *Stroke* são compostos por elementos *Point* que contêm como atributos as coordenadas do ponto. Finalmente toda esta estrutura encontra-se agregada sobre um elemento *Gestures*.
A aplicação incluída com a *framework* implementa um *parser* para esta estrutura *XML*.

4.3 Desenvolvimento

Esta secção descreve como foi implementada a *framework SDCI* utilizando as tecnologias e especificação referida nas secções anteriores. Procura também entrar mais em detalhe sobre o funcionamento da *framework* no dispositivo móvel, no computador e como é feita a interação entre os dois utilizando o protocolo especificado pela *SDCI*.

4.3.1 Dispositivo Móvel

Para implementar a arquitetura definida pela *framework SDCI*, especificada no capítulo 3, utilizando a tecnologia *Ionic*, foi utilizada a estrutura apresentada na figura 4.1.

Como primeiro componente da arquitetura temos o *SocketService*. O *SocketService* é implementado utilizando *Services* de *Angular.Js*. Estes serviços de *Angular.Js* são equiparáveis a classes *singleton*. Podem ser injetados como dependência noutros serviços, e todos eles recebem a mesma instância do objeto. Assim, o *SocketService* possui toda a lógica necessária para o funcionamento da *socket*, desde funções de conexão e envio de mensagens até ao *loop* de receção de mensagens.

O *Parser*, como referido no capítulo 3 é responsável por validar o tipo das mensagens transmitidas, assim como os parâmetros obrigatórios para cada tipo de mensagem. No caso da aplicação móvel o *Parser* implementa dois estados de funcionamento, o estado "*Starting*" onde é feita a configuração da aplicação e o estado "*Working*"que representa o estado normal de funcionamento da aplicação.
O estado inicial está ativo até a primeira interação com o servidor ser concluída. No primeiro estado o *Parser* encontra-se à espera de uma mensagem do tipo *newScreen* enviada pelo servidor. Recebida esta mensagem, o *Parser* muda para o segundo estado no qual passa a aceitar qualquer tipo de mensagem de manipulação de interface ou de *acknowledgement*.

O *InterfaceService* é o serviço responsável pela implementação da componente *InterfaceHandler* especificada no capítulo 3. Este serviço contém as funções necessárias para processar os diferentes tipos de mensagem de manipulação de interface enviadas pelo servidor. Dependendo do tipo de mensagem recebida, este serviço executa as tarefas necessárias desde adicionar uma interface recebida, até modificar as propriedades da interface presente.

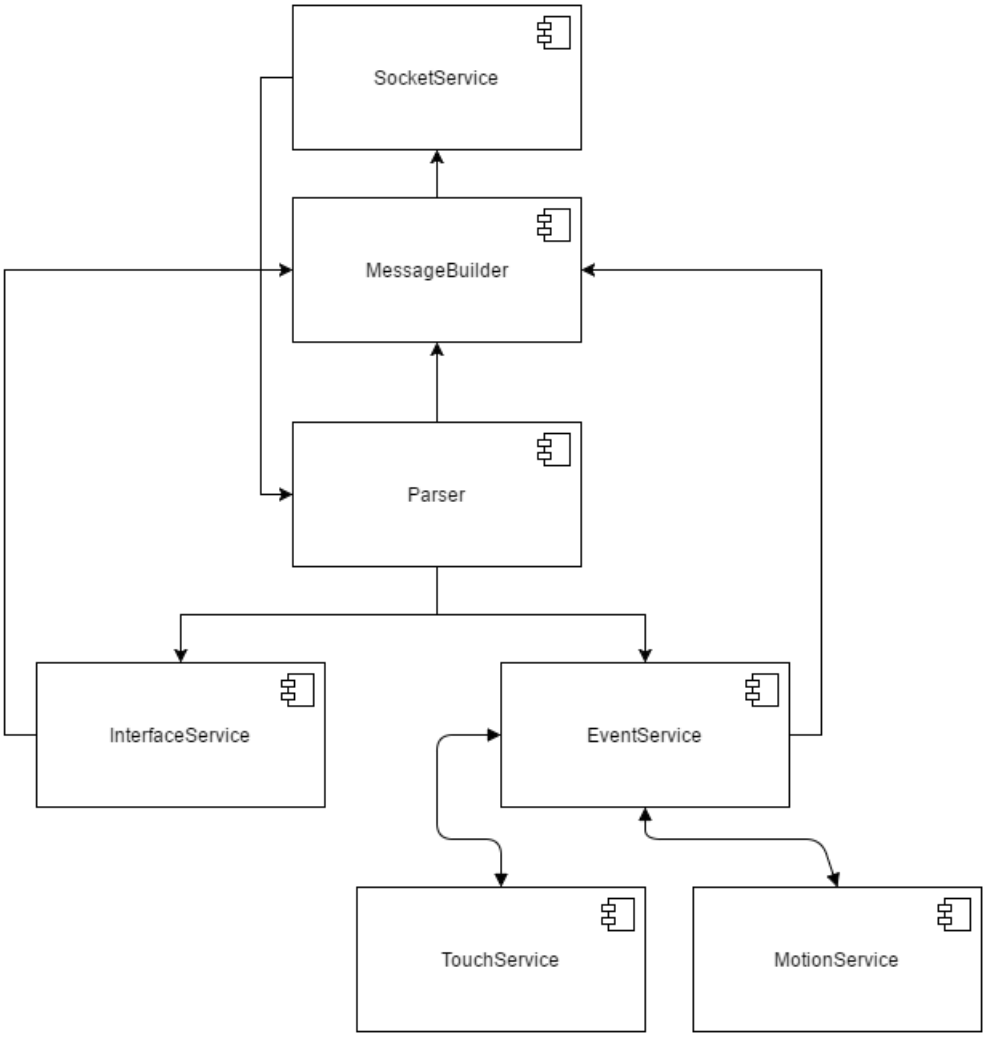


Figura 4.1: Representação da arquitetura implementada na aplicação móvel

O *EventService* é o serviço que implementa a componente *EventHandler* também especificada no capítulo 3. O *EventService* regista as funções despoletadas pelos eventos do ecrã multi-toque e do acelerómetro e associa-as aos elementos que subscrevem os eventos.

Para realizar a subscrição dos eventos foram implementados dois serviços auxiliares, o *TouchService* e o *MotionService*. Estes serviços definem as funções necessárias para subscrever os eventos de toque e do acelerómetro respetivamente. O *TouchService* faz uso da biblioteca *Hammer.js* para subscrever os eventos de toque, assim como possui funções que permitem a subscrição de eventos utilizando as funções públicas de *Javascript*. O *MotionService* utiliza apenas as funções da linguagem para subscrever os eventos do acelerómetro.

4.3.2 Computador

No computador como referido anteriormente, foi desenvolvida uma extensão em *Python* para o *Blender*. Esta extensão permitiu, utilizando a *framework* desenvolvida, a interação com o *Blender* através de toque e do acelerómetro. A extensão permitia efetuar operações geométricas (mover, escalar e rodar), assim como criar novas primitivas como: cubos, cilindros e cones.

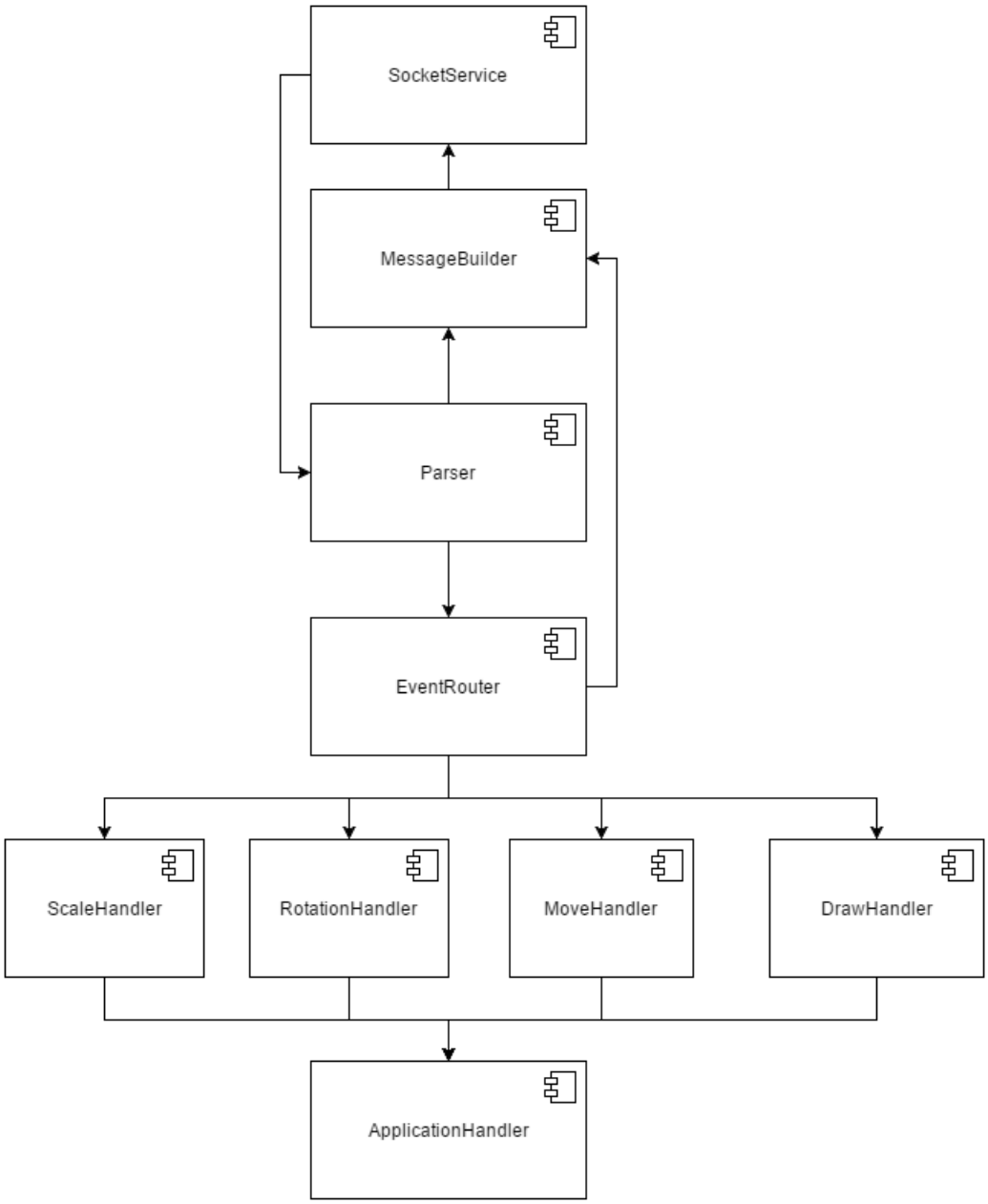


Figura 4.2: Representação da arquitetura implementada no servidor

Na implementação desta extensão foi seguida a arquitetura apresentada na figura 4.2.

Nesta arquitetura, tanto o *SocketService* como o *Parser* têm implementações e funções semelhantes às da aplicação do dispositivo móvel.

O *EventRouter* é a classe responsável por registar os diferentes *handlers* necessários para cada elemento que subscreva eventos na aplicação móvel. Para atingir este objetivo utiliza um dicionário para associar o *id* do elemento a funções definidas pelos diferentes *Handlers* da camada seguinte.

Na camada seguinte são definidas várias classes que implementam os *handlers* para as mensagens recebidas. Por exemplo, a classe *ScaleHandler* implementa funções que devem ser executadas após a receção de mensagens de eventos, e que devam ser processadas para efetuar uma operação de escalamento na aplicação final (o *software* de modelação 3D *Blender*).

A classe *ApplicationHandler* define as funções que interagem diretamente com a *API* disponibilizada pelo *Blender*.

4.3.3 Fluxo de dados

Assume-se, aqui, que a aplicação já estabeleceu conexão com o servidor e que se encontra pronta a ser utilizada. Esta secção explica com mais detalhe o fluxo de interações entre as diferentes camadas do sistema, desde o *input* do utilizador no dispositivo móvel até ao envio da resposta no servidor.

Recebendo, no dispositivo móvel, *input* do utilizador, os *handlers* registados pelo *EventService* do dispositivo são invocados e processam os dados recebidos pelo sensor, calculando, por exemplo, distâncias e a velocidade do movimento. De seguida é enviada uma mensagem, como exemplificado na listagem 4.4 para o servidor.

Após a receção da mensagem no servidor, a partir da *socket* instanciada no *SocketService*, a mesma é transmitida ao *Parser* que valida os parâmetros existentes na mensagem. Terminada a validação da mensagem pelo *Parser*, esta é enviada para o *EventRouter*.

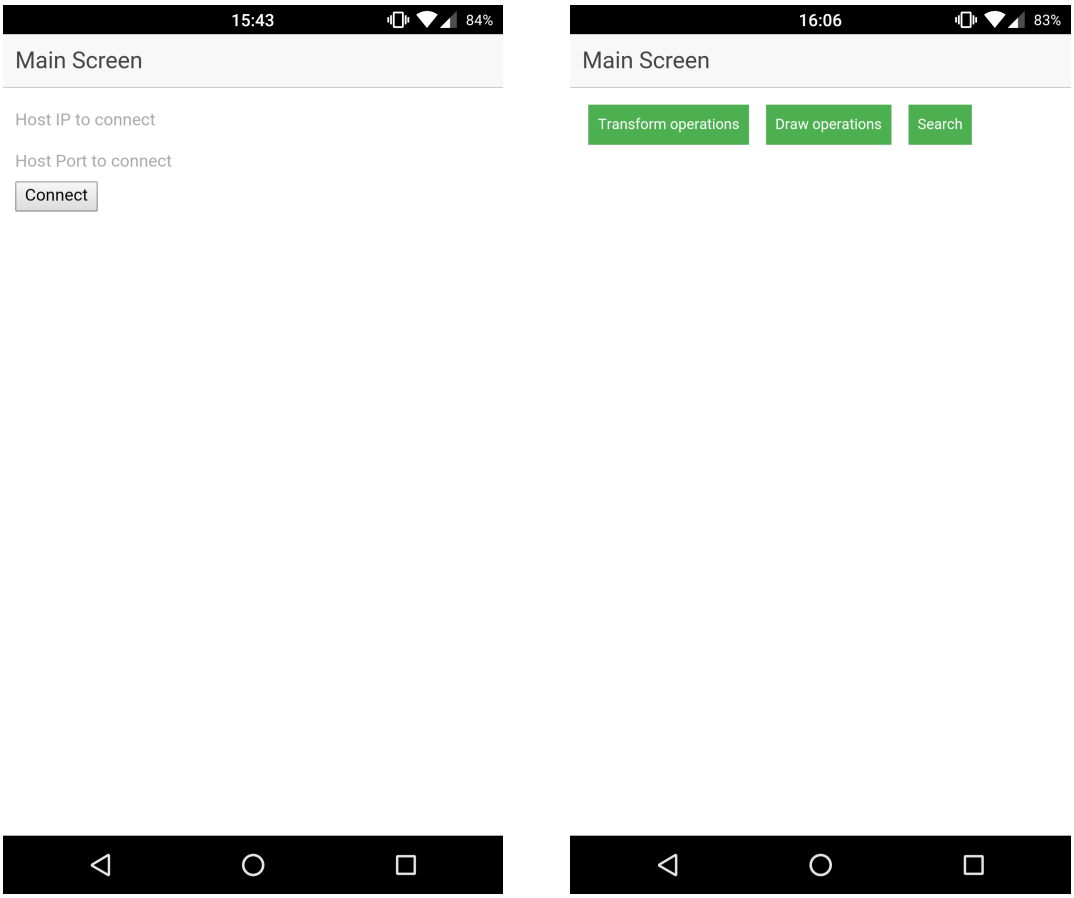
O *EventRouter* do servidor, na posse da mensagem, procura a função correta a executar. A função é identificada a partir do id do elemento contido na mensagem. Este id deve ter uma função associada num dicionário presente no *EventRouter*. Encontrada a função, esta é executada e processa os valores recebidos pelo *EventRouter*, invocando de seguida o *ApplicationHandler*.

Concluídas as ações sobre o *Blender*, o *handler* invoca a classe *MessageBuilder*. Esta classe define vários métodos que constroem as mensagens de resposta a enviar para a aplicação móvel.

Finalmente, no dispositivo móvel, recebida e concluída a validação da resposta pelo *Parser*, o mesmo invoca o *InterfaceService*. Este serviço contém as funções necessárias para processar os diferentes tipos de mensagem de manipulação de interface enviadas pelo servidor. Dependendo do tipo de mensagem recebida, este serviço executa as tarefas necessárias desde adicionar uma interface recebida a modificar as propriedades da interface presente. Terminada esta fase, o *EventService* é o próximo serviço a processar a mensagem para proceder à subscrição dos eventos requisitados pela mensagem.

4.4 Funcionalidades implementadas

A aplicação desenvolvida para o dispositivo móvel faz uso de todas as funcionalidades fornecidas pelo protocolo definido em 3.3. Isto implica que a aplicação recebe todos os seus ecrãs a partir do servidor após estabelecer uma conexão.
Isto permite que esta aplicação possa ser reutilizada para diferentes propósitos conectando-se a diferentes servidores. Apesar de a aplicação aqui ser utilizada com o servidor desenvolvido e interagir com o *Blender*, seria possível desenvolver outro servidor para outros objetivos.



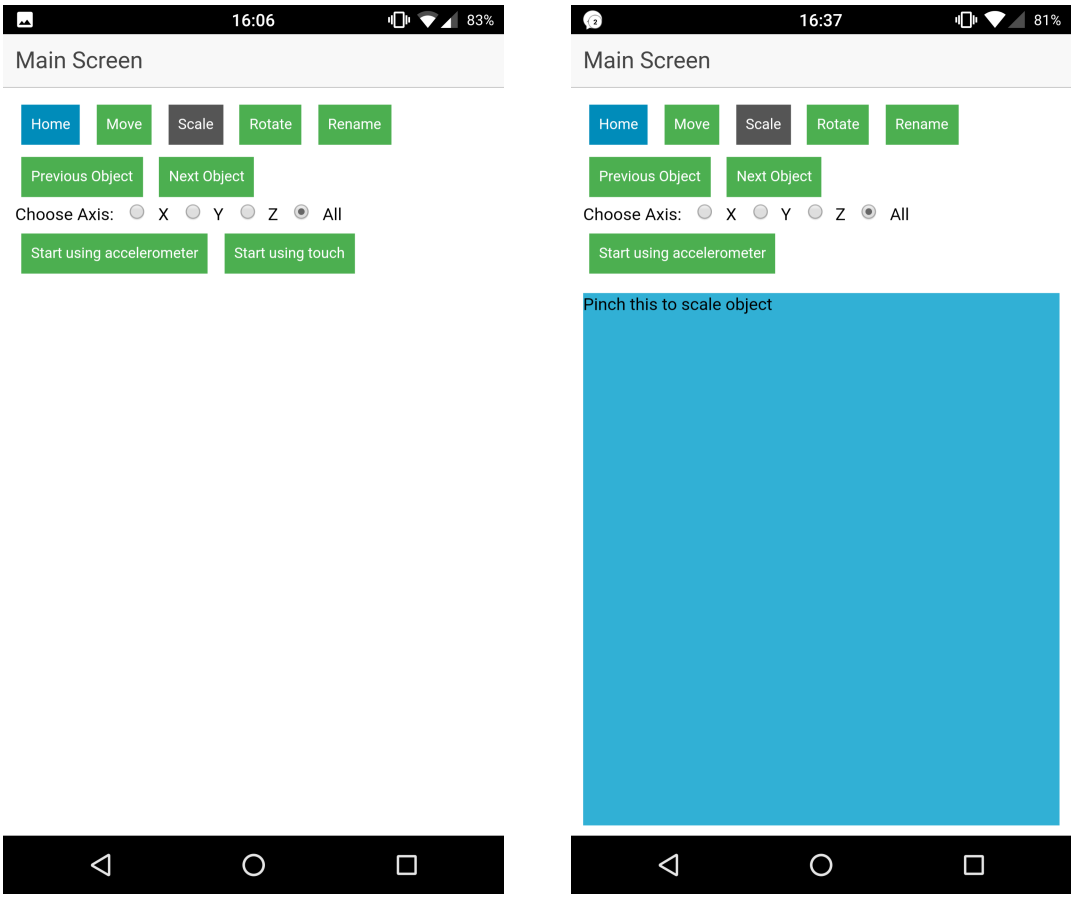
(a) Ecrã inicial da aplicação móvel

(b) Primeiro ecrã da aplicação servido pelo computador

Figura 4.3: Ecrãs iniciais da aplicação

Na figura 4.3a é possível visualizar o primeiro e único ecrã implementado na aplicação móvel. Este ecrã tem como objetivo recolher as informações necessárias para conectar com o servidor externo. Após ser estabelecida a conexão com o servidor, a aplicação passa a exibir um ecrã enviado pelo servidor, apresentado na figura 4.3b.

Assim, o servidor permite ao utilizador da aplicação móvel efetuar uma variedade de operações sobre o *Blender* (figura 4.4). É possível efetuar operações geométricas sobre objetos selecionados pelo *Blender* ou pela aplicação. Estas operações geométricas incluem, mover, escalar e rodar os objetos, sendo possível, de forma complementar, definir o método de interação a utilizar para a respetiva operação: acelerómetro ou toque.



(a) Ecrã operação de escalamento

(b) Operação de escalamento utilizando toque

Figura 4.4: Ecrãs de operações

Estes dois métodos de interação estão presentes para todas as operações geométricas para permitir demonstrar as funcionalidades da *framework*, assim como para permitir uma comparação entre os dois métodos de interação como veremos no capitulo 5.

Selecionando a opção do acelerómetro, o servidor desencadeia um conjunto de operações: este botão troca de cor e o texto apresentado é alterado, demonstrando que a sua funcionalidade foi modificada, desligando, agora, a utilização do acelerómetro.
Se, em alternativa, a opção de toque for selecionada, a aplicação passa a apresentar o ecrã presente na figura 4.4b. A área azul apresentada é uma área sobre a qual o utilizador pode aplicar um gesto de *pinch* para aplicar a operação de escalamento sobre o objeto selecionado.

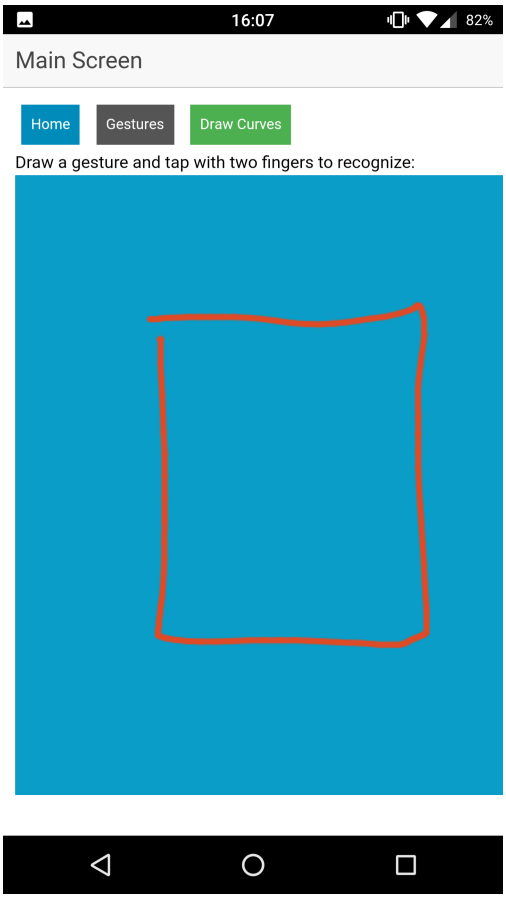


Figura 4.5: Ecrã de reconhecimento de gestos

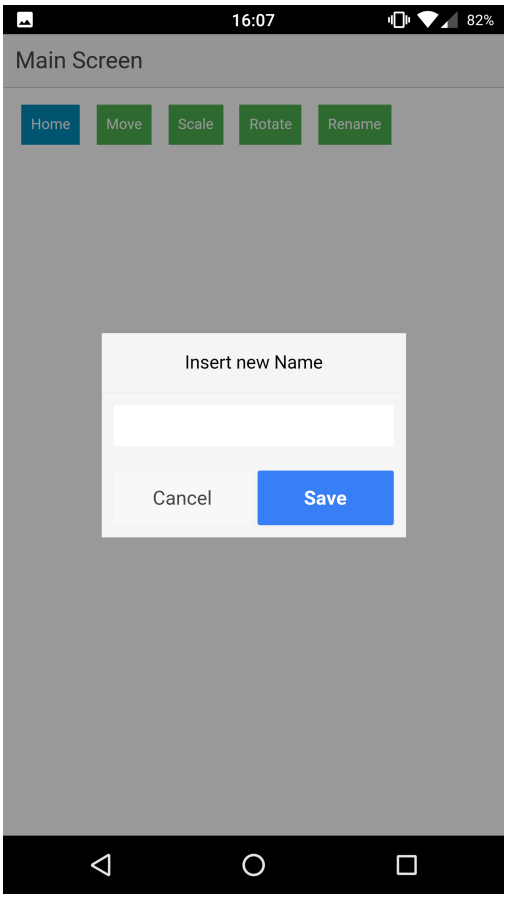


Figura 4.6: Alerta para renomear um objeto

Para além das operações geométricas é também possível criar primitivas no *Blender*, como cubos ou cilindros. Para implementar esta funcionalidade foi utilizado o reconhecimento de gestos disponibilizado pela *framework*. Na figura 4.5 visualiza-se o ecrã de reconhecimento de gestos. Como podemos ver pela imagem, a partir do desenho de um simples quadrilátero é possível criar um cubo no *Blender*. Os gestos que podem ser identificados pela aplicação são especificados pelo servidor, seguindo o formato *XML* apresentado na secção 4.2.2. A aplicação, após reconhecer o gesto, informa o servidor do gesto identificado, e o mesmo associa este gesto à primitiva do cubo.

Uma vez que a criação de primitivas pelo *Blender* atribui nomes automáticos aos objetos, foi também implementada uma funcionalidade que permite renomear estes objetos. Na figura 4.6 é possível visualizar um alerta que contém uma caixa de texto para introdução de um novo nome para o objeto. Após o utilizador confirmar o novo nome, este é enviado para o servidor onde é utilizado para alterar o nome do objeto atualmente selecionado.

Finalmente, apesar de o principal trabalho do *add-on* para o *Blender* ser executado no *background*, foi criada uma pequena interface que permita inserir os dados necessários para efetuar a ligação.

Na figura 4.7 vemos a interface criada no *Blender*. No painel do lado esquerdo, foi criado um separador *Server*, que possui campos para inserir o *ip* e a porta que o servidor deve utilizar para abrir a *socket*. Para iniciar o *add-on*, tudo o que o utilizador precisa de fazer é preencher estes dados e clicar no botão *Start Server*.

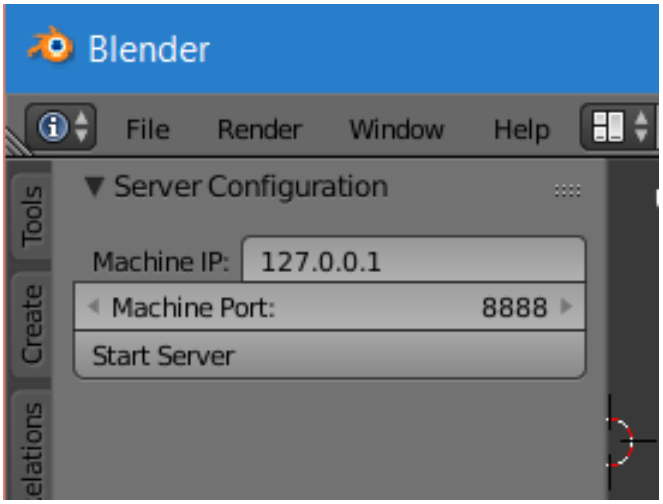


Figura 4.7: Interface criada no Blender (painel do lado esquerdo)

4.5 Conclusão

Neste capítulo apresentou-se uma descrição detalhada da implementação efetuada para a *framework SDCI* especificada no capítulo 3.

O capítulo inicia por uma apresentação das tecnologias utilizadas na implementação. A implementação desta *framework* abrange dois dispositivos, um dispositivo móvel e um computador. Para o desenvolvimento da aplicação móvel optou-se pela utilização da *framework* de desenvolvimento *Ionic*. O *Ionic* permite o desenvolvimento de aplicações multi-plataforma (*android* e *iOS*), utilizando as linguagens de programação *HTML* e *javascript*. Para o desenvolvimento da aplicação de computador, procurou-se uma aplicação que possibilitasse a extensão das suas funcionalidades e pudesse beneficiar das funcionalidades da *framework* proposta. Assim, optou-se pela aplicação de modelação 3D *Blender*. Esta aplicação permite o desenvolvimento de extensões que expandem as suas funcionalidades, e consegue tirar partido das várias funcionalidades propostas pela *framework SDCI*.

Após a apresentação das tecnologias, foi detalhadamente apresentada a especificação das mensagens do protocolo de comunicação especificado pela *framework SDCI*. Foram especificadas e exemplificadas as diferentes mensagens do protocolo: mensagens de interface, mensagens de eventos e mensagens de informação.

De seguida, apresentou-se a arquitetura utilizada na implementação da *framework* nos dois dispositivos, no dispositivo móvel e no computador. As arquiteturas utilizadas nos dois dispositivos expandem sobre a arquitetura especificada na secção 3.2. Para além disso, explicou-se a interação entre os dois dispositivos, assim como entre as várias componentes da arquitetura de cada dispositivo.

Finalmente, foi apresentada uma descrição das funcionalidades implementadas pela aplicação. Esta descrição aborda as funcionalidades implementadas pela extensão desenvolvida para o *Blender*, que fazem uso das vantagens que a *SDCI* fornece, como a utilização do reconhecimento de gestos para criar objetos no *Blender*, ou ainda, a utilização do acelerómetro ou do toque para a realização de operações geométricas.

Capítulo 5

Avaliação

Neste capítulo é descrito em detalhe o processo de avaliação utilizado para avaliar a usabilidade da *framework SDCI*.

5.1 Metodologia

Utilizando a extensão para o *Blender* especificada no capítulo 4, foi desenhado um caso de estudo que permitisse avaliar tanto a usabilidade como o funcionamento da *SDCI*. Este caso de estudo foi planeado para demorar cerca de cinquenta minutos por utilizador, incluindo vinte minutos de tempo máximo para este se familiarizar com o sistema.

Durante a aprendizagem, foram apresentados aos utilizadores dois guiões (anexo 7.1) que incluíam instruções para realizar tarefas básicas necessárias para a avaliação do sistema. Estes guiões explicavam passo a passo como criar novos objetos, realizar operações geométricas e selecionar objetos. Cada guião focava-se numa parte especifica do sistema; um possibilitava a aprendizagem da aplicação *Blender*, o outro a utilização da *framework* desenvolvida. Durante a fase de aprendizagem procedeu-se de uma forma mais interativa, com os utilizadores a serem inicialmente ajudados a interagir com a aplicação.

Na fase de avaliação foi pedido aos utilizadores para recriarem o modelo apresentado na figura 5.1, utilizando três métodos de interação diferentes. Este modelo era acompanhado por um guião (anexo 7.2) que indicava a ordem pela qual as atividades inerentes à construção do modelo deviam ser efetuadas. Os métodos de interação aplicados e avaliados durante a recriação do modelo foram os seguintes:

- Aplicação utilizando apenas multi-toque
- Aplicação utilizando apenas acelerómetro
- Aplicação utilizando apenas rato e teclado

Os primeiros dois métodos da lista utilizam a *framework* desenvolvida para permitir a interação com a aplicação. O último método da lista utiliza apenas o computador, com o rato e teclado. Todos os intervenientes executaram os três métodos, para permitir uma comparação mais direta sobre a eficácia de cada um. No entanto, a ordem da utilização dos métodos foi aleatória para minimizar o impacto dessa sequência nos resultados obtidos.

Figura 5.1: Modelo apresentado aos utilizadores da experiência

Para permitir a avaliação correta da *framework* e dos métodos de interação propostos foram retiradas várias métricas. As métricas contabilizadas durante a realização das três tarefas foram:

- Tempo de execução
- Erros cometidos pelo utilizador
- Erros da *framework* (por exemplo, falhas no reconhecimento de gestos)
- Atrasos percetíveis na comunicação entre os dispositivos

O tempo de execução da tarefa é uma das métricas mais importantes para possibilitar a comparação da eficiência entre os métodos de interação no cenário apresentado. O número de erros cometidos pelo utilizador representa o número de vezes que os utilizadores decidiram anular uma operação efetuada pelos próprios. Esta métrica permite comparar o número médio de falhas que os utilizadores cometem utilizando um dado método de interação.

Nas tarefas que envolviam a utilização da *framework*, foi contabilizado o número de erros da *SDCI*. Estes erros da *framework* foram contabilizados, principalmente, no reconhecimento de gestos livres, quando um gesto que deveria ser reconhecido pela *framework* era mal reconhecido. Por exemplo, um utilizador desenhar um cilindro, e este ser reconhecido como um quadrado era contabilizado como um erro da *framework*. A recolha do número de falhas da *framework* permite

avaliar a eficiência e usabilidade da mesma. Foi também contabilizado o número de vezes que a usabilidade foi comprometida devido a atrasos na comunicação. Esta métrica permite avaliar a eficácia do protocolo de comunicação definido quando aplicado em situações reais.

Entre a realização de cada tarefa, era possibilitado aos utilizadores realizar um intervalo de dois minutos. Durante este intervalo preparava-se a próxima tarefa a realizar e apresentava-se aos utilizadores um questionário com três perguntas (anexo 7.3).

Este questionário incluía uma pergunta para identificar o método de interação utilizado, e duas perguntas retiradas do *After-Scenario Questionnaire*. Este questionário foi escolhido por permitir, de acordo com *Lewis*, uma aplicação muito genérica e resultados fiáveis. Este questionário tem como objetivo avaliar do ponto de vista do utilizador a facilidade e o tempo com que completou a tarefa [Lew95, FPT12]. As perguntas utilizadas no questionário foram avaliadas utilizando uma escala *Likert* de sete pontos.

Finalmente, para permitir uma boa ligação com o servidor foi criado um *hotspot* a partir do dispositivo móvel, onde os únicos dispositivos conectados eram o computador e o próprio dispositivo. Esta medida foi efetuada para evitar influências sobre o teste devido à utilização da rede pública do ambiente utilizado nos testes.

Seguindo a metodologia apresentada, foi realizado o caso de estudo com treze voluntários. Destes treze voluntários, quatro foram retirados por falta de métricas recolhidas face aos restantes voluntários. Isto reduziu a amostra para um total de nove pessoas. Devido à reduzida amostra de voluntários estes testes não podem ser utilizados para retirar conclusões exatas sobre a *framework* ou sobre os métodos de interação avaliados. No entanto, podem ser vistos como um teste piloto permitindo identificar problemas na *SDCI* e avaliar a viabilidade de realizar outros testes com uma amostra mais considerável.

5.2 Resultados obtidos

Os resultados apresentados foram obtidos a partir de uma amostra de nove pessoas, sete homens e duas mulheres, com idades compreendidas entre os dezoito e trinta e cinco anos. Quando inquiridos sobre a sua experiência com ferramentas de modelação como o *Blender*, a maioria considerou-se pouco experiente obtendo uma média de 2,67 na escala de *Likert* com um desvio padrão de 1,66.

5.2.1 Resultados objetivos

Na figura 5.2 é apresentado um gráfico com a média do tempo de execução da tarefa para cada método de interação. Observando o gráfico é possível verificar que a utilização do rato e teclado possibilitou os tempos de execução mais rápidos, com uma média de 3,77 minutos para completar a tarefa. O método de interação com o pior tempo de execução foi a aplicação utilizando o acelerómetro com 8,91 minutos.

O uso da aplicação com toque posiciona-se ligeiramente melhor que o uso da aplicação com o acelerómetro com uma média de 7,09 minutos para completar a tarefa.

A figura 5.3 apresenta o gráfico com a média de operações anuladas por método de interação. Aqui a utilização do *Blender* com rato e teclado levou a um maior número de operações anuladas com uma média de 2,33 operações anuladas por tarefa.

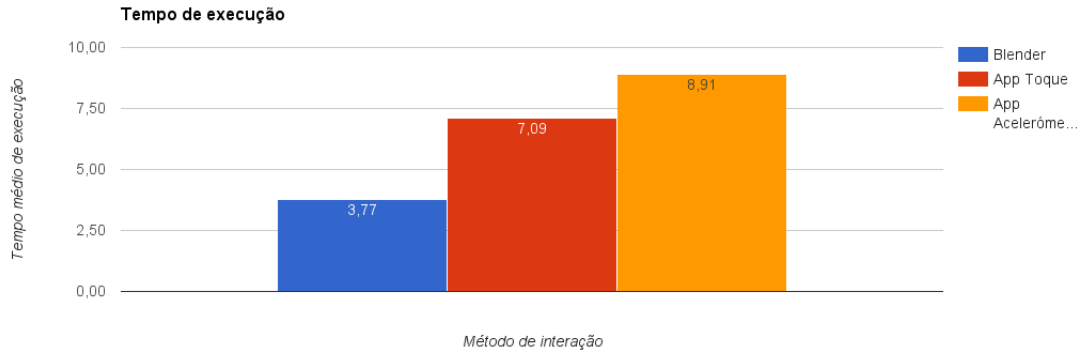


Figura 5.2: Gráfico do tempo de execução para cada método de interação em minutos

Por outro lado, e como se apresenta na figura 5.3, a aplicação com toque apresenta uma média de operações anuladas de 0,22 operações por sessão. Esta diferença pode ser explicada pela inexistência de um atalho na aplicação para anular a operação. Apesar dos utilizadores poderem utilizar o teclado para anular a operação, a utilização da aplicação móvel parece incentivar a correção de erros ao invés de recomeçar a operação. No caso do acelerómetro, para anular uma operação, primeiro era necessário terminar a operação de movimento e só depois utilizar o teclado para cancelar a operação. Como este processo é mais longo do que simplesmente corrigir a rota do objeto, isto pode ter contribuído para um número menor de operações anuladas.

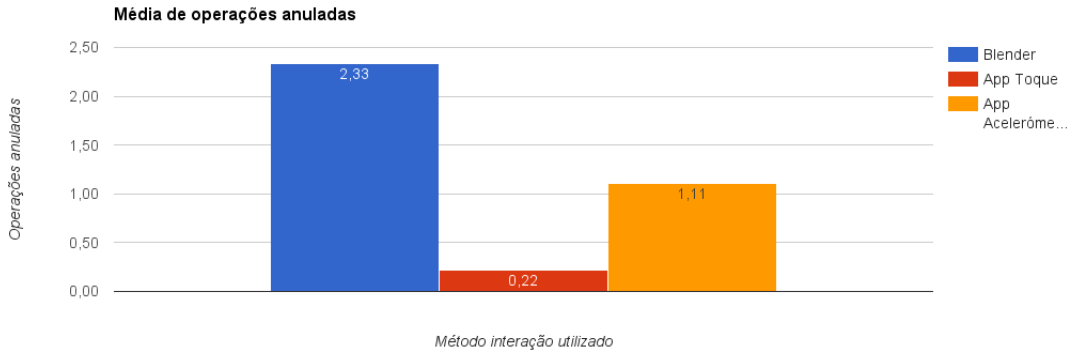


Figura 5.3: Gráfico com a média do número de operações anuladas por método de interação

Para analisar a eficácia da *framework SDCI* foram contabilizados o número de erros no reconhecimento de eventos e o número de vezes que o *delay* entre o dispositivo e o computador foi percetível.

Na figura 5.4 é apresentado o gráfico com a média de falhas no reconhecimento de eventos por parte da *SDCI*. Este número foi contabilizado sempre que um gesto que deveria ser reconhecido pela aplicação não era identificado ou sempre que um evento, por exemplo o toque num botão, deveria despoletar uma ação mas nada acontecia.

Nos dois métodos de interação que utilizavam a *framework*, a média de falhas no reconhecimento de eventos encontra-se entre três a quatro eventos não reconhecidos por sessão. Através da observação dos testes foi possível verificar que a maioria destes erros aconteceram devido a um grupo de botões radiais, utilizados na interface com toque e acelerómetro, que não reconheciam os toques sobre eles. Isto pode ter acontecido por estes botões apresentarem um tamanho mais reduzido do que os botões utilizados no resto da aplicação. Uma possível solução para este problema pode ser a configuração da biblioteca utilizada para o reconhecimento de eventos, uma vez que esta permite configurar valores mínimos de tempo e raio do toque efetuado antes de despoletar um evento.

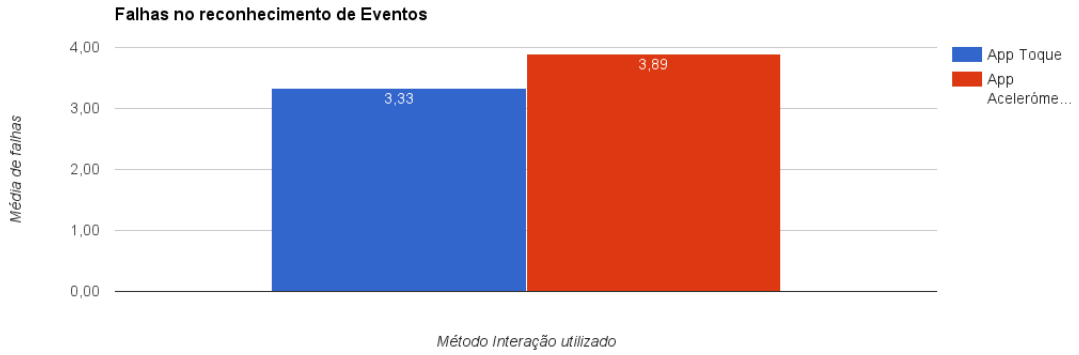


Figura 5.4: Gráfico com a média de falhas no reconhecimento de eventos por método de interação

Os erros de falhas no reconhecimento de eventos também apresentaram um desvio padrão mais elevado (2,45 e 1,62) do que o resto das métricas retiradas devido à dificuldade que alguns utilizadores tiveram com a funcionalidade de reconhecimento de gestos.

O gráfico apresentado na figura 5.5 apresenta a média do número de vezes que o *delay* entre eventos despoletados na aplicação do dispositivo e ações no computador foi percetível. Ambos os métodos de interação que utilizavam a *framework* apresentaram valores reduzidos para esta métrica, sendo os valores médios de 0,78 e 0,89 para o toque e acelerómetro respetivamente. Na observação dos testes foi possível verificar que estes *delays* aconteciam essencialmente nas operações que despoletavam o envio de várias mensagens num curto espaço de tempo para o servidor, como na operação de mover objeto. Este ponto poderia ser melhorado reduzindo o número de mensagens enviadas para o servidor neste tipo de eventos, mas isto exigiria um tratamento dos eventos retidos na aplicação móvel para possibilitar na mesma ao servidor efetuar as operações sem saltos entre os valores recebidos.

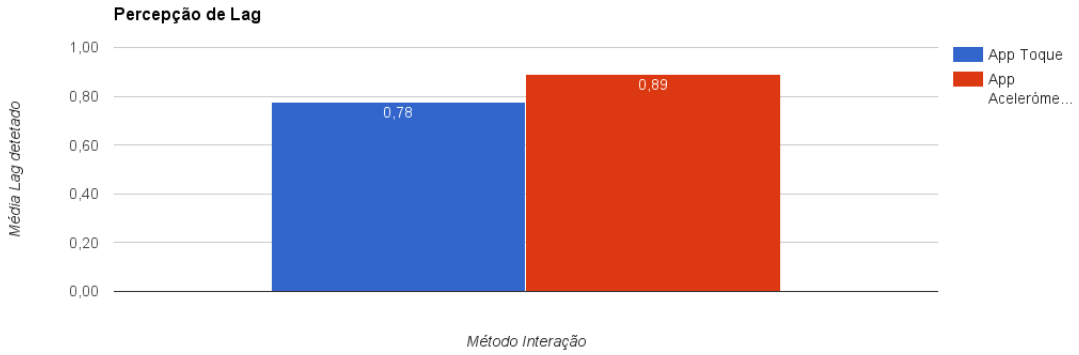


Figura 5.5: Gráfico com a média de *delays* detetados por método de interação

5.2.2 Resultados subjetivos

Nesta subsecção são apresentados os resultados obtidos a partir do questionário colocado aos utilizadores, assim como um resumo dos comentários feitos por eles durante a experiência.

Uma das questões a responder no final de cada tarefa era sobre o grau de satisfação do utilizador com o tempo necessário para completar a tarefa. Esta e a pergunta seguinte foram avaliadas sobre uma escala de sete pontos de *Likert*, correspondendo o número um à opção "discordo completamente"e o número sete à opção "concordo completamente".

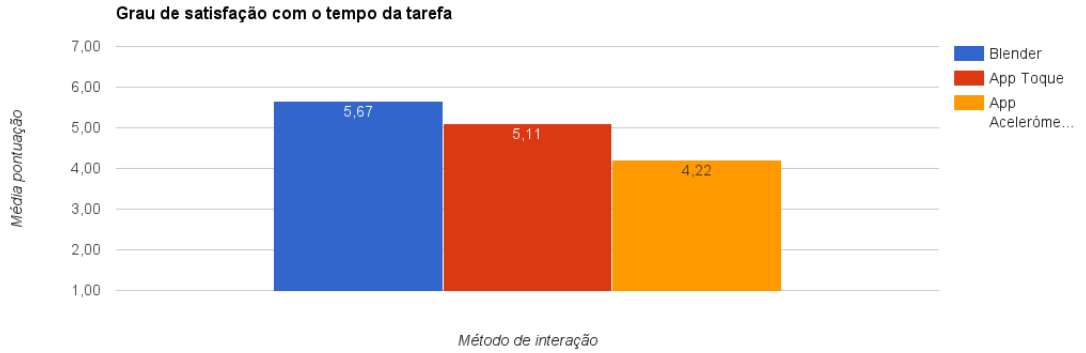


Figura 5.6: Gráfico sobre o grau de satisfação dos utilizadores com o tempo em que completaram a tarefa

Como podemos ver no gráfico da figura 5.6, a utilização do *Blender* com o rato e teclado obteve a melhor classificação com uma pontuação média de 5,67, seguida pela utilização da aplicação com toque com uma pontuação média de 5,11. Finalmente, o uso da aplicação com o acelerómetro obteve a pior pontuação com um valor de 4,22. Os valores obtidos nesta questão refletem os valores retirados para o tempo de execução de cada tarefa, onde o uso do rato e teclado foi mais rápido enquanto a utilização da aplicação com o acelerómetro foi significativamente mais lenta.

A outra pergunta a responder no final de cada tarefa era sobre o grau de satisfação do utilizador com a facilidade que completou a tarefa.

Nesta questão o método que obteve melhor pontuação foi a utilização da aplicação com toque, como podemos ver pelo gráfico da figura 5.7.

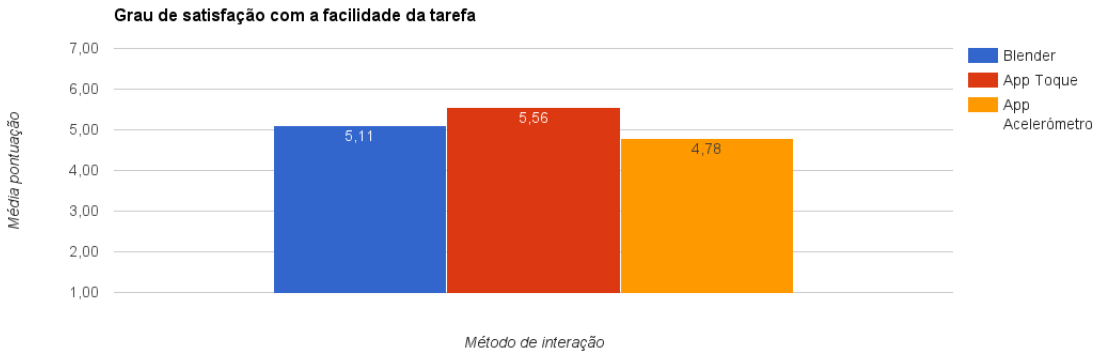


Figura 5.7: Gráfico sobre o grau de satisfação dos utilizadores com a facilidade com que completaram a tarefa

Apesar dos utilizadores demonstrarem estar conscientes que concluíram a tarefa mais eficientemente apenas com o teclado e o rato, a partir dos resultados obtidos nesta questão demonstrou-se alguma afinidade pelo uso do toque para este tipo de operações. A utilização da aplicação com toque obteve uma pontuação média de 5,56 seguida de perto pelo uso do teclado e rato com 5,11. Por último aparece a aplicação com acelerómetro com uma pontuação de 4,78.

No final de cada sessão era colocada aos utilizadores uma última questão sobre o método que preferiram utilizar. Esta pergunta assume um teor mais subjetivo do que as anteriores, mas permite obter uma ideia sobre a apreciação dos utilizadores sobre estes métodos de interação.

Como é possível observar no gráfico da figura 5.8, e refletindo as respostas da última questão apresentada, o método preferido pelos utilizadores foi a aplicação com toque. O segundo método preferido pelos utilizadores foi a utilização da aplicação com o acelerómetro, seguido finalmente pelo o uso de teclado e rato. A presença da aplicação com o acelerómetro como segundo método preferido apresenta-se em contraposição com os resultados das métricas e das questões avaliadas anteriormente, onde o acelerómetro apresenta uma eficiência visivelmente pior.

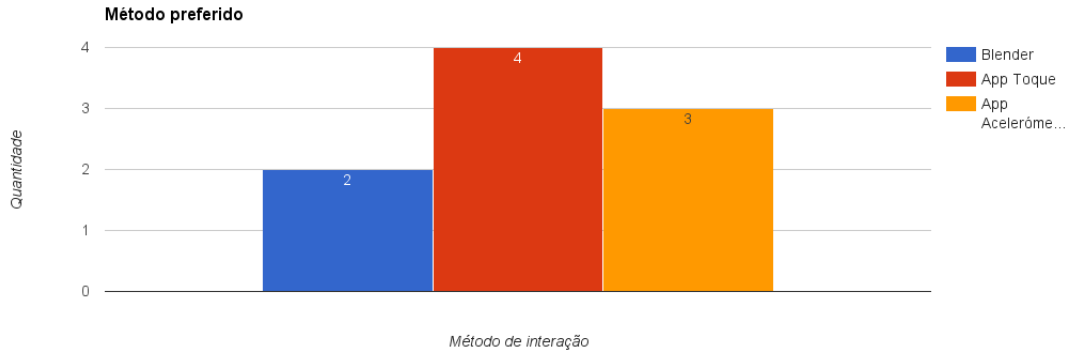


Figura 5.8: Gráfico sobre o método preferido dos utilizadores

5.2.3 Comentários recolhidos

Esta secção apresenta um resumo dos comentários realizados pelos utilizadores durante e após a realização dos testes.

Os utilizadores identificaram durante os seus comentários dois problemas com a aplicação. O uso dos botões radiais foi identificada como difícil pelos utilizadores, devido ao seu tamanho reduzido, uma vez que por várias situações eram necessários vários toques para ser despoletado o evento sobre o botão. Como referido anteriormente isto pode ser melhorado por uma melhor configuração de alguns parâmetros da biblioteca utilizada para o registo de eventos.

A utilização do acelerómetro também foi referida como problemática, devido ao método utilizado para terminar a recolha de eventos. Na implementação atual, os utilizadores precisavam de carregar num botão para ativar o uso do acelerómetro, e carregavam novamente quando quisessem parar o movimento. Isto tornou-se problemático porque obrigava os utilizadores a saber constantemente onde se encontrava o botão responsável por terminar o movimento. Este erro não está diretamente relacionado com a *framework*, mas sim com a interface desenhada pelo servidor. Um método possível para resolver este problema pode ser a implementação de um ecrã que permita clicar em qualquer lado para terminar o movimento. Isto liberta os olhos do utilizador para o computador não necessitando que este mantenha presente a localização do botão.

Alguns utilizadores também se sentiram confusos com o método utilizado para mover objetos com o toque. Na implementação atual, mover um objeto com o toque requeria um movimento horizontal sobre o ecrã, independente do eixo selecionado. Esta decisão foi tomada para manter alguma uniformidade no gesto necessário para efetuar o movimento, uma vez que não se encontrou um método confortável que pudesse ser utilizado para mover o objeto sobre o terceiro eixo.

No entanto, vários utilizadores referiram que teria sido mais intuitivo efetuar um movimento vertical quando moviam o objeto sobre o eixo vertical.

Os utilizadores referiram também um atraso incomodativo quando o ecrã de reconhecimento de gestos era ativado. Este atraso deve-se à *framework* carregar os gestos a serem reconhecidos no momento que o ecrã é requisitado. Isto poderia ser melhorado carregando os gestos para a biblioteca de reconhecimento numa fase inicial de preparação da aplicação. Esta modificação removeria este *delay* e com uma pequena modificação na biblioteca de reconhecimento de gestos poderia ser possível impedir o reconhecimento de gestos que não pertencessem ao ecrã atualmente a usar a funcionalidade. A implementação não foi inicialmente executada desta maneira, uma vez que os atrasos não tinham sido percetíveis anteriormente à experiência.

Finalmente, alguns utilizadores sugeriram funcionalidades adicionais para a extensão como a adição de um botão para duplicar objetos e para anular operações, assim como a possibilidade de controlarem o *viewport* do *Blender* utilizando o dispositivo móvel.

5.3 Conclusão

Neste capítulo foi apresentado em detalhe o caso de estudo realizado para permitir a avaliação da *framework* desenvolvida.

Apesar da amostra reduzida deste caso de estudo, como explicado na secção anterior, é possivel retirar alguns indicadores sobre a usabilidade e vantagem dos métodos de interação propostos, assim como a eficiência da *framework* proposta.

Analisando os resultados apresentados na secção 5.2.1, é possível concluir que a utilização do *Blender* com rato e teclado continua a ser o método mais eficaz e rápido para a execução deste tipo de tarefas. Apesar deste método de interação apresentar um maior número de operações anuladas, isto pode ser devido à natureza mais interativa que os métodos propostos apresentam. A inexistência de um botão para anular operações no dispositivo móvel também inclina os utilizadores para procederem à correção dos erros ao invés de anularem a operação, mesmo quando os utilizadores eram incentivados a utilizar o teclado e o rato como complemento da aplicação quando necessário.

Apesar de as métricas retiradas indicarem, como método mais eficiente, a utilização do rato e teclado, os resultados obtidos pelo questionário, apresentados na secção 5.2.2 indicam que existe alguma preferência pela utilização do toque como método de interação. As respostas ao questionário colocado entre tarefas indicam que os utilizadores sentiram que efetuaram a tarefa mais rapidamente com o rato e teclado. No entanto os utilizadores indicaram sentir uma maior facilidade na realização da tarefa utilizando o método de toque. Estes valores vão de encontro com os

comentários feitos durante a realização da experiência onde os utilizadores referiram que sentiam que a utilização do toque era bastante mais intuitiva.

Dos três métodos avaliados durante a experiência a utilização do acelerómetro foi a menos satisfatória. A utilização da aplicação com o acelerómetro apresentou os piores tempos de conclusão da tarefa proposta, assim como a pior usabilidade de acordo com os questionários realizados. Estes resultados podem ser explicados pelo facto de este ser um método de interação menos comum para os utilizadores. É possível que uma fase de treino mais direcionada para o uso do acelerómetro, ou a possibilidade de os utilizadores calibrarem a sensibilidade do sensor, se traduzisse em melhores resultados. No entanto, este foi considerado o segundo método de interação preferido pelos utilizadores.

No geral, a *framework* apresentou uma eficiência satisfatória, mas definitivamente com espaço para melhorias. Como indicado pelos utilizadores durante a realização da experiência, a utilização dos botões radiais foi complicada devido à aplicação falhar no reconhecimento de toques sobre estes botões. Este problema aconteceu devido ao tamanho mais reduzido destes botões face ao resto da interface apresentada. Isto pode ser resolvido experimentando um pouco com as configurações possíveis da biblioteca utilizada para o reconhecimento de eventos de toque. Esta biblioteca permite a configuração de valores mínimos para a duração e raio do toque detetado necessário para despoletar um evento.

Para além disto, a aplicação apresentou algum *delay* em determinadas operações. Ações que despoletaram um elevado número de eventos a serem enviados para o servidor, como o uso do acelerómetro, algumas vezes causaram atrasos percetíveis ao utilizador entre o movimento efetuado e a realização da operação no computador. Apesar deste fator ser fortemente afetado por condições externas, como a qualidade e tipo de ligação utilizada, na realização do teste foram tomadas precauções para reduzir estas influências. Contudo, se estes atrasos foram visíveis apesar das prevenções efetuadas, significa que o tratamento deste tipo de eventos deve ser melhorado para procurar reduzir ainda mais este problema. A utilização de diferentes meios de comunicação, como *Bluetooth* ou cabo também pode ser ponderada, considerando, antes da sua implementação, as dificuldades que estes meios podem trazer ao sistema.

Concluindo, dos três métodos de interação avaliados, a utilização do rato e teclado comprovou ser a mais eficiente para a tarefa proposta, enquanto o uso do acelerómetro demonstrou ser a mais ineficiente. No entanto, vale a pena salientar que o uso do toque apesar de menos eficiente que o rato e teclado foi preferido pelos utilizadores e considerado o mais fácil para completar a tarefa. Finalmente, apesar da *framework* permitir a realização dos testes de um modo satisfatório, ficou claro que existe espaço para algumas melhorias que devem ser efetuadas no futuro.

Capítulo 6

Conclusões e Trabalho Futuro

Neste capítulo apresenta-se um resumo da dissertação, assim como as conclusões retiradas e trabalho a desenvolver no futuro.

O relatório inicia-se pela introdução do contexto desta dissertação abordando temas desde limitações na interação pessoa-computador até à proliferação dos dispositivos móveis, levantando a hipótese da utilização dos dispositivos móveis para complementar a interação pessoa-computador. É também neste capítulo que são definidos os objetivos principais propostos para esta dissertação.

Após uma análise de trabalhos relacionados na área, desde métodos interação até alguns conceitos teóricos e tecnologias de desenvolvimento, é apresentada a solução proposta por esta dissertação.
A solução proposta neste trabalho passa pela integração dos dispositivos móveis como *smartphones* e *tablets* na interação pessoa-computador, sendo os principais objetivos a utilização do ecrã multi-toque e dos vários sensores disponíveis no dispositivo para interagir com uma aplicação num computador.

Assim, foi especificada a *Smart Device Computer Integration (SDCI)*, uma *framework* que procura facilitar a integração dos dispositivos móveis como complementos na interação com computadores. Foram apresentadas as funcionalidades principais disponibilizadas pela *SDCI*, como a manipulação da interface do dispositivo a partir de um servidor externo. Para além disso foi detalhada a arquitetura proposta pela *framework*, uma arquitetura com várias camadas e que se estende tanto ao dispositivo móvel como ao computador que implementa o servidor. Demonstrou-se ainda o funcionamento do protocolo de comunicação definido pela *SDCI*, foram especificadas as diferentes mensagens utilizadas, assim como a interação entre elas. Para complementar a especificação da *framework*, é também especificada uma estrutura *XML* a utilizar na definição de gestos

para possibilitar o seu reconhecimento.

De seguida foi descrita a implementação da *framework* proposta pela dissertação, desde a arquitetura implementada até à tecnologia utilizada. Para ser possível apresentar uma prova de conceito da *framework* especificada, foi desenvolvida uma aplicação móvel que fornece as funcionalidades da *framework* a qualquer servidor que implemente o protocolo e a estrutura definida pela mesma. Para validar o uso da *SDCI* foi especificada uma extensão para o *Blender* onde foi implementado um servidor capaz de interagir com a aplicação móvel desenvolvida.

Para além disso foi exemplificado o funcionamento desta extensão, que permitia ao dispositivo móvel uma serie de funcionalidades para interagir com o *Blender*, como a criação de objetos e operações geométricas.

Finalmente apresentou-se o método de avaliação utilizado, no qual foi proposto aos utilizadores a criação de um modelo no *Blender* utilizando três métodos de interação, rato e teclado, toque e acelerómetro. A partir desta avaliação foi possível verificar que o rato e teclado foram mais rápidos do que os restantes métodos, mas que os utilizadores demonstraram alguma afinidade pela utilização do toque como método de interação. Por outro lado, o uso do acelerómetro foi o menos preferido pelos utilizadores e também o que apresentou menor eficiência na realização da tarefa proposta.

Para terminar, foi possível encontrar respostas para as questões de investigação apresentadas no primeiro capítulo deste documento. A *framework* desenvolvida permite a utilização dos valores recolhidos pelos vários sensores para enriquecer a interação com uma aplicação no computador. É transmitida ao computador uma variedade de informação, permitindo a este utilizar e manipular os valores consoante a tarefa a realizar. Para além disso, foi possível encontrar soluções para a adaptabilidade da interface do dispositivo, fornecendo ao computador vários métodos de manipulação da interface, como definir e adicionar novos elementos a interfaces. Ambas estas funcionalidades são suportadas pelo protocolo de comunicação especificado pela *framework* que permite manter um canal de comunicação de dois sentidos entre os dispositivos, transmitindo informação desde os sensores até à manipulação da interface.

6.1 Trabalho Futuro

Como foi referido no capítulo 5, os utilizadores sugeriram algumas melhorias para a extensão desenvolvida para a integração com o *Blender*.

Algumas das melhorias sugeridas passam pela inclusão de algumas funcionalidades para melhorar a usabilidade, como o aumento da área de toque sobre os botões de radiais para a escolha

dos eixos a efetuar a operação. Foi também sugerido a adição de uma função que permitisse duplicar objetos e anular operações.

Sobre a *framework SDCI*, é também possível fazer algumas melhorias. A transmissão dos gestos a serem reconhecidos pela *framework* deve ser melhorada. Atualmente a definição dos gestos é transmitida simultaneamente com o ecrã que necessite deles, isto levou à existência de um *delay* quando este ecrã era carregado pela aplicação. Este processo pode ser melhorado se a definição dos gestos for efetuada imediatamente a seguir ao processo de ligação com o servidor, introduzindo mais uma fase no *setup* da aplicação.

O tratamento de mensagens do acelerómetro pode também ser melhorado. Uma vez que este sensor despoleta um elevado número de eventos quando está a ser utilizado, um tratamento melhor destes eventos pode ser implementado para reduzir o número de mensagens enviadas para o servidor, reduzindo assim os *delays* observados durante os testes efetuados com utilizadores.

Excluindo estas duas melhorias, o desenvolvimento da *SDCI* passa pela introdução de novas funcionalidades ao nível do controlo da interface e dos sensores. Podem ser incluídos novos sensores na implementação da *framework* como o sensor de luz ambiente, ou de proximidade, assim como a inclusão da câmara fotográfica.

O controlo da interface também pode ser melhorado acrescentando suporte a tipos de interface mais especial, como alertas. Neste momento a *framework* suporta a adição de um alerta com uma caixa de texto, onde pode ser configurado o titulo e subtítulo. A adição de outros tipos de parâmetros para estes alertas, como número de botões traz mais vantagens ao uso da *framework*.

Finalmente, recomenda-se a execução de mais testes com utilizadores, possivelmente utilizando outro tipo de aplicação, como jogos ou aplicações que possam beneficiar mais uma interação com o toque ou o acelerómetro.

Referências

[AS16] Jorik Tangelder Alexander Schmitz, Chris Thoburn. Hammer.js, acedido em junho 2016. URL: http://hammerjs.github.io/.

[Aug16] Aymeric Augustin. Python Websockets, acedido em junho 2016. URL: https://github.com/aaugustin/websockets.

[AW10] Lisa Anthony e Jacob O Wobbrock. A Lightweight Multistroke Recognizer for User Interface Prototypes. *Proceedings of Graphics Interface (GI '10). Ottawa, Ontario (May 31-June 2, 2010)*, pages 245–252, 2010.

[BBRS06] Rafael Ballagas, Jan Borchers, Michael Rohs e Jennifer G. Sheridan. The smart phone: A ubiquitous input device. *IEEE Pervasive Computing*, 5(1):70–71, 2006. doi:10.1109/MPRV.2006.18.

[BFAS15] Matthias Baldauf, Peter Fröhlich, Florence Adegeye e Stefan Suette. Investigating On-Screen Gamepad Designs for Smartphone-Controlled Video Games. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 12(1s):1–21, 2015. URL: http://dl.acm.org/citation.cfm?id=2837676.2808202, doi:10.1145/2808202.

[BGMF11] Xiaojun Bi, Tovi Grossman, Justin Matejka e George Fitzmaurice. Magic Desk : Bringing Multi-Touch Surfaces into Desktop Work. pages 2511–2520, 2011.

[BJJ98] Keith A Butler, Bonnie E John e Robert J K Jacob. Human-Computer Interaction : Introduction and Overview. *Computer*, (April):105–106, 1998.

[BTE11] Jens Bauer, Sebastian Thelen e Achim Ebert. Using Smart Phones for Large-Display Interaction. *Proceedings of the International Conference on User Science and Engineering (iUSEr'11)*, pages 42–47, 2011. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6150533, doi:10.1109/iUSEr.2011.6150533.

[Co16] Drifty Co. Ionic: Advanced html5 hybrid mobile app framework, acedido em junho 2016. URL: http://ionicframework.com/docs/.

[eI16] ecma International. JSON - Javascript Object Notation, acedido em junho 2016. URL: http://json.org/.

[Fou16] Blender Foundation. blender.org - home of the blender project, acedido em junho 2016. URL: https://www.blender.org/.

[FPT12] R Francese, I Passero e G Tortora. Wiimote and Kinect: Gestural User Interfaces add a Natural third dimension to HCI. *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 116–123, 2012. URL: <GotoISI>://WOS:000323214900020, doi:Doi 10.1145/2254556.2254580.

[Fuc16] Thomas Fuchs. Zepto.js: the aerogel-weight jquery compatible javascript library, acedido em junho 2016. URL: http://zeptojs.com/#.

[FWC84] James D. Foley, Victor L. Wallace e Peggy Chan. The human factors of computer graphics interaction techniques. *IEEE Computer Graphics and Applications*, 4(11):13–48, 1984. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6429355, doi:10.1109/MCG.1984.6429355.

[FWSB07] Clifton Forlines, Daniel Wigdor, Chia Shen e Ravin Balakrishnan. Direct-touch vs. mouse input for tabletop displays. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*, page 647, 2007. URL: http://portal.acm.org/citation.cfm?doid=1240624.1240726, doi:10.1145/1240624.1240726.

[Gui16] Android API Guides. Sensors overview android developers. http://developer.android.com/guide/topics/sensors/sensors_overview.html, acedido em fevereiro 2016.

[HHCC07] Uta Hinrichs, Mark Hancock, Sheelagh Carpendale e Christopher Collins. Examination of Text-Entry Methods for Tabletop Displays. *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07)*, pages 105–112, 2007. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4384118, doi:10.1109/TABLETOP.2007.10.

[HTB14] Steven Houben, Paolo Tell e Jakob E. Bardram. ActivitySpace. *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces - ITS '14*, (November 2015):119–128, 2014. URL: http://dl.acm.org/citation.cfm?doid=2669485.2669493, doi:10.1145/2669485.2669493.

[IF16] Alexey Melnikov Ian Fette. Rfc 6455 - the websocket protocol, acedido em junho 2016. URL: https://tools.ietf.org/html/rfc6455.

[JHKB06] Seokhee Jeon, Jane Hwang, Gerard J. Kim e Mark Billinghurst. Interaction techniques in large display environments using hand-held devices. *Proceedings of the ACM symposium on Virtual reality software and technology - VRST '06*, page 100, 2006. doi:10.1145/1180495.1180516.

[KBBC05] Martin Kaltenbrunner, Till Bovermann, Ross Bencina e Enrico Costanza. TUIO: A protocol for table-top tangible user interfaces. *Neuroinformatics*, pages 1–5, 2005. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.155.3890{&}amp;rep=rep1{&}amp;type=pdf.

[KKY+15] Hyun-suh Kim, Howard Kim, Inhwan Yoon, Chul-ho Jung e Yong Hwan. User Performance Differences between Graphics Tablet and Mouse in Graphic Applications : Focus on Controllability and Accuracy. 10(9):167–174, 2015.

REFERÊNCIAS

[LA16] Jacob O. Wobbrock Lisa Anthony. $N Multistroke Recognizer, acedido em junho 2016. URL: https://depts.washington.edu/aimgroup/proj/dollar/ndollar.html.

[Lew95] James R Lewis. IBM Computer Usability Satisfaction Questionnaires : Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction, 7(1)*, (1):57–78, 1995.

[Liu10] Weiyuan Liu. Natural user interface - Next mainstream product user interface. *2010 IEEE 11th International Conference on Computer-Aided Industrial Design and Conceptual Design, CAID and CD'2010*, 1:203–205, 2010. doi:10.1109/CAIDCD.2010.5681374.

[LV10] Yang Li e Mountain View. Protractor : A Fast and Accurate Gesture Recognizer. *Proceeding CHI '10 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2169–2172, 2010.

[MN13] Fabrice Matulic e Moira C Norrie. Pen and Touch Gestural Environment for Document Editing on Interactive Table top s. *Its'13*, pages 41–50, 2013. doi:10.1145/2512349.2512802.

[SAL14] ALEXEY SALNIKOV. Natural Interface to Improve Human-Computer Interaction for People with Upper Limb Disabilities. 2014.

[Set16] Jairaj Sethi. Touchy.js, acedido em junho 2016. URL: https://github.com/jairajs89/Touchy.js.

[Tec16] Unity Technologies. Unity - game engine, acedido em junho 2016. URL: https://unity3d.com/pt/unity.

[uwd16] uwdata. Gestrec - a javascript implementation of the protractor gesture recognizer, acedido em junho 2016. URL: https://github.com/uwdata/gestrec.

[VGT+10] Pedro G Villanueva, José A JA Gallud, Ricardo Tesoriero, P. Gonzalez-Villanueva e S Romero. WallShare: A Collaborative Multi-pointer System for Portable Devices. *Info-Ab.Uclm.Es*, page 416, 2010. URL: http://www.info-ab.uclm.es/descargas/thecnicalreports/DIAB-09-10-1/wallshare-v1.pdf$\delimiter"026E30F$nhttp://doi.acm.org/10.1145/1842993.1843094, doi:10.1145/1842993.1843094.

[Zab11] Stanislaw Zabramski. Careless touch: A comparative evaluation of mouse, pen, and touch input in shape tracing task. *Proceedings of the 23rd Australian Computer-Human Interaction Conference on - OzCHI '11*, pages 329–332, 2011. URL: http://dl.acm.org/citation.cfm?doid=2071536.2071588, doi:10.1145/2071536.2071588.

[ZSS97] Shumin Zhai, Barton A Smith e Ted Selker. Improving Browsing Performance : A study of four input devices for scrolling and pointing tasks. *Interact97*, 17:286–292, 1997.

REFERÊNCIAS

Capítulo 7

Anexo

Este capítulo apresenta alguns anexos relevantes para uma melhor compreensão da dissertação.

7.1 Guiões Aprendizagem

7.1.1 Guião Blender

Tarefa informal, para o utilizador treinar a utilização da aplicação e diferentes modos de interação Máximo 10 minutos para testar a aplicação

Tarefa - Desenhar 3 objetos diferentes e aplicar diferentes transformações

Controlo da janela do Blender

1. Mover o rato enquanto mantem premida a roda para ajustar a janela de visualização
2. Utilizar os botões do teclado numérico para ajustar a janela de visualização

Seleção de objetos

1. Clicar sobre o objeto com o botão direito do rato

Criar objeto

1. Selecionar o painel “Create” (lado esquerdo da aplicação)
2. Seleccionar o objeto a desenhar

Escalar objecto

1. Selecionar o painel “Tools” (lado esquerdo da aplicação)
2. Clicar no botão “Scale” para entrar no modo de escala
3. Selecionar eixo a escalar clicando no botão X/Y/Z

4. Clicar no botão esquerdo do rato para finalizar

Mover objeto

1. Selecionar o painel “Tools” (lado esquerdo da aplicação)
2. Clicar no botão “Translate” para entrar no modo de escala
3. Selecionar eixo a mover clicando no botão X/Y/Z
4. Clicar no botão esquerdo do rato para finalizar

Rodar objeto

1. Selecionar o painel “Tools” (lado esquerdo da aplicação)
2. Clicar no botão “Rotate” para entrar no modo de escala
3. Selecionar eixo a rodar clicando no botão X/Y/Z
4. Clicar no botão esquerdo do rato para finalizar

7.1.2 Guião Aplicação

Tarefa informal, para o utilizador treinar a utilização da aplicação e diferentes modos de interação Máximo 10 minutos para testar a aplicação

Tarefa - Desenhar 3 objetos diferentes e aplicar diferentes transformações

Controlo da janela do Blender

1. Mover o rato enquanto mantem premida a roda para ajustar a janela de visualização
2. Utilizar os botões do teclado numérico para ajustar a janela de visualização

Seleção de objetos

1. Clicar sobre o objeto com o botão direito do rato ou
2. Utilizar os botões “next object” ou “previous object” no ecrã de transformações

Desenhar objeto

1. Selecionar o botão “Draw operations”
2. Selecionar o botão “Gestures”
3. Desenhar um quadrado/cilindro/cone/torus/smiley (aqui mostrar as imagens possiveis)
4. Clicar sobre o desenho com dois dedos para criar um objecto no blender

Escalar objecto

1. Selecionar o botão “Transform operations”
2. Clicar no botão “Scale” para entrar no modo de escala
3. Selecionar eixo a escalar clicando no botão correspondente
4. Clicar no botão “Start using touch” ou “Start using accelerometer”
5. Se utilizar toque:
    (a) Aplicar sobre o rectangulo azul um gesto de pinch com dois dedos para escalar o objeto
6. Se utilizar o acelerometro:
    (a) Rodar ligeiramente o telemóvel para a esquerda/direita para escalar o objeto

Mover objeto

1. Selecionar o botão “Transform operations”
2. Clicar no botão “Move” para entrar no modo de mover
3. Selecionar eixo a mover clicando no botão correspondente
4. Clicar no botão “Start using touch” ou “Start using accelerometer”
5. Se utilizar toque:
    (a) Aplicar sobre o rectangulo azul um slide para a esquerda/direita para mover o objeto
6. Se utilizar o acelerómetro:
    (a) Rodar ligeiramente o telemóvel para a esquerda/direita para mover o objeto

Rodar objeto

1. Selecionar o botão “Transform operations”
2. Clicar no botão “Rotate” para entrar no modo de rotação
3. Selecionar eixo a mover clicando no botão correspondente
4. Clicar no botão “Start using touch” ou “Start using accelerometer”
5. Se utilizar toque:
    (a) Aplicar sobre o rectangulo azul um gesto de rotação com dois dedos para rodar o objeto
6. Se utilizar o acelerómetro:
    (a) Rodar ligeiramente o telemóvel para a esquerda/direita para rodar o objeto

7.2 Guião Principal

Tarefa a avaliar com 3 métodos de interação - Blender, Aplicação utilizando toque e Aplicação utilizando acelerómetro para as transformações geométricas.

Após a conclusão da tarefa utilizando um método de interação o utilizador terá uma pausa de 2 minutos para relaxar e responder a duas perguntas sobre a tarefa efetuada.

Durante os testes serão recolhidas algumas métricas como tempo decorrido e número de erros efetuados. Estes dados serão utilizados para tentar determinar a viabilidade deste método de interação face aos métodos atuais.

É também pedido que os utilizadores façam comentários durante a realização das tarefas sobre a experiencia de usabilidade dos métodos utilizados. Estes comentários serão registados e analisados posteriormente para uma melhor avaliação da plataforma.

Tarefa - Desenhar boneco como na figura apresentada

Boneco

1. Criar tronco do boneco
   (a) Desenhar cubo
   (b) Mover/Escalar/Rodar objeto
2. Criar perna esquerda do boneco
   (a) Desenhar cilindro
   (b) Mover/Escalar/Rodar objeto
3. Criar perna direita do boneco
   (a) Desenhar cilindro
   (b) Mover/Escalar/Rodar objeto
4. Criar braço esquerdo do boneco
   (a) Desenhar cilindro
   (b) Mover/Escalar/Rodar objeto
5. Criar braço direito do boneco
   (a) Desenhar cilindro

(b) Mover/Escalar/Rodar objeto

6. Criar cabeça do boneco

   (a) Desenhar cara do macaco

   (b) Mover/Escalar/Rodar objeto

7.3 Questionário

Questionário avaliação usabilidade

*Obrigatório

1. **Idade** *
   Marcar apenas uma oval.
   - ◯ < 18
   - ◯ 18 < 25
   - ◯ 25 < 35
   - ◯ 35 < 45
   - ◯ > 45

2. **Género**
   Marcar apenas uma oval.
   - ◯ Masculino
   - ◯ Feminino

3. **Experiencia com ferramentas de modelação**
   Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Pouco experiente	◯	◯	◯	◯	◯	◯	◯	Muito Experiente

Método de interação 1

4. **Método de interação avaliado** *
   Marcar apenas uma oval.
   - ◯ Blender
   - ◯ Aplicação utilizando toque
   - ◯ Aplicação utilizando acelerómetro

5. **Estou satisfeito(a) com a facilidade com que completei a tarefa?** *
   Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Discordo completamente	◯	◯	◯	◯	◯	◯	◯	Concordo completamente

6. **Estou satisfeito(a) com o tempo que demorei a completar a tarefa?** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Discordo completamente	◯	◯	◯	◯	◯	◯	◯	Concordo completamente

Método de interação 2

7. **Método de interação avaliado** *
Marcar apenas uma oval.

- ◯ Blender
- ◯ Aplicação utilizando toque
- ◯ Aplicação utilizando acelerómetro

8. **Estou satisfeito(a) com a facilidade com que completei a tarefa?** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Discordo completamente	◯	◯	◯	◯	◯	◯	◯	Concordo completamente

9. **Estou satisfeito(a) com o tempo que demorei a completar a tarefa?** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Discordo completamente	◯	◯	◯	◯	◯	◯	◯	Concordo completamente

Método de interação 3

10. **Método de interação avaliado** *
Marcar apenas uma oval.

- ◯ Blender
- ◯ Aplicação utilizando toque
- ◯ Aplicação utilizando acelerómetro

11. **Estou satisfeito(a) com a facilidade com que completei a tarefa?** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Discordo completamente	◯	◯	◯	◯	◯	◯	◯	Concordo completamente

12. **Estou satisfeito(a) com o tempo que demorei a completar a tarefa?** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Discordo completamente	◯	◯	◯	◯	◯	◯	◯	Concordo completamente

Final

13. **Qual foi o método de interação preferido?** *
Marcar apenas uma oval.

- ◯ Blender
- ◯ Aplicação utilizando toque
- ◯ Aplicação utilizando acelerómetro