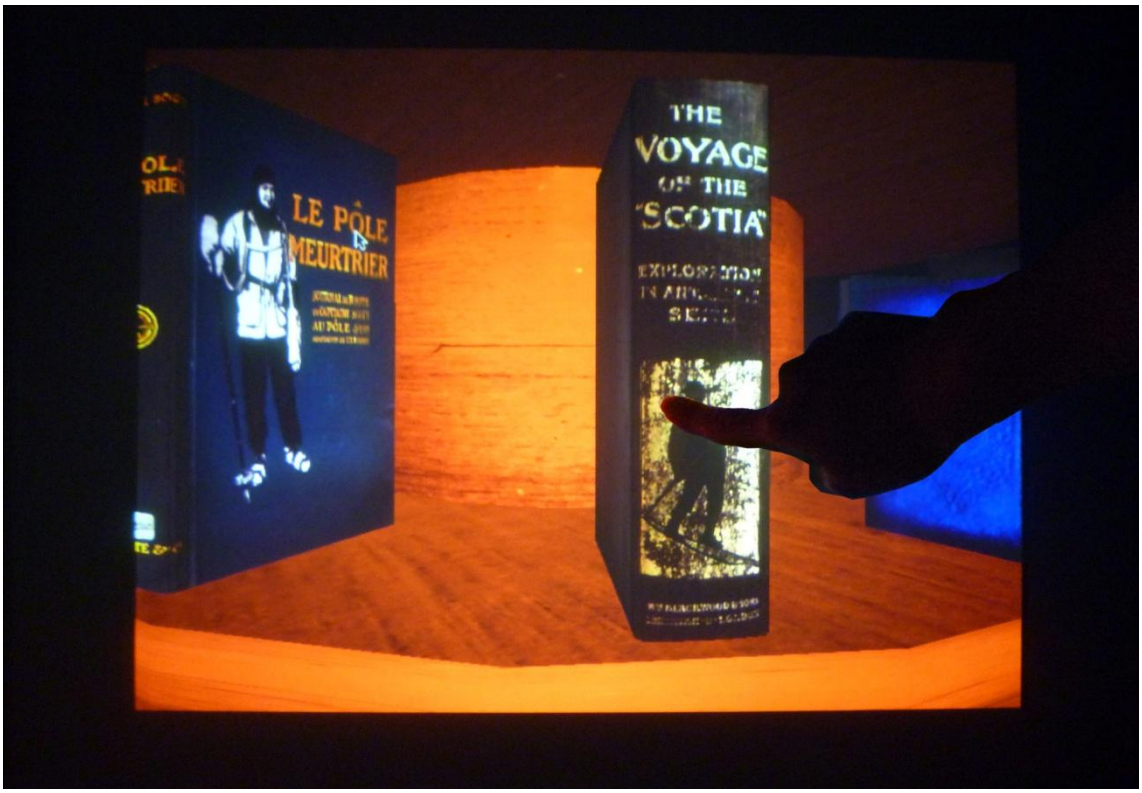


ESTANTE DIGITAL

Relatório do Trabalho Final de curso

ANA MARIA MOUTINHO

ESTANTE DIGITAL

Relatório do Trabalho Final de curso

Relatório do Trabalho Final de Curso da Licenciatura em Engenharia Informática, da Universidade Lusófona de Humanidades e Tecnologias.

Orientador: Professor Auxiliar Sérgio Guerreiro

Universidade Lusófona de Humanidades e Tecnologias

Escola de Comunicação, Artes e Tecnologias da Informação

Lisboa

2011

Resumo

O presente projecto final de curso consiste no desenvolvimento de um protótipo que implementa algumas funcionalidades e consiste num módulo do projecto "Estante Digital". Este protótipo é composto por uma superfície multitoque e uma aplicação que permite o utilizador interagir visualmente com objectos 3D através do toque. A aplicação permite que um ou mais utilizadores possam interagir em simultâneo.

Palavras-chave: Estante Digital; Multitoque; Arquitectura Interactiva; TUIO; Unity 3

Abstract

This final project pretends to develop a prototype that implements some functionalities of the project "Digital Shelf". This prototype consists of a multi-touch surface and an application that allows the user to interact visually with 3D objects by touch. The application allows one or more users to interact simultaneously.

Key-words: Digital Shelf; Multitouch; Interactive Architecture; TUIO; Unity 3

Abreviaturas e Símbolos

DI	Diffused Illumination
DSI	Diffused Surface Illumination
FTIR	Frustrated Total Internal Reflection
GUI	Graphical User Interface
LED-LP	LED Light Plane
LLP	Laser Light Plane
NUI-Group	Natural User Interface Group
TUI	Tangible User Interface
TUIO	Tangible User Interface Objects
CCV	Community Core Vision
FPS	*Frames* per second

Índice

Índice de Figuras

Introdução

O presente trabalho final de curso consiste no desenvolvimento de uma aplicação que operacionaliza a representação à escala 1:1, ao nível da visualização/reprodução, exibição e manipulação de produtos digitais (ex: livros e revistas), respeitando os comportamentos naturais de percepção visual, de memória e de aprendizagem.

A solução encontrada corresponde a uma alteração de paradigma no relacionamento entre os utilizadores e os produtos digitais, isto é, através da transposição do actual GUI – *Graphical User Interface* para o próprio ambiente construído[1], TUI - *Tangible User Interface.*

O protótipo representa um módulo, parte constituinte de uma estante digital composta por vários módulos incorporados na estrutura do ambiente construído e à escala do mesmo.

O protótipo é apresentado numa superfície multitoque, construída com base na técnica LLP, *Laser Light Plane.* A aplicação é desenvolvida em C#, na ferramenta de desenvolvimento de jogos Unity 3, com recurso a objectos 3D construídos em 3D Studio Max e ao protocolo TUIO, para comunicação com a superfície multitoque, enviado através do interface xTouch.

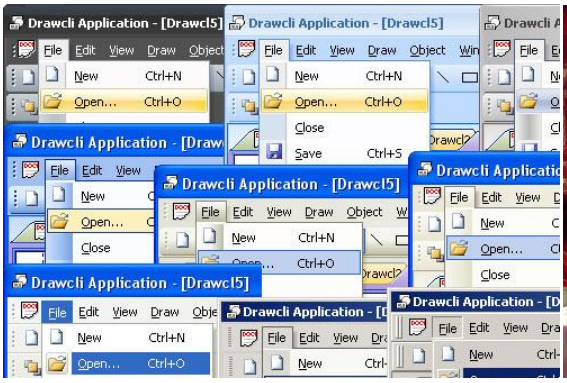
Este projecto permite explorar vários domínios de conhecimento, como a electrónica, design, programação e 3D, em que todos comunicam entre si.

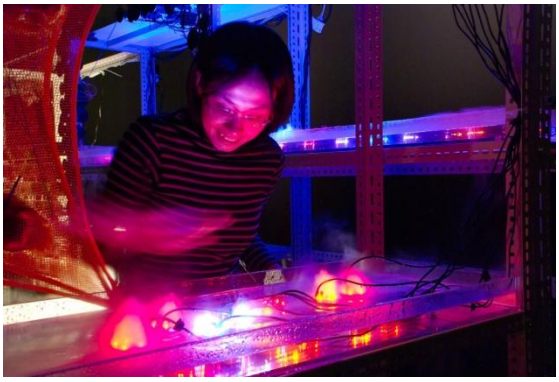
[1] O ambiente tangível passa a ser o interface de informação digital.

1. Enquadramento

1.1. Graphical User Interface / Tangible User Interface

Há menos de uma década, os interfaces de computação ignoravam as manifestações circundantes e concentravam grandes actividades de informação numa *janela* (ver figura 1). O convencional GUI *Graphical User Interface* é um tipo de interface que permite ao utilizador, interagir com dispositivos digitais através de elementos gráficos como ícones e outros indicadores visuais. A interacção é feita geralmente através de um rato ou um teclado, com os quais o utilizador é capaz de seleccionar símbolos e manipulá-los de forma a obter algum resultado prático. Actualmente o GUI continua a ser muito utilizado, no entanto, o presente trabalho centra a sua investigação noutro tipo de interface o TUI *Tangible User Interface*[2], que transforma o próprio ambiente construído no interface de informação digital e a interacção é feita através do multitoque (Ulmer & Eishii, 2000), (ver figura 2).





Figuras 1 e 2 – GUI (à esq.) e TUI (à dir.).

De forma a que seja possível implementar na prática o TUI, é necessário um diálogo com os componentes arquitectónicos, de forma a que toda a infra-estrutura electrónica esteja incorporada na própria estrutura arquitectónica e invisível ao utilizador - Computação Pervasiva (Saha & Mukherjee, 2003)[3]. Este diálogo entre a computação e a arquitectura é comummente designado por Arquitectura Interactiva.

[2] Os TUIs permitem aumentar o mundo físico através da integração da informação digital nos objectos físicos do quotidiano e nos ambientes envolventes do ser-humano.

[3] A Computação Pervasiva, termo utilizado pela IBM, define que os meios de computação estão distribuidos nos diferentes ambientes do utilizador, consiste numa tecnologia distribuida, que está sempre presente mas não é intrusiva.

1.2. Arquitectura Interactiva

A Arquitectura Interactiva não é simplesmente reactiva ou adaptável a mudanças circunstanciais. Pelo contrário, é baseada no conceito de comunicação bidireccional a qual requer duas partes activas. Naturalmente a comunicação entre duas pessoas é interactiva, ambos ouvem (*input*), pensam (*process*) e falam (*output*). A Arquitectura Interactiva é principalmente a comunicação entre os diferentes componentes de um edifício e a comunicação entre pessoas e os componentes do mesmo edifício (Bullivant, 2007: 49).

Figura 3: *Prototype For An Emotive Wall*, 2009, Hyperbody, http://www.bk.tudelft.nl/live/pagina.jsp?id=bff05884-fb66-4585-b270-0b9c4f079917&lang=en.

Prototype For An Emotive Wall é um projecto que pode ser inserido na Arquitectura Interactiva, a figura 3 ilustra um exemplo importante na mutação e transformação dos espaços, desenvolvido pelo Hyperbody[4] e apresentado pela Festo[5] na Hannover Messe[6] de 20 a 24 de Abril de 2009. *Prototype For An Emotive Wall* é composto por 7 módulos que em tempo real movimentam-se para a frente e para trás exibindo diferentes padrões lumínicos e som localizado. A superfície da parede é coberta por um camada de LEDs que reage à presença do participante.

[4] Hyperbody é um grupo de investigação da Faculdade de Arquitectura da Universidade Técnica de Delft (Technische Universiteit Delft) dirigido pelo Professor Kas Oosterhuis e tem por objectivo explorar novas técnicas e métodos para o design e construção de Arquitectura Interactiva, virtual e não-standardizada.

[5] Festo é uma empresa com sede em Esslinger na Alemanha que produz actuadores pneumáticos e eléctricos para Automação Industrial. O site oficial é http://www.festo.com/.

[6] Hannover Messe é a maior feira mundial em Tecnologia Industrial orientada para os campos da mobilidade e automação. Esta feira apresenta inovações, tecnologias e materiais do mundo da indústria. Site oficial: http://www.hannovermesse.de.

Kas Oosterhuis[7], o coordenador do grupo Hiperbody afirma "An emotive wall is a wall that responds to the user, a wall that has a character, a wall that can move because it wants to." (Oosterhuis, 2009).

1.3. Tecnologias Multitoque e Linguagem Gestual

O multitoque[8], é uma tecnologia relativamente recente (década de 1980) e que cada vez mais é possível encontrar no nosso quotidiano, nos telemóveis, na televisão (meteorologia, telejornal), visualização de mapas, em bares (nas superfícies das mesas) ou em museus para apresentar conteúdos de informação interactivos. É uma tecnologia que em geral tem sido bem recebida pelo público e que cria algum tipo de curiosidade. Um dispositivo com a tecnologia Multitoque rapidamente pode tornar-se num objecto de entretenimento.

Um ecrã Multitoque permite desenhar, arrastar, aumentar ou diminuir objectos, por um utilizador ou vários em simultâneo. A figura 4 exemplifica alguns gestos que permitem a utilização dos dedos, que em conjugação com determinados movimentos representam uma acção específica. Da mesma forma que existem várias técnicas para construir um ecrã Multitoque, existe também inúmeras aplicações e *softwares* a serem desenvolvidos que suportam o Multitoque. Algumas técnicas para construir uma superfície Multitoque são: Frustrated Total Internal Reflection [FTIR]; Diffused Illumination [DI]; Laser Light Plane [LLP]; Diffused Surface Illumination [DSI] ou LED Light Plane [LED-LP], que necessitam de uma câmara e de um projector. Uma vez que estas técnicas recorrem à retroprojecção, existe o inconveniente de ter que existir alguma profundidade na parede. Contudo a relação entre a dimensão da projecção e a distância do projector à parede (ecrã) pode ser controlada através de espelhos ou de projectores que ampliam a projecção com curta distância.

[7] Kas Oosterhuis, Arquitecto, Investigador e Professor na Faculdade de Arquitectura na Universidade Técnica de Delft, é Coordenador do grupo de Investigação Hyperbody e do *Protospace Laboratory*. As suas áreas de investigação são a Arquitectura Interactiva, o comportamento em tempo real dos edifícios, Design Colaborativo e Design Paramétrico.

[8] O Multitoque corresponde a um sistema de interacção Homem-máquina que é composto por uma superfície/ecrã táctil que reconhece múltiplos contactos em simultâneo que são interpretados pelo sistema. Desta forma permite a vários utilizadores interagirem com o mesmo computador.

Apesar de algumas plataformas Multitoque já estejam a ser comercializadas como o iPhone[9], ipad[10], kindle[11] ou a Microsoft Surface[12], ainda não existe uma linguagem gestual universal. O exemplo seguinte desenvolvido por Ryan Lee[13] é uma tentativa de elaborar a base dessa linguagem.

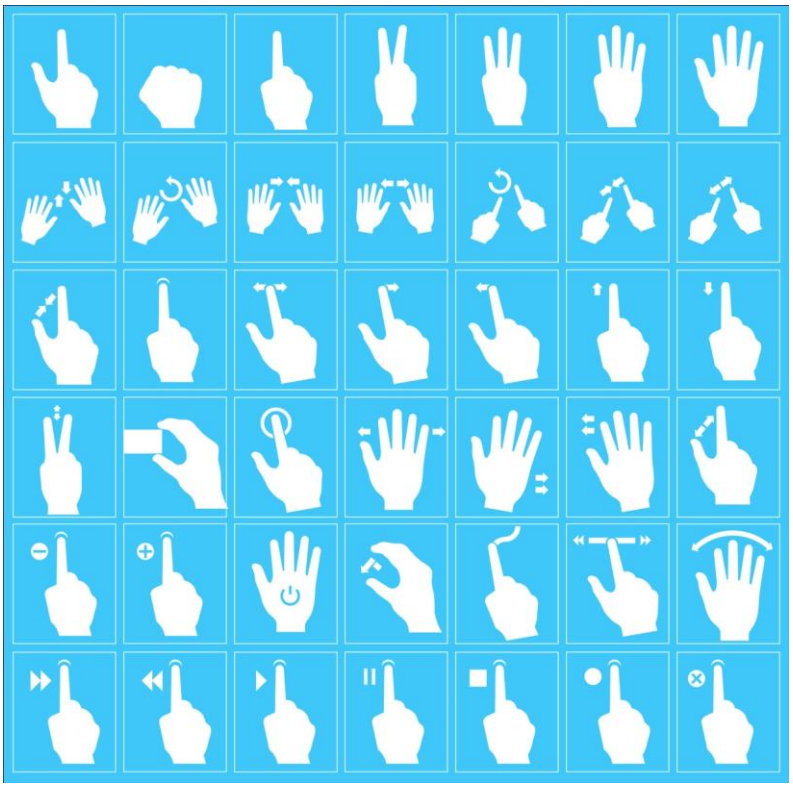


Figura 4- Exemplos de Linguagem Gestual - *Gesturecons*, 2010, *wire-framing* de ícones gestuais desenvolvido por Ryan Lee, http://gesturecons.com/.

Em paralelo com as grandes empresas que desenvolvem *hardware* e *software* com sistema multitoque, existem algumas comunidades de investigação *online*, como NUI Group – *Natural User Interface Group*[14], que é uma comunidade centrada no

[9] iPhone é um *smartphone* desenvolvido pela Apple Inc. Acedido a 25 de Junho de 2011 em http://www.apple.com/iphone

[10] iPad é um dispositivo em formato *tablet* produzido pela Apple Inc. Acedido a 25 de Junho de 2011 em http://www.apple.com/ipad

[11] Kindle é aparelho criado pela empresa americana Amazon, que tem como função principal ler livros electrónicos (e-book). Acedido a 25 de Junho de 2011 em http://www.amazon.com/gp/product/B002Y27P3M/ref=r_kdia_h_i_gl

[12] Microsoft Surface é uma mesa Multitoque desenvolvida pela Microsoft Corporation. Acedido a 25 de Junho de 2011 em http://www.microsoft.com/surface

[13] http://gesturecons.com/.

[14] http://nuigroup.com

desenvolvimento de "natural user interface" e que disponibilizam inúmeras aplicações com código aberto.

Outra comunidade *online* que tem desenvolvido investigação especificamente entre o multitoque e o 3D é xTuio[15], comunidade esta, que foi bastante útil para o desenvolvimento deste projecto.

1.4. Apresentação do Projecto

O presente trabalho refere-se à Computação Pervasiva e à Arquitectura Interactiva, no sentido em que o interface gráfico e de interacção são as próprias paredes do ambiente construído e os elementos gráficos representados são reproduções 3D à escala 1:1, dos correspondentes objectos físicos e a interacção decorre através de um sistema multitoque.

Consiste numa estante digital multitoque configurada para a exposição, visualização e manipulação de "artigos" 3D na escala 1:1 e tem por objectivo facilitar a relação entre o utilizador e o ambiente digital, respeitando comportamentos de aprendizagem naturais, de percepção visual e de construção da memória, através do recurso à escala 1:1 (ver figura 5).

O utilizador interage com o display através de linguagem gestual que é identificada por um sistema multitoque. Uma vez o artigo seleccionado, este destaca-se na estante podendo ser rodado nos vários eixos, aberto para aceder ao conteúdo, folheado folha a folha, ou folheado rapidamente. Qualquer artigo pode ser enviado para um periférico como um *e-books* ou uma impressora. O utilizador também pode executar uma pesquisa simples ou complexa à base de dados do dispositivo ou *online* e adquirir mais artigos compatíveis, através de *e-commerce*.

A presente invenção possibilita a exibição e manipulação de artigos em vários contextos de aplicação como a biblioteconomia, coleccionismo ou a museografia, para exibição de colecções onde a escala natural é significativamente portadora de informação tais como colecções de numismática, entomologia, etnografia e arte.

[15] http://xtuio.com

O presente trabalho final de curso consiste na implementação prática do projecto apresentado anteriormente, mas ao nível de um protótipo em que nem todas as funcionalidades estão implementadas.

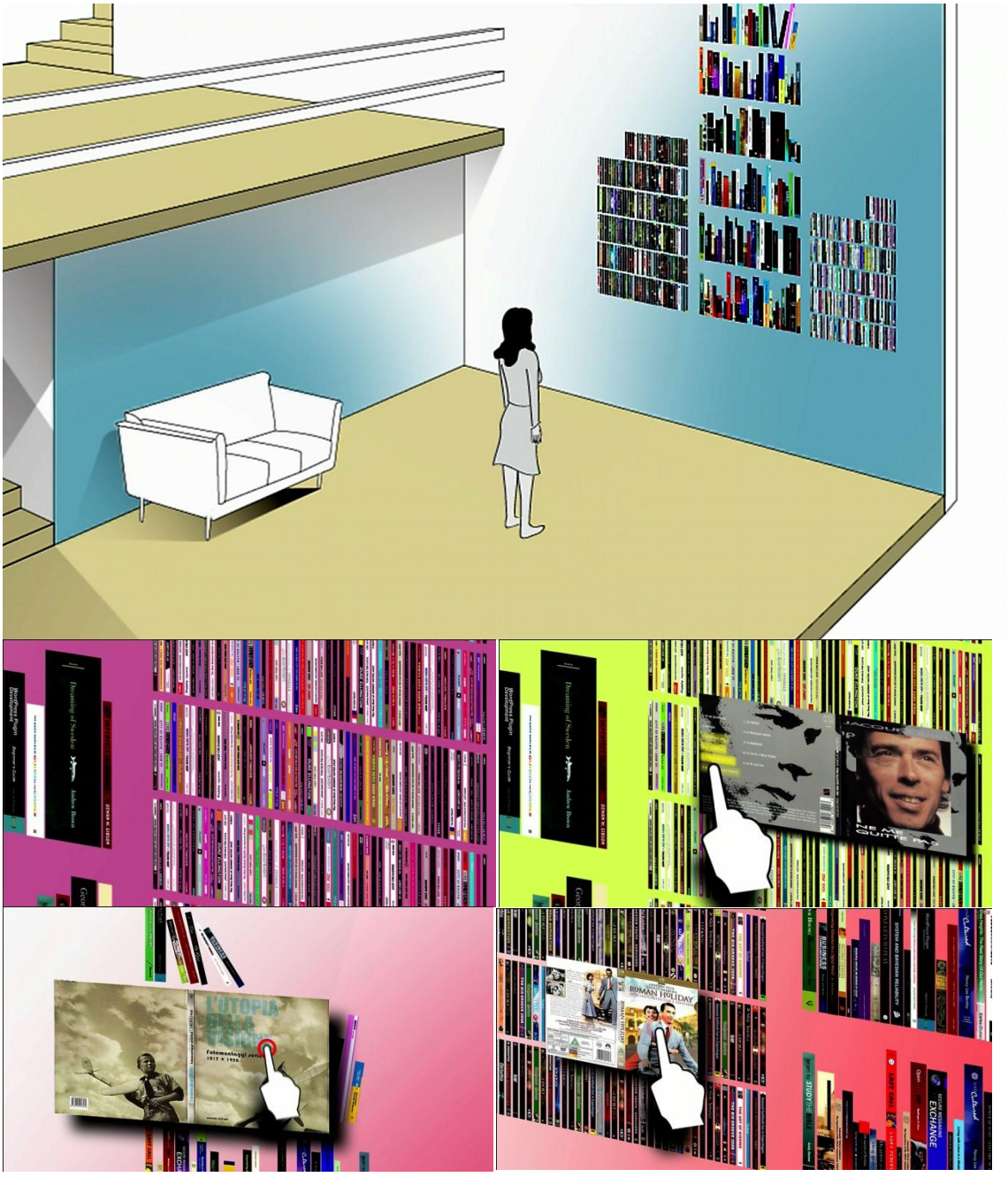


Figura 5 – Ilustração do Projecto *Estante Digital*

2. Método – Desenvolvimento do Protótipo

O protótipo desenvolvido denomina-se “Estante Digital”, é desenvolvido em C# na ferramenta de desenvolvimento de jogos Unity 3, com recurso a objectos 3D construídos na ferramenta de modelação 3D Studio Max, as texturas são feitas em Photoshop e o protótipo é apresentado numa superfície multitoque construída com a técnica LLP *Laser Light Plane*. O protocolo de comunicação entre a aplicação e a mesa é o TUIO, e é necessário recorrer ao interface xTouch, para estabelecer a comunicação.

Como referido na apresentação do projecto, este protótipo é um elemento que faz parte de um conjunto, como ilustrado na figura 6, o protótipo corresponde à área azul da figura, mas representa algumas das funções que o resto da superfície executaria.

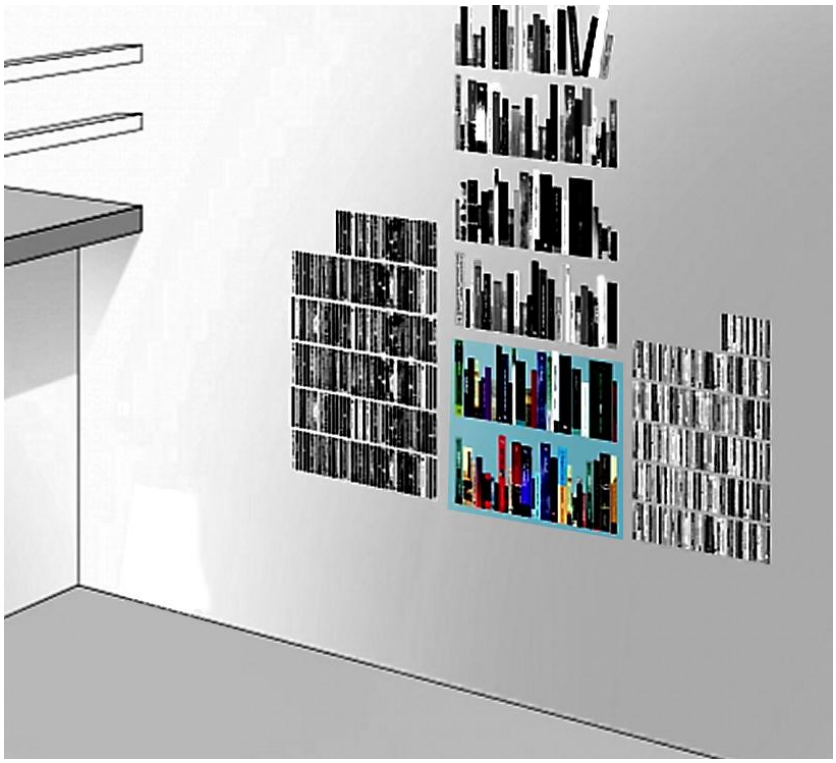


Figura 6 - Módulo da Estante Digital

2.1. Escolha de *Software*

Numa primeira parte da elaboração do protótipo foi realizada uma análise sobre o melhor *software* a utilizar para a sua execução, entre os vários *softwares* foram analisados os seguintes: adobe flash (as3), blender(pyton), processing (processing) e OpenGL, no entanto a escolha recaiu sobre o Unity3, pelo facto de ser uma ferramenta bastante robusta, com elevada performance gráfica, com uma comunicação segura com o protocolo TUIO e a possibilidade de programar em duas linguagens, o Java e o C#, ambas linguagens de programação exploradas no contexto prático de algumas disciplinas ao longo da presente licenciatura.

Uma vez seleccionado o ambiente de desenvolvimento, a execução do protótipo envolveu duas fases de desenvolvimento, nomeadamente o desenvolvimento da aplicação e a construção da superfície multitoque.

Lista dos Programas Utilizados:

- Unity 3
- 3D Studio Max
- TUIO Simulator
- CCV
- xTouch
- Photoshop

2.2. Processo de desenvolvimento da Aplicação

A Estante Digital representa um excerto de uma colecção privada de diários de viagem ao Pólo Norte e ao Pólo Sul, com publicações do início e metade do século XIX (ver figura 7). O processo de desenvolvimento da aplicação Estante Digital teve início com a construção de réplicas em 3D, com as dimensões (altura, espessura e largura) correspondente aos livros em suporte físico.

Figura 7 - Excerto da Colecção de diários de Viagem ao Pólo Norte e Pólo Sul

Os objectos em 3D foram construídos com recurso à ferramenta de modelação 3D Studio Max e exportados no formato (.3DS), (ver figura 8).

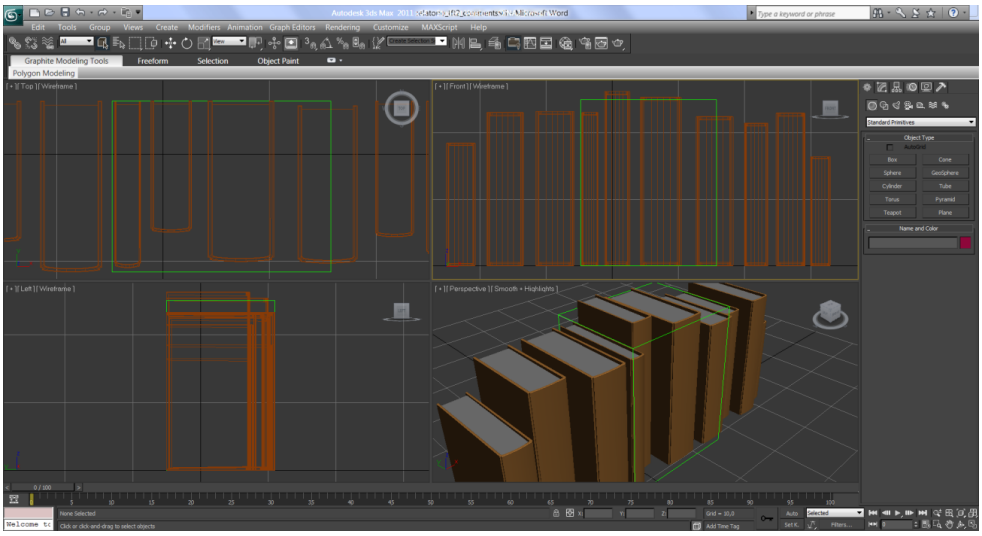


Figura 8 - Construção dos livros em 3D no programa 3D Studio Max.

As texturas aplicadas nos objectos tiveram por base fotografias dos livros nos vários ângulos que foram posteriormente agrupadas no programa Photoshop CS5, de forma a ter uma imagem planificada de cada livro (ver figura 9). Cada textura teve que ser "trabalhada" e editada em Photoshop, para que passasse despercebido qualquer alteração lumínica das fotografias e para que fizessem uma união mais fiel (ver figura 10).

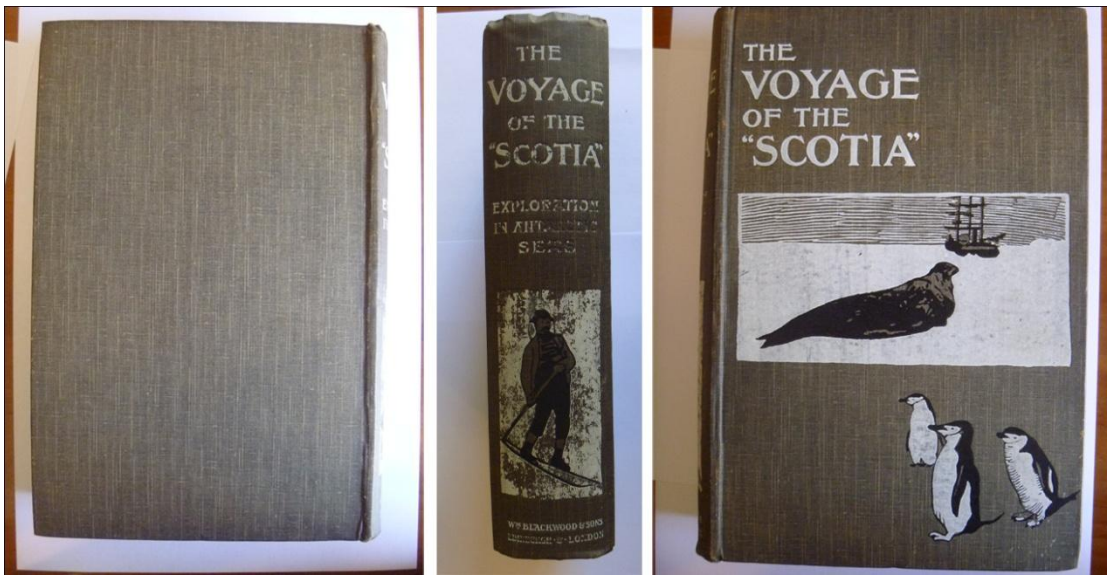


Figura 9. Fotografias com as vistas frontal, lateral e traseira do livro.

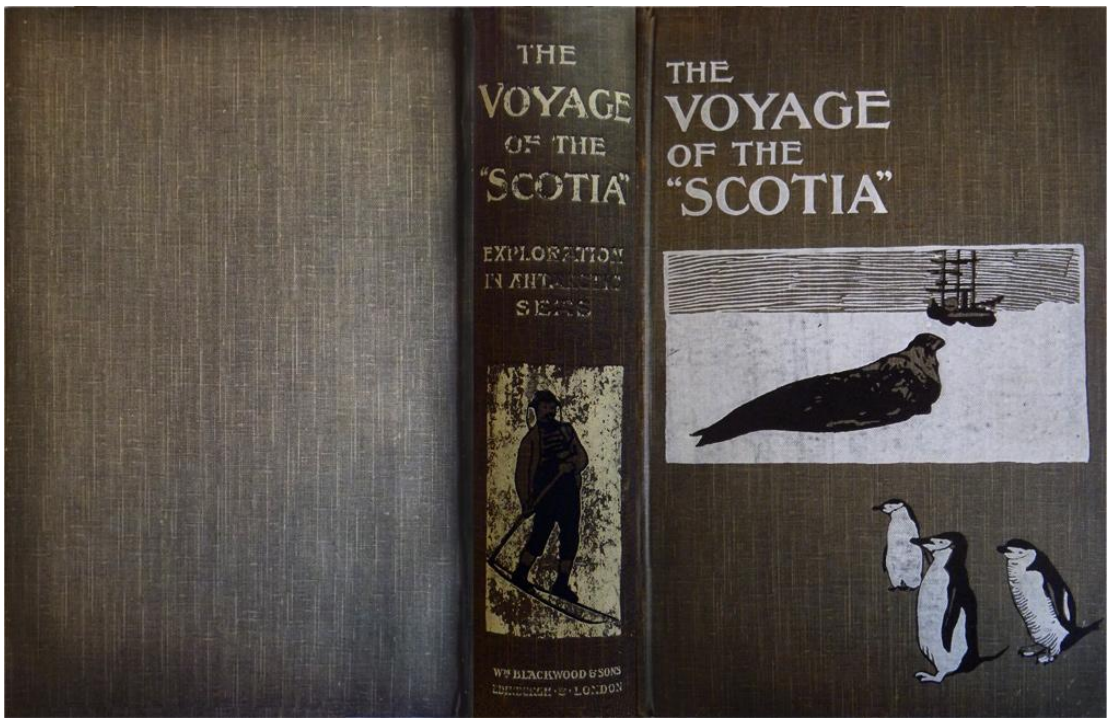


Figura 10. Livro Planificado em Photoshop CS5, com dimensões 5000 x 3224 píxeis, 330px de resolução, no formato (.psd).

Os objectos em 3D foram importados para a ferramenta de desenvolvimento de jogos Unity 3, tal como as texturas no formato (.psd)[16]. A compatibilidade entre o Unity 3 e os programas da Autodesk e da Adobe, permite um grande dinamismo e rapidez de desenvolvimento do trabalho, porque tanto o (.psd) como o (.3ds) são formatos aceites e a qualquer momento é possível fazer alterações, que são imediatamente actualizadas.

[16] (.psd) é o formato padrão da Adobe para documentos do Photoshop.

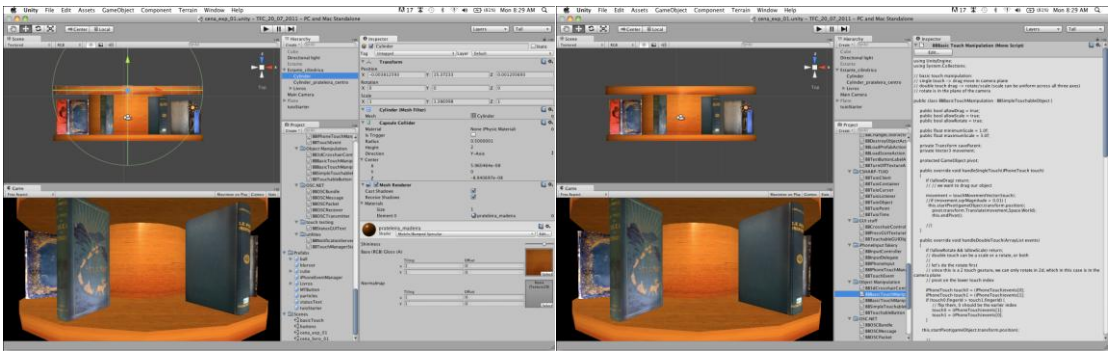


Figura 11 e 12 - Interface do Unity 3 e as propriedades disponíveis no inspector (à dir.) de um objecto cilíndrico e Interface do Unity 3 e um script em C# (à esq.).

O cenário e a apresentação da Estante teve algumas fases de evolução do grafismo, passando também por uma tentativa de apresentação de numismática, como pode ser observado na figura 14. A figura 13, representa o grafismo final da aplicação.

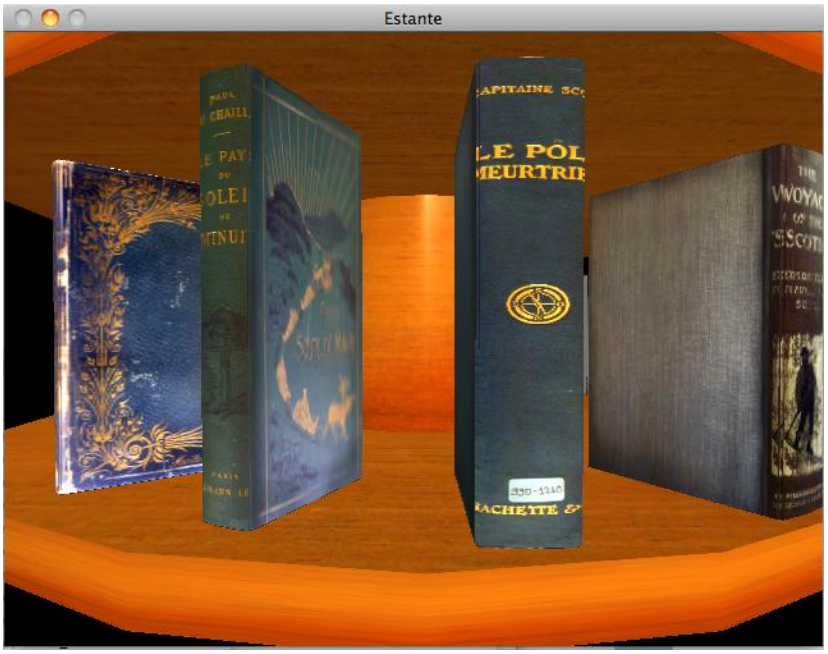


Figura 13. Grafismo final da aplicação.

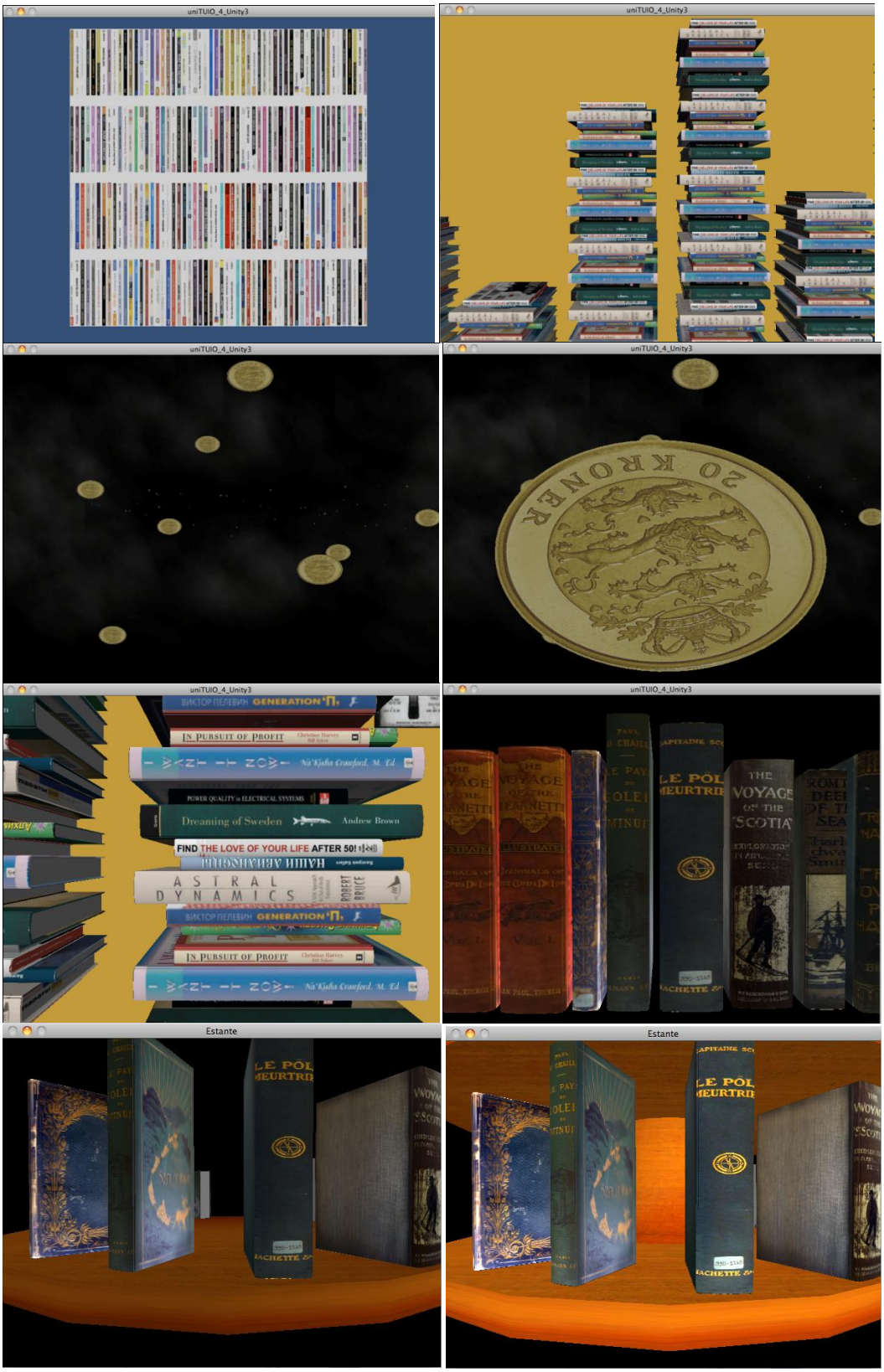


Figura 14. Evolução do grafismo da Estante, da disposição e da selecção do tipo de objecto.

A manipulação dos objectos através do multitoque foi feita com o recurso a scripts em C#, e este trabalho teve por base alguns códigos em C# desenvolvidos pela comunidade de investigação *online*, NUI Group – *Natural User Interface Group*[17] e pelo xtuio[18].

Esta aplicação foi desenvolvida antes da construção da mesa multitoque, o que implicou o recurso a um simulador de TUIO, *TUIO Simulator*, para poderem ser realizados testes de funcionalidade ao protótipo (ver figura 15).

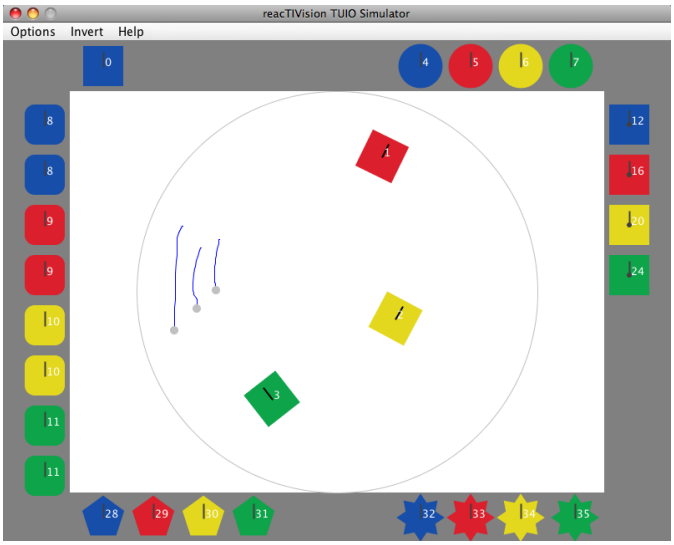


Figura 15 -Tuio Simulator[19]

[17] http://nuigroup.com

[18] http://xtuio.com

[19] http://reactivision.sourceforge.net/ .

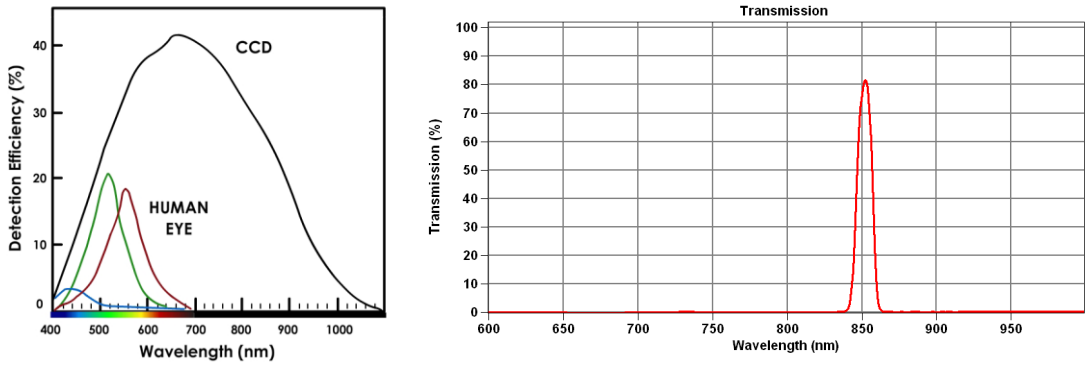
2.3. Processo de desenvolvimento da Superfície Multitoque

1º Passo: Aquisição dos Componentes

A lista dos componentes teve por base, a sugestão de vários autodidactas que colocam as suas experiências *online* (ver apêndice – caderno de encargos). E por esta razão houve problemas de compatibilidades entre componentes, o que implicou voltar a fazer as encomendas e esperar pelo envio, uma vez que nenhum dos componentes é comercializado em Portugal.

2º Passo: Manipulação da câmaraPS3

A camara tem que ser manipulada de forma a captar apenas a amplitude de IR transmitida pelo laser, que neste caso a amplitude é de 850nm (ver figuras 16 e 17). A camara da Playstation é a camara aconselhada por várias comunidades[20] *online* para a construção de superfícies multitoque, pois permite uma velocidade de 60*fps* e com uma resolução de 640x480 píxeis, com excelente preço e performance.



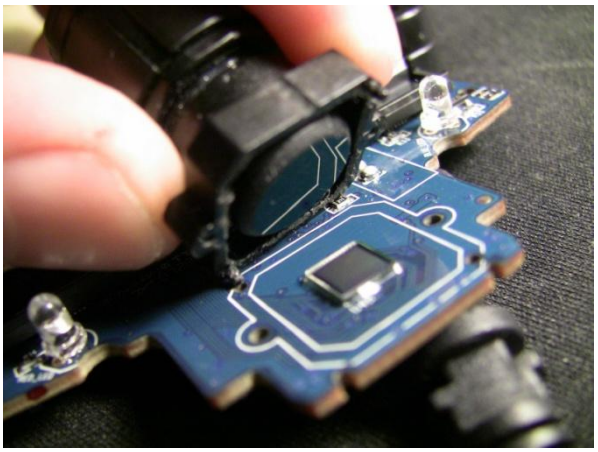
Figuras 16 e 17: Cumprimentos de Onda (à esq.) e representação do cumprimento de onda que é captado pela camera, de 850nm (à dir.).

[20] http://codelaboratories.com/research/view/ps3-eye-disassembly

Camara Playstation PS3

Primeiro é necessário desaparafusar todos os parafusos no exterior e interior de forma a que a placa de circuitos da camara fique solta.

Em seguida é necessário retirar todo o suporte da lente, com muito cuidado, para não riscar o sensor OV07720 CMOS.



Todo o suporte retirado da lente é substituído por novos componentes, nomeadamente um novo suporte para o filtro (M12x0,5), um filtro (850FIB12) e uma lente.

filtro de IR

suporte

Lente

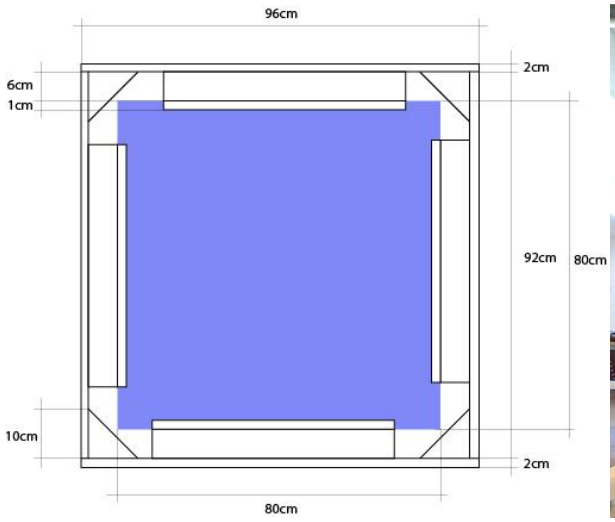
Finalmente pode-se voltar a colocar a camara dentro do revestimento original.

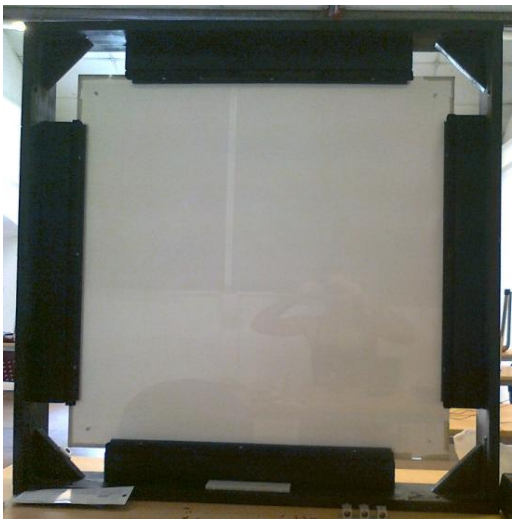
3º Passo: Construção da estrutura

Primeiro é necessário revestir o acrílico com uma tela *rosco* cinzenta, de forma a que fique muito bem esticada.

A construção da estrutura foi particularmente complicada, porque o acrílico com 80cm X 80 cm, tinha que ficar suportado pelo centro das laterais e teria que existir uma margem nos cantos para poder inserir os lasers posteriormente. Para tal foi adquirido:

- 2 tábuas de madeira com C:96cm, L:11cm E:2cm;
- 2 tábuas com C:92cm, L:11cm, E: 2 cm;
- 4 tábuas com C: 60cm, L: 6cm, E: 60cm;
- 8 tábuas com C: 60cm, L: 1cm, E: 1cm;
- 4 tábuas em formato equilátero com 10cm de lado
- cola pregos, pregos e parafusos

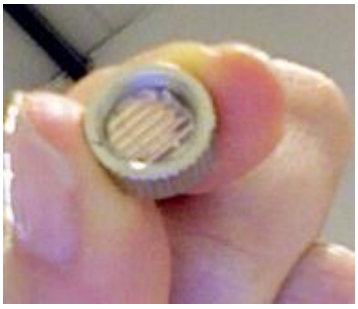




4º Passo: Calibração dos lasers

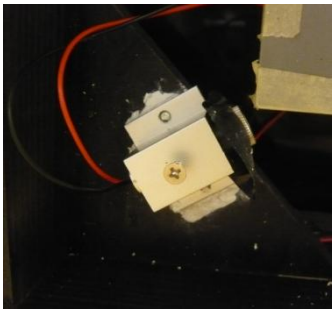
Este passo é possivelmente o que exige uma maior concentração e precisão. Os lasers emitem uma amplitude de onda com 850nm, esta amplitude não é visível a olho nu, aliás é bastante nociva à visão humana. Por esta razão toda a calibração tem que ser feita através da imagem capturada pela camara e visualizada no próprio computador.

Suporte do laser



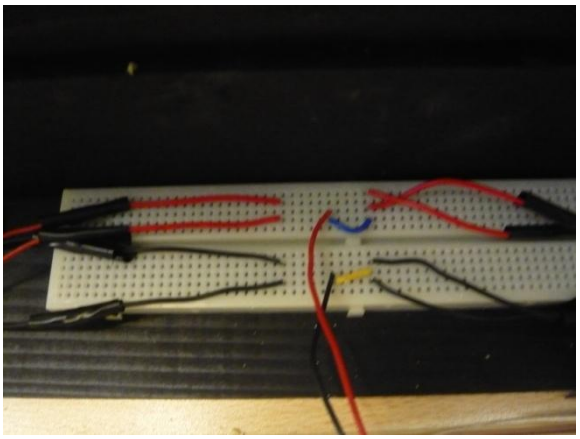
Line Generator 120º

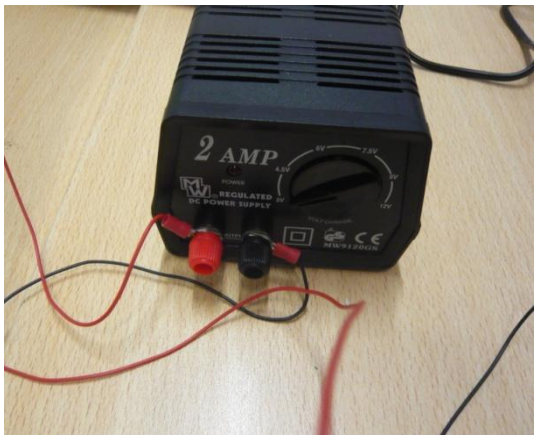
Laser de 850nm



Laser inserido na estrutura: 10m

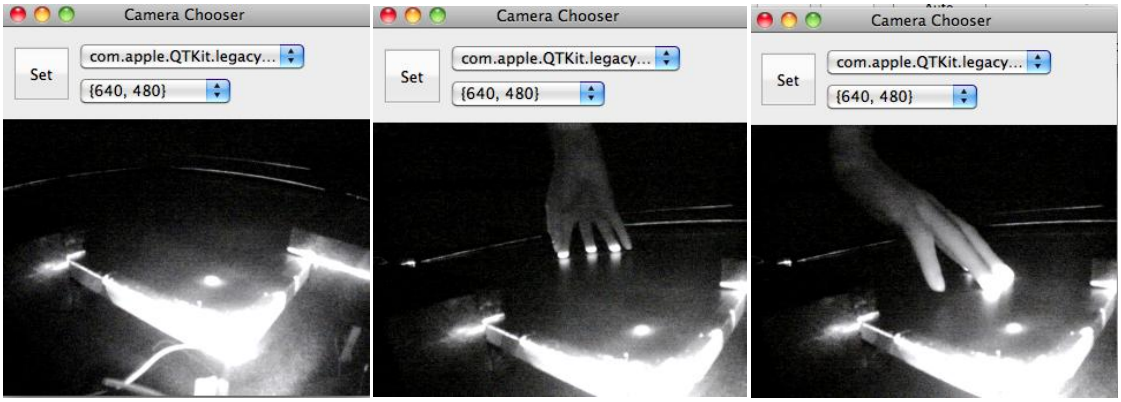
Cada fio de cada laser tem que ser soldado a extensões de fios e ligados a uma *breadboard* (ver figura 18) e a voltagem do transformador não pode ser superior a 3,2v (ver figura 19).





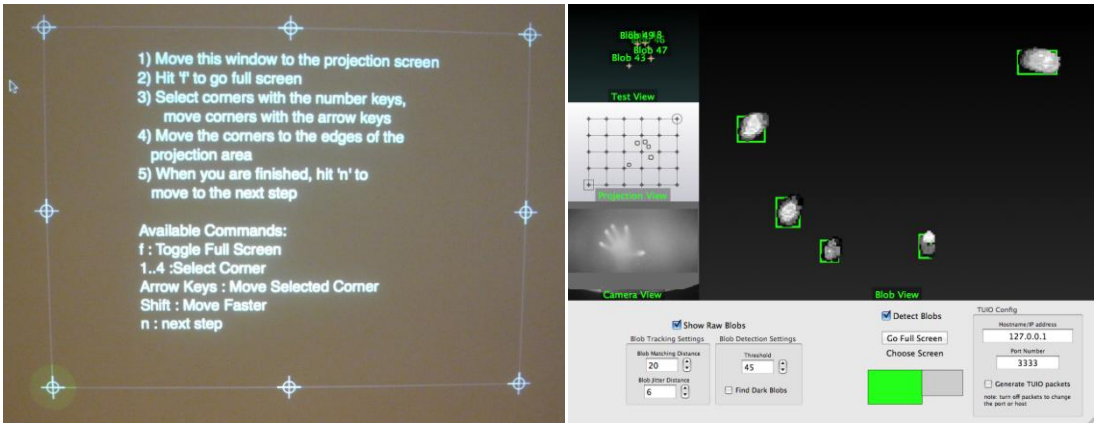
Figuras 18 e 19: Breadboard (à esq), e o transformador (à dir.).

A calibração dos lasers na superfície consiste por um lado em conseguir uma linha fina e definida do laser e por outro que essa linha esteja um milimetro acima da superficie acrilica (ver figuras 20, 21 e 22).



Figuras 20, 21 e 22 – Calibração dos lasers na superficie.

5º Passo: Calibração da Superficie Multitoque



Figuras 23 e 24: Grelha de calibração da superficie(à esq.), identificação dos blobs[21] captados pela câmara(à dir.).

Neste protótipo a calibração da superficie multitoque é feita com o recurso ao programa *xTouch*[22], que calibra e envia o protocolo TUIO para a aplicação desenvolvida em Unity 3, no entanto foram também feitos testes com o recurso ao CCV (Community Core Vision)[23], que tem a mesma utilidade. A figura 24 representa o processo de calibração dos limites da superficie táctil, enquanto que a figura 24 representa a identificação dos *blobs* no programa xTouch.

[21] Blob "An area of interest in an image that is either lighter or darker than its surrounding", Detecção de Blobs consiste em: "Process of detecting regions or areas of interest in an image that are either lighter or darker than their surrounding" e Tracking de Blobs é o processo de "Assigning an identifier (ID) to a blob in order to track it over time". Disponível em http://sethsandler.com/multitouch/terminology/, acedido a 31 de Agosto de 2011.

[22] http://benbritten.com/2010/01/07/xtouch-now-with-more-added-opencv/

[23] http://ccv.nuigroup.com/

Funcionalidades implementadas pelo xTouch

- Manipulação de dados de imagens
- I/O de imagem e vídeo
- Manipulação de matrizes e vetores
- Rotinas de álgebra linear
- Estruturas de dados dinâmicas
- Processamento de imagem básico
- Análise estrutural
- Calibração de câmera
- Análise de movimento (**tracking**)
- Reconhecimento de objetos
- GUI Básico
- Identificação de imagem

2.4 Processamento de Imagem

As figuras que se seguem (25-29), representam o processamento da imagem capturada pela câmara. Uma vez a imagem capturada, o xTouch permite *limpar* de forma a tornar mais nitido os toques na superficie, através das seguintes funcionalidades: *Threshold*, *Brightness*, *Contrast* e *Leve.* Quanto mais nítida a imagem estiver, mais preciso é o processo de identificação dos toques (blobs), cujas coordenadas são porteriormente enviadas para a aplicação Estante Digital.

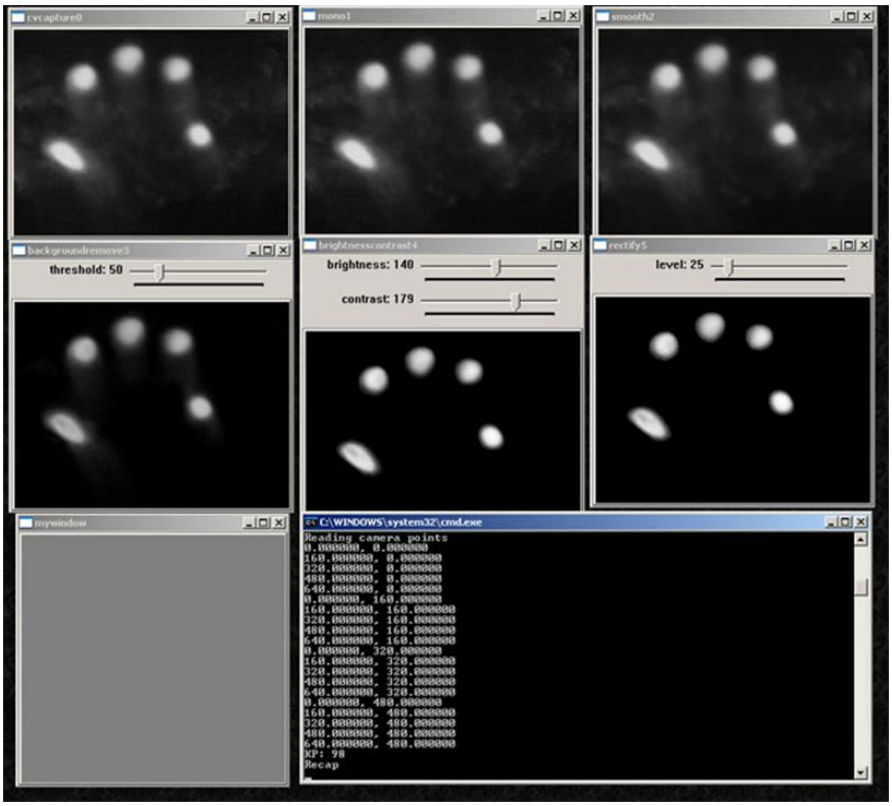


Figura 25 – Manipulação da imagem capturada (após serem capturadas, as imagens contém blobs brancos (as digitais do utilizador), a imagem é processada e as coordenadas dos blobs são obtidas.)

Figuras 26 e 27 - Imagem capturada (à esq.) e identificação do Blob (à dir.).

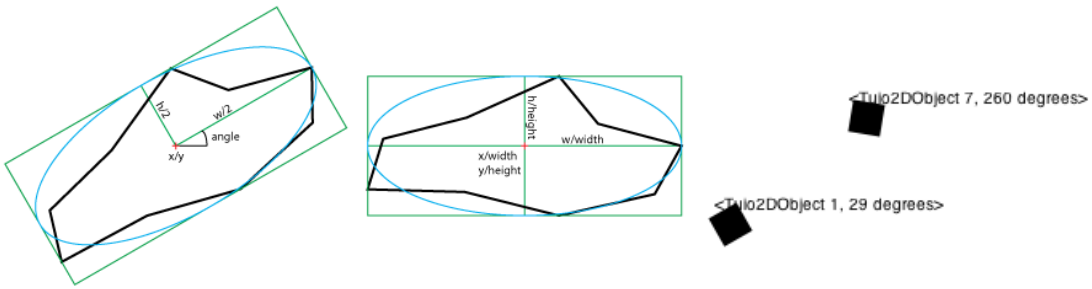


Figura 28 e 29 – Detecção de Blobs (à esq.) e objecto Tuio (à dir.)

2.5 Protocolo TUIO

No presente trabalho recorremos ao protocolo TUIO, que como pode ser observado na figura 30, o protocolo TUIO, define uma interface de comunicação entre aplicações e dispositivos sensíveis a toques. Basicamente este dispositivo recebe informações sobre toques, e envia estas informações através do protocolo TUIO, de forma que outras aplicações possam receber essas informações e processá-las apropriadamente. A figura 30 ilustra este processo, onde é utilizada uma câmara para captar as informações dos toques e de objectos sobre uma mesa, a aplicação que receberá essas informações pode utilizá-las para projectar dados de volta à mesa, no entanto, no presente trabalho recorremos apenas ao toque.

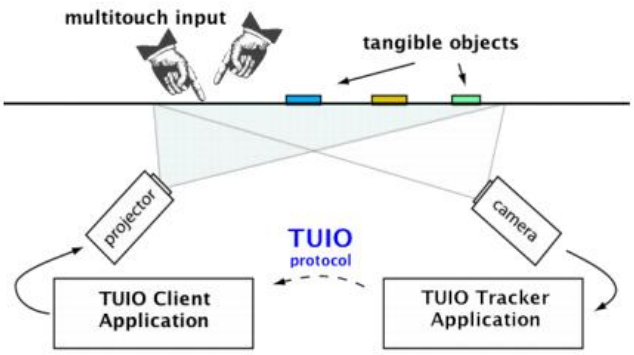


Figura 30 – Exemplo de utilização do protocolo TUIO
http://lvelho.impa.br/i3d09/demos/TuioSketch/node3.html.

TUIO tem por base o protocol Open Sound Control (OSC):

The TUIO protocol is encoded using the Open Sound Control format, which provides an efficient binary encoding method for the transmission of arbitrary controller data. Therefore the TUIO messages can be basically transmitted through any channel that is supported by an actual OSC implementation. The default transport method for the TUIO protocol is the encapsulation of the binary OSC bundle data within UDP packets sent to the default TUIO port number 3333. This default transport method is usually referred as TUIO/UDP, and most implementations are based on this method due to its simplicity and speed when sent over a local or wide area network. Since OSC is not directly bound to a dedicated transport method, alternative transport channels such as TCP can be employed to transmit the OSC encoded TUIO data.

Implementation Details

The TUIO protocol defines two main classes of messages: SET messages and ALIVE messages. SET messages are used to communicate information about an object's state such as position, orientation, and other recognized states. ALIVE messages indicate the current set of objects present on the surface using a list of unique Session IDs.

To avoid possible errors evolving out of packet loss, no explicit ADD or REMOVE messages are included in the TUIO protocol. The receiver deduces object lifetimes by examining the difference between sequential ALIVE messages.

In addition to SET and ALIVE messages, FSEQ messages are defined to uniquely tag each update step with a unique frame sequence ID. An optional

SOURCE message identifies the TUIO source in order to allow source multiplexing on the client side. To summarize:

- Object attributes are sent after each state change using a SET message
- The client deduces object addition and removal from SET and ALIVE messages
- On object removal an updated ALIVE message is sent
- FSEQ messages associate a unique frame id with a bundle of SET and ALIVE messages
- An optional SOURCE message identifies the source application and address

Message & Bundle Format

Since TUIO is implemented using Open Sound Control (OSC) [4] it follows its general syntax. A TUIO client implementation therefore has to make use of an appropriate OSC library such as oscpack or liblo, and has to listen to the following message types and bundle structure, which in this example are referring to the two-dimensional Cursor profile.

```
/tuio/2Dcur source application@address
/tuio/2Dcur alive s_id0 ... s_idN
/tuio/2Dcur set s_id x_pos y_pos x_vel y_vel m_accel
/tuio/2Dcur fseq f_id
```

A typical TUIO bundle will contain an initial ALIVE message, followed by an arbitrary number of SET messages that can fit into the actual bundle capacity and a concluding FSEQ message. A minimal TUIO bundle needs to contain at least the compulsory ALIVE and FSEQ messages. The FSEQ frame ID is incremented for each delivered bundle, while redundant bundles can be marked using the frame sequence ID -1.

Blob Description

The blob profile carries the basic information about untagged generic objects (blobs). The message format describes the inner ellipse of an oriented bounding box, with its center point, the angle of the major axis, the two dimensions as well as the blob area. Therefore this compact format carries information about the approximate elliptical blob enclosure, but also allows the reconstruction of the oriented bounding box. The blob area is normalized in pixels/width*height,

providing quick access to the overall blob size. The blob dimensions are the normalized values after performing an inverse rotation by -angle.[24]

O programa utilizado para enviar messagens OSC no formato TUIO foi o xTouch.

2.6. Arquitectura de Software

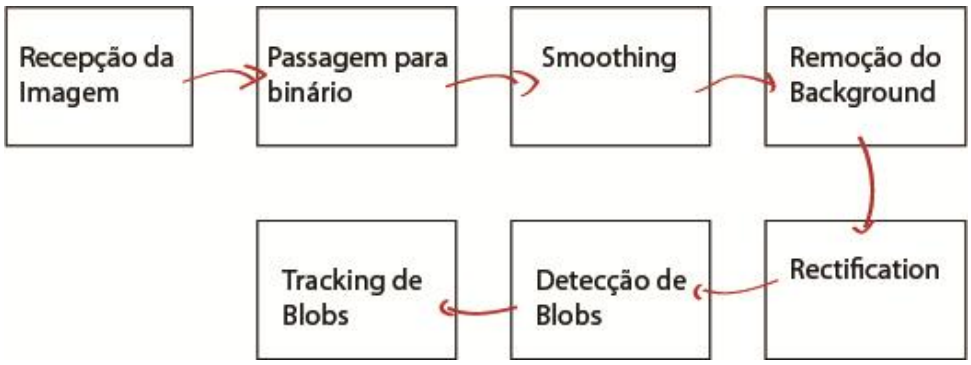


Figura 31 – Fluxograma do Tracking de Blobs

[24] Disponível em http://www.tuio.org/?specification. Acedido a 31 de Agosto de 2011.

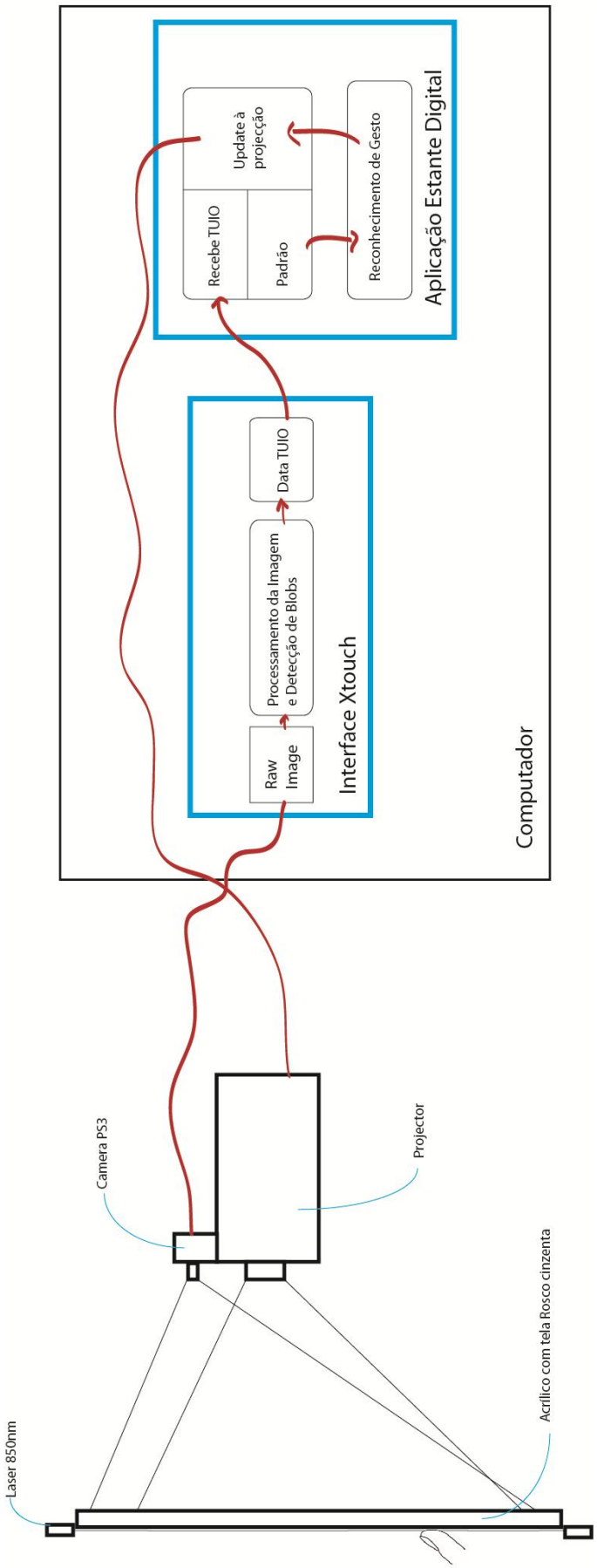


Figura 32- Esquema que representa o fluxo de actividades

3. Resultados

O protótipo tem uma estrutura de A:1m e L1m (ver figura 34), construido na tecnologia LLP, que permite a visualização de artigos digitais (réplicas dos documentos em suporte fisico), permitindo colocar em evidencia características como espessura, cor, estado de conservação e a dimensão.

Permite preservar a integridade dos artigo em surporte fisico, uma vez que esta colecção são livros com quase um século de existência. Para utilizadores que têm deficiências respiratórias ou sofrem de alergias em contacto com os livros ou documentos antigos, desta forma podem aceder aos conteúdos sem constrangimentos fisicos.

O utilizador interage através de linguagem gestual, num ecrã multitoque. O multitoque permite que vários utilizadores interagirem com o sistema em simultâneo, e com os dedos a interacção torna-se mais intuitiva (ver figura 33).

Figura 33 a)– A Estante Digital em Funcionamento

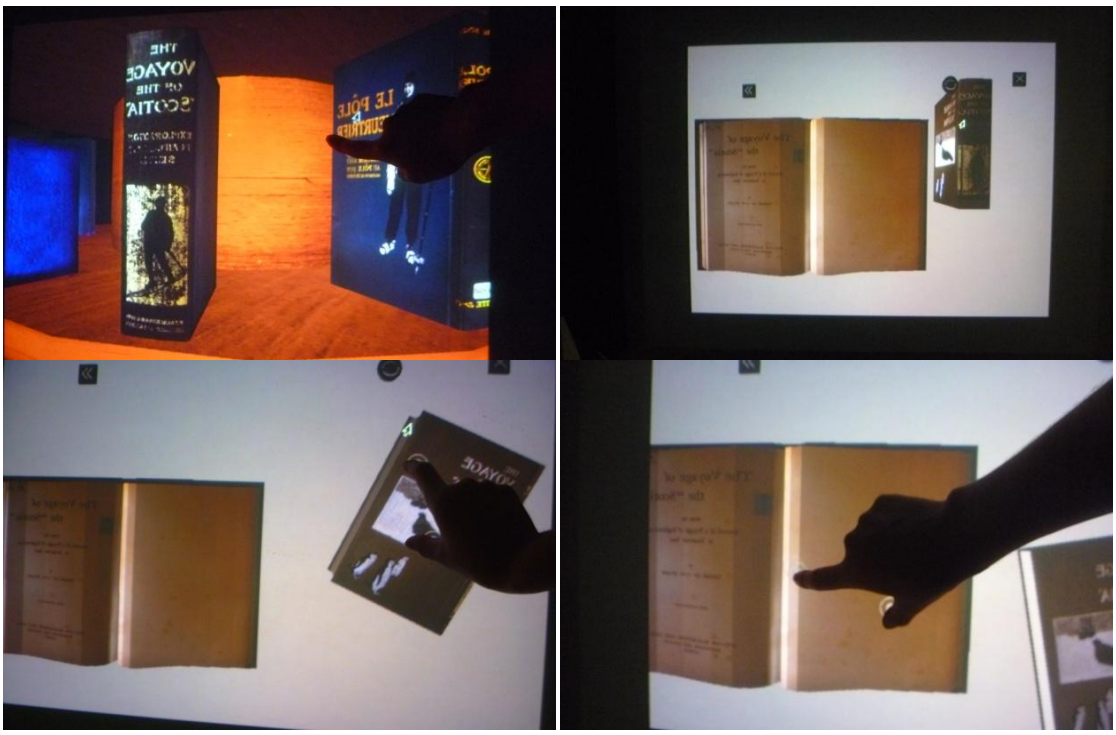


Figura 33 b)– A Estante Digital em Funcionamento

Figura 34 – Superficie Multitoque construída.

Conclusão

Este projecto final de curso, desenvolvido durante o 2º Semestre da Licenciatura em Engenharia Informática (2010/2011), consistiu na elaboração de um protótipo. Este protótipo foi desenvolvido tanto ao nível físico de hardware (ligação dos lasers, manipulação de dispositivos e manipulação de imagem capturada), como na comunicação do hardware e o software e finalmente no desenvolvimento de uma aplicação gráfica que possibilita interacção com um ou vários utilizadores em simultâneo.

Este trabalho possibilitou por um lado colocar em prática conhecimentos de programação adquiridos durante a licenciatura, aplicar conhecimentos de manipulação gráfica de imagens adquiridos em experiências anteriores e adquirir novos conhecimentos ao nível de electrónica.

A elaboração deste protótipo teve alguns impasses, nomeadamente a escolha e aprendizagem de novos softwares como Unity 3 e 3d Studio Max; dificuldade na construção da estrutura que suporta o acrílico, pois inicialmente a superfície iria ser na horizontal, contudo por falta de um projector de curto alcance teve que ser na vertical, o que implicou a construção de uma estrutura em madeira robusta e equilibrada.

Por fim, a calibração dos lasers, por não serem visíveis ao olho humano, esta tinha que ser feita através da imagem capturada pela câmara. Esta tarefa não foi fácil para ser executada por uma pessoa, tive dificuldade em calibrar e segurar na câmara ao mesmo tempo.

No entanto estas dificuldades foram superadas, o protótipo funciona e consegue transmitir com clareza as suas potencialidades para possíveis aplicações futuras.

Referências Bibliográficas

Bullivant, L. (2007). 4dsocial: Interactive Environments. *Architectural Design, Wiley-Academy*, (vol 77, July/August, No 4.). London: Wiley-Academy.

Oosterhuis, K. (2009). *InteractiveWall: Prototype For An Emotive Wall*. Acedido a 19 de Abril de 2010 em http://www.bk.tudelft.nl/live/pagina.jsp?id=bff05884-fb66-4585-b270-0b9c4f079917&lang=en.

Saha, D. & Mukherjee, A. (2003). Pervasive Computing: A Paradigm for the 21st Century. IEEE Computer.

Ullmer, B.; Ishii, H. (2000). Emerging frameworks for tangible user interfaces. In: *IBM Systems Journal*, p. 915931. jul. 2000. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.2676&rep=rep1&type=pdf. Acedido a 10 de Junho de 2011.

Weiser, M. (1991). The Computer for the twenty-first century. *Scientific American* 265(3).

Referências Online

http://gesturecons.com/.

http://nuigroup.com

http://codelaboratories.com/research/view/ps3-eye-disassembly

http://benbritten.com/2010/01/07/xtouch-now-with-more-added-opencv/

http://ccv.nuigroup.com/

Apêndice

Caderno de Encargos – Construção da Superficie Multitoque
Construção de um Laser Light Plane (LLP)

Lista de Componentes necessários	**Imagem**
Camera Sony PlayStation Eye (PS3)	
Suporte da Lente M12x0,5	
Lentes Pack de 6 Lentes	
Filtro 850FIB12 850 10 45	
Acrílico (8 a 10mm de espessura)	
4 x Laser 850nm 10mW 3.2VDC Laser Module	
4 x Line Generator 120º	
Tela Rosco (Grey/Black)	
Projector	
Pasta DAS	
Suporte de laser	

Source Code

Código utilizado para manipular os livros (rodar/ampliar/mover):

```
using UnityEngine;
using System.Collections;

public class BBBasicTouchManipulation : BBSimpleTouchableObject {
	public bool allowDrag = true;
	public bool allowScale = true;
	public bool allowRotate = true;
	public float minimumScale = 1.0f;
	public float maximumScale = 3.0f;
	private Transform saveParent;
	private Vector3 movement;
	protected GameObject pivot;
	public override void handleSingleTouch(iPhoneTouch touch) //toque simples que permite movimentar o objecto no plano da camera
	{
		if (!allowDrag) return;

		movement = touchMovementVector(touch);
		  this.startPivot(gameObject.transform.position);
			pivot.transform.Translate(movement,Space.World);
			this.endPivot();
	}

	public override void handleDoubleTouch(ArrayList events) //o duplo toque pode significar ampliar/reduzir ou rodar num eixo, ou ambos
	{
		if (!allowRotate && !allowScale) return;

		iPhoneTouch touch0 = (iPhoneTouch)events[0];
		iPhoneTouch touch1 = (iPhoneTouch)events[1];
		if (touch0.fingerId > touch1.fingerId) {

			touch0 = (iPhoneTouch)events[1];
			touch1 = (iPhoneTouch)events[0];
		}

	  this.startPivot(gameObject.transform.position);

		//
		//		////////////////////////////////// ROTAÇÃO
		//

		float zDistanceFromCamera = Vector3.Distance(renderingCamera.transform.position,gameObject.transform.position);

		Vector3 screenPosition0 = new Vector3(touch0.position.x,touch0.position.y,zDistanceFromCamera);
		Vector3 lastScreenPosition0 = new Vector3(touch0.position.x - touch0.deltaPosition.x,touch0.position.y -
touch0.deltaPosition.y,zDistanceFromCamera);

		Vector3 screenPosition1 = new Vector3(touch1.position.x,touch1.position.y,zDistanceFromCamera);
		Vector3 lastScreenPosition1 = new Vector3(touch1.position.x - touch1.deltaPosition.x,touch1.position.y -
touch1.deltaPosition.y,zDistanceFromCamera);


		float angleNow = Mathf.Atan2(screenPosition0.x - screenPosition1.x, screenPosition0.y - screenPosition1.y) *
Mathf.Rad2Deg;
		float angleThen = Mathf.Atan2(lastScreenPosition0.x - lastScreenPosition1.x, lastScreenPosition0.y - lastScreenPosition1.y)
* Mathf.Rad2Deg;

		float angleDelta = angleNow - angleThen;

		if (allowRotate) pivot.transform.RotateAround(gameObject.transform.position,renderingCamera.transform.position-
gameObject.transform.position,angleDelta);

		//
		//		////////////////////////////  AMPLIAR/REDUZIR
		//
		if (allowScale) {
			float distNow = (screenPosition0 - screenPosition1).magnitude;
			float distThen = (lastScreenPosition0 - lastScreenPosition1).magnitude;

			float scale = distNow/distThen;

			if (transform.localScale.x * scale < minimumScale) scale = minimumScale/transform.localScale.x;
			if (transform.localScale.x * scale > maximumScale) scale = maximumScale/transform.localScale.x;

			Vector3 local = pivot.transform.localScale;

			local.x *= scale;
			local.y *= scale;
			local.z *= scale;
			pivot.transform.localScale = local;
		}
```

```
                this.endPivot();
        }

        ////// start Pivot
        virtual protected void startPivot(Vector3 pivotPosition)
        {
                if (pivot == null) {
                        pivot = new GameObject();
                        pivot.name = "BBBasicTouchManipulation Pivot";
                        pivot.transform.position = pivotPosition;
                }

                saveParent = gameObject.transform.parent;
                gameObject.transform.parent = null;
                pivot.transform.parent = saveParent;
                gameObject.transform.parent = pivot.transform;
        }
        ////// end Pivot
        virtual protected void endPivot()
        {
                gameObject.transform.parent = saveParent;
                pivot.transform.parent = null;
                Destroy(pivot);
        }
        ////// touchMovementVector
        public Vector3 touchMovementVector(iPhoneTouch touch)
        {
                float zDistanceFromCamera = Vector3.Distance(renderingCamera.transform.position,gameObject.transform.position);

                Vector3 screenPosition = new Vector3(touch.position.x,touch.position.y,zDistanceFromCamera);
                Vector3 lastScreenPosition = new Vector3(touch.position.x - touch.deltaPosition.x,touch.position.y -
touch.deltaPosition.y,zDistanceFromCamera);

                Vector3 cameraWorldPosition = this.renderingCamera.ScreenToWorldPoint(screenPosition);
                Vector3 lastCameraWorldPosition = this.renderingCamera.ScreenToWorldPoint(lastScreenPosition);

                return cameraWorldPosition - lastCameraWorldPosition;
        }


}
```

Código que permite a estante cilindrica rodar em torno no seu eixo central:

O que difere o codigo seguinte do anterior é que apenas com um toque simples, a única acção possivel é rodar em torno de um eixo Y central.

```
public override void handleSingleTouch(iPhoneTouch touch)
        {
                if (!allowDrag) return;


                  this.startPivot(gameObject.transform.position);

                        float zDistanceFromCamera =
Vector3.Distance(renderingCamera.transform.position,gameObject.transform.position);

                Vector3 screenPosition0 = new Vector3(0,-3,-2); // eixo sobre o qual o cilindro roda
                Vector3 lastScreenPosition0 = new Vector3(0,-3,-2); // eixo sobre o qual o cilindro roda
                Vector3 screenPosition1 = new Vector3(touch.position.x,touch.position.y,zDistanceFromCamera); //posição do toque
                Vector3 lastScreenPosition1 = new Vector3(touch.position.x - touch.deltaPosition.x,touch.position.y -
touch.deltaPosition.y,zDistanceFromCamera);

                float angleNow = Mathf.Atan2[25](screenPosition0.x - screenPosition1.x, screenPosition0.y - screenPosition1.y) *
Mathf.Rad2Deg[26];
                float angleThen = Mathf.Atan2(lastScreenPosition0.x - lastScreenPosition1.x, lastScreenPosition0.y - lastScreenPosition1.y)
* Mathf.Rad2Deg;

                float angleDelta = angleNow - angleThen; // o ângulo a rodar é a difrença entre o angulo do toque inicial e o angulo que o
toque final

                        pivot.transform.RotateAround(gameObject.transform.position,Vector3.forward,-angleDelta);
                        this.endPivot();

        }
```

[25] Devolve o ângulo em radianos

[26] Converte radianos para graus

Código utilizado para seleccionar o livro:

```
var myLevel : String;

function doTouchDown () { // se o objecto estiver pressionado, o programa carrega outra cena “myLevel”

	Application.LoadLevel(myLevel);

}
```

Para que o programa identifique que o objecto está pressionado é utilizado o seguinte código (identifica um, dois ou mais toques):

```
using UnityEngine;
using System.Collections;

public class BBTouchableButton : BBSimpleTouchableObject {

	public GameObject notificationObject;
	public string touchDownMessage = "doTouchDown";
	public string touchUpMessage = "doTouchUp";

	private bool touchDown = false;

	public override void startup ()
	{
		if (notificationObject == null) notificationObject = gameObject;
	}

	public override void handleSingleTouch(iPhoneTouch aTouch)
	{
		if (touchDown) return;
		notificationObject.SendMessage(touchDownMessage,SendMessageOptions.DontRequireReceiver);
		touchDown = true;
	}

	public override void handleManyTouches(ArrayList touches)
	{
		handleSingleTouch(touches[0] as iPhoneTouch);
	}

	public override void handleDoubleTouch(ArrayList touches)
	{
		handleSingleTouch(touches[0] as iPhoneTouch);
	}

	public override void noTouches()
	{
		if (touchDown) notificationObject.SendMessage(touchUpMessage,SendMessageOptions.DontRequireReceiver);
		touchDown = false;
	}


}
```

Código desenvolvido por reacTIVISION, que permite ao Unit 3 receber as mensagens TUIO – OPEN SOURCE

```
/*
	TUIO C# Library - part of the reacTIVision project
	http://reactivision.sourceforge.net/

	Copyright (c) 2005-2009 Martin Kaltenbrunner <mkalten@iua.upf.edu>

	This program is free software; you can redistribute it and/or modify
	it under the terms of the GNU General Public License as published by
	the Free Software Foundation; either version 2 of the License, or
	(at your option) any later version.
*/

using System;
using System.Threading;
using System.Collections;
using System.Collections.Generic;

using OSC.NET;

public class BBTuioClient {}

namespace TUIO
{
	/**
	 * The TuioClient class is the central TUIO protocol decoder component. It provides a simple callback infrastructure using the {@link
```

```
TuioListener} interface.
	* In order to receive and decode TUIO messages an instance of TuioClient needs to be created. The TuioClient instance then generates
TUIO events
	* which are broadcasted to all registered classes that implement the {@link TuioListener} interface.<P>
	* <code>
	* TuioClient client = new TuioClient();<br/>
	* client.addTuioListener(myTuioListener);<br/>
	* client.start();<br/>
	* </code>
	*
	* @author Martin Kaltenbrunner
	* @version 1.4
	*/
	public class TuioClient
	{
		private bool connected = false;
		private int port = 3333;
		private OSCReceiver receiver;
		private Thread thread;

		private object cursorSync = new object();
		private object objectSync = new object();

		private Dictionary<long,TuioObject> objectList = new Dictionary<long,TuioObject>(32);
		private List<long> aliveObjectList = new List<long>(32);
		private List<long> newObjectList = new List<long>(32);
		private Dictionary<long,TuioCursor> cursorList = new Dictionary<long,TuioCursor>(32);
		private List<long> aliveCursorList = new List<long>(32);
		private List<long> newCursorList = new List<long>(32);
		private List<TuioObject> frameObjects = new List<TuioObject>(32);
		private List<TuioCursor> frameCursors = new List<TuioCursor>(32);

		private List<TuioCursor> freeCursorList = new List<TuioCursor>();
		private int maxCursorID = -1;

		private int currentFrame = 0;
		private TuioTime currentTime;

		private List<TuioListener> listenerList = new List<TuioListener>();

		private string statusString;
		private int packetCount = 0;
		private int packetErrorCount = 0;
		private int bundleCount = 0;
		private int messageCount = 0;
		private int cursorCount = 0;
		private int objectCount = 0;

		/**
		 * The default constructor creates a client that listens to the default TUIO port 3333
		 */
		public TuioClient() {}

		/**
		 * This constructor creates a client that listens to the provided port
		 *
		 * @param port the listening port number
		 */
		public TuioClient(int port) {
			this.port = port;
		}

		/**
		 * Returns the port number listening to.
		 *
		 * @return the listening port number
		 */
		public int getPort() {
			return port;
		}

		/**
		 * The TuioClient starts listening to TUIO messages on the configured UDP port
		 * All reveived TUIO messages are decoded and the resulting TUIO events are broadcasted to all registered TuioListeners
		 */
		public void connect() {

			TuioTime.initSession();
			currentTime = new TuioTime();
			currentTime.reset();

			try {
				receiver = new OSCReceiver(port);
				connected = true;
				statusString = "Thread Started";
				thread = new Thread(new ThreadStart(listen));
```

```
                                        thread.Start();
                                } catch (Exception e) {
                                        Console.WriteLine("failed to connect to port "+port);
                                        Console.WriteLine(e.Message);
                                        statusString = "failed to connect to port " + port + " " + e.Message;
                                }
                        }

                        /**
                         * The TuioClient stops listening to TUIO messages on the configured UDP port
                         */
                        public void disconnect() {
                                        if (receiver!=null) receiver.Close();
                                        receiver = null;

                                        aliveObjectList.Clear();
                                        aliveCursorList.Clear();
                                        objectList.Clear();
                                        cursorList.Clear();
                                        freeCursorList.Clear();
                                        frameObjects.Clear();
                                        frameCursors.Clear();

                                        connected = false;
                        }

                        /**
                         * Returns true if this TuioClient is currently connected.
                         * @return   true if this TuioClient is currently connected
                         */
                        public bool isConnected() { return connected; }
                public int currentFrameNumber() { return currentFrame; }
                public string getStatusString()
                {
                        return statusString + " PC:" + packetCount + " PEC:" + packetErrorCount+ " BC:" + bundleCount+ " MC:" + messageCount
+ " OC:" + objectCount + " CC:" + cursorCount ;
                }

                        private void listen() {
                                statusString = "Listening:" + connected;
                                while(connected) {
                                        try {
                                                OSCPacket packet = receiver.Receive();
                                                packetCount++;
                                                if (packet!=null) {
                                                        if (packet.IsBundle()) {
                                                                bundleCount++;
                                                                ArrayList messages = packet.Values;
                                                                for (int i=0; i<messages.Count; i++) {
                                                                        messageCount++;
                                                                        processMessage((OSCMessage)messages[i]);
                                                                }
                                                        } else processMessage((OSCMessage)packet);
                                                } else {
                                                        packetErrorCount++;
                                                        Console.WriteLine("null packet");
                                                }
                                        } catch (Exception e) {
                                                Console.WriteLine(e.Message);
                                                statusString = "CATCH IN LISTEN THREAD: " + e.Message;
                                        }
                                }
                        }

                        /**
                         * The OSC callback method where all TUIO messages are received and decoded
                         * and where the TUIO event callbacks are dispatched
                         *
                         * @param message       the received OSC message
                         */

                        private void processMessage(OSCMessage message) {
                                string address = message.Address;
                                ArrayList args = message.Values;
                                string command = (string)args[0];

                                if (address == "/tuio/2Dobj") {
                                        objectCount++;
                                        if (command == "set") {

                                                long s_id = (int)args[1];
                                                int f_id = (int)args[2];
                                                float xpos = (float)args[3];
                                                float ypos = (float)args[4];
                                                float angle = (float)args[5];
                                                float xspeed = (float)args[6];
```

```
                                                float yspeed = (float)args[7];
                                                float rspeed = (float)args[8];
                                                float maccel = (float)args[9];
                                                float raccel = (float)args[10];

                                                lock(objectSync) {
                                                        if (!objectList.ContainsKey(s_id)) {
                                                                TuioObject addObject = new
TuioObject(s_id,f_id,xpos,ypos,angle);
                                                                frameObjects.Add(addObject);
                                                        } else {
                                                                TuioObject tobj = objectList[s_id];
                                                                if (tobj==null) return;
                                                                if((tobj.getX()!=xpos) || (tobj.getY()!=ypos) ||
(tobj.getAngle()!=angle) || (tobj.getXSpeed()!=xspeed) || (tobj.getYSpeed()!=yspeed) || (tobj.getRotationSpeed()!=rspeed) ||
(tobj.getMotionAccel()!=maccel) || (tobj.getRotationAccel()!=raccel)) {

                                                                        TuioObject updateObject = new
TuioObject(s_id,f_id,xpos,ypos,angle);

        updateObject.update(xpos,ypos,angle,xspeed,yspeed,rspeed,maccel,raccel);
                                                                        frameObjects.Add(updateObject);
                                                                }
                                                        }
                                                }

                                        } else if (command == "alive") {

                                                newObjectList.Clear();
                                                for (int i=1;i<args.Count;i++) {
                                                        // get the message content
                                                        long s_id = (int)args[i];
                                                        newObjectList.Add(s_id);
                                                        // reduce the object list to the lost objects
                                                        if (aliveObjectList.Contains(s_id))
                                                                aliveObjectList.Remove(s_id);
                                                }

                                                // remove the remaining objects
                                                lock(objectSync) {
                                                        for (int i=0;i<aliveObjectList.Count;i++) {
                                                                long s_id = aliveObjectList[i];
                                                                TuioObject removeObject = objectList[s_id];
                                                                removeObject.remove(currentTime);
                                                                frameObjects.Add(removeObject);
                                                        }
                                                }

                                        } else if (command=="fseq") {
                                                int fseq = (int)args[1];
                                                bool lateFrame = false;

                                                if (fseq>0) {
                                                        if (fseq>currentFrame) currentTime = TuioTime.getSessionTime();
                                                        if ((fseq>=currentFrame) || ((currentFrame-fseq)>100)) currentFrame =
fseq;
                                                        else lateFrame = true;
                                                } else if ((TuioTime.getSessionTime().getTotalMilliseconds()-
currentTime.getTotalMilliseconds())>100) {
                                                        currentTime = TuioTime.getSessionTime();
                                                }

                                                if (!lateFrame) {

                                                        IEnumerator<TuioObject> frameEnum = frameObjects.GetEnumerator();
                                                        while(frameEnum.MoveNext()) {
                                                                TuioObject tobj = frameEnum.Current;

                                                                switch (tobj.getTuioState()) {
                                                                        case TuioObject.TUIO_REMOVED:
                                                                                TuioObject removeObject = tobj;
                                                                                removeObject.remove(currentTime);

                                                                                for (int i=0;i<listenerList.Count;i++)
{
                                                                                        TuioListener listener =
(TuioListener)listenerList[i];
                                                                                        if (listener!=null)
listener.removeTuioObject(removeObject);
                                                                                }
                                                                                lock(objectSync) {

        objectList.Remove(removeObject.getSessionID());
                                                                                }
                                                                                break;
                                                                        case TuioObject.TUIO_ADDED:
```

```
                                                                                TuioObject addObject = new
TuioObject(currentTime,tobj.getSessionID(),tobj.getSymbolID(),tobj.getX(),tobj.getY(),tobj.getAngle());
                                                                                lock(objectSync) {

                objectList.Add(addObject.getSessionID(),addObject);
                                                                                }
                                                                                for (int i=0;i<listenerList.Count;i++)
{
                                                                                        TuioListener listener =
(TuioListener)listenerList[i];
                                                                                        if (listener!=null)
listener.addTuioObject(addObject);
                                                                                }
                                                                                break;
                                                                        default:
                                                                                TuioObject updateObject =
getTuioObject(tobj.getSessionID());
                                                                                if (
(tobj.getX()!=updateObject.getX() && tobj.getXSpeed()==0) || (tobj.getY()!=updateObject.getY() && tobj.getYSpeed()==0) )

                updateObject.update(currentTime,tobj.getX(),tobj.getY(),tobj.getAngle());
                                                                                else

                updateObject.update(currentTime,tobj.getX(),tobj.getY(),tobj.getAngle(),tobj.getXSpeed(),tobj.getYSpeed(),tobj.getRotationSpeed(),tobj.
getMotionAccel(),tobj.getRotationAccel());

                                                                                for (int i=0;i<listenerList.Count;i++)
{
                                                                                        TuioListener listener =
(TuioListener)listenerList[i];
                                                                                        if (listener!=null)
listener.updateTuioObject(updateObject);
                                                                                }
                                                                                break;
                                                                        }
                                                                }

                                                                for (int i=0;i<listenerList.Count;i++) {
                                                                        TuioListener listener = (TuioListener)listenerList[i];
                                                                        if (listener!=null) listener.refresh(new
TuioTime(currentTime));
                                                                }

                                                                List<long> buffer = aliveObjectList;
                                                                aliveObjectList = newObjectList;
                                                                // recycling the List
                                                                newObjectList = buffer;
                                                        }
                                                        frameObjects.Clear();
                                                }

                                        } else if (address == "/tuio/2Dcur") {
                                                cursorCount++;
                                                if (command == "set") {

                                                        long s_id = (int)args[1];
                                                        float xpos = (float)args[2];
                                                        float ypos = (float)args[3];
                                                        float xspeed = (float)args[4];
                                                        float yspeed = (float)args[5];
                                                        float maccel = (float)args[6];

                                                        lock(cursorList) {
                                                                if (!cursorList.ContainsKey(s_id)) {

                                                                        TuioCursor addCursor = new TuioCursor(s_id,-1,xpos,ypos);
                                                                        frameCursors.Add(addCursor);

                                                                } else {
                                                                        TuioCursor tcur = (TuioCursor)cursorList[s_id];
                                                                        if (tcur==null) return;
                                                                        if ((tcur.getX()!=xpos) || (tcur.getY()!=ypos) ||
(tcur.getXSpeed()!=xspeed) || (tcur.getYSpeed()!=yspeed) || (tcur.getMotionAccel()!=maccel)) {
                                                                                TuioCursor updateCursor = new
TuioCursor(s_id,tcur.getCursorID(),xpos,ypos);

                updateCursor.update(xpos,ypos,xspeed,yspeed,maccel);
                                                                                frameCursors.Add(updateCursor);
                                                                        }
                                                                }
                                                        }

                                                } else if (command == "alive") {

                                                        newCursorList.Clear();
                                                        for (int i=1;i<args.Count;i++) {
```

```
                                        // get the message content
                                        long s_id = (int)args[i];
                                        newCursorList.Add(s_id);
                                        // reduce the cursor list to the lost cursors
                                        if (aliveCursorList.Contains(s_id))
                                                aliveCursorList.Remove(s_id);
                                }

                                // remove the remaining cursors
                                lock(cursorSync) {
                                        for (int i=0;i<aliveCursorList.Count;i++) {
                                                long s_id = aliveCursorList[i];
                                                if (!cursorList.ContainsKey(s_id)) continue;
                                                TuioCursor removeCursor = cursorList[s_id];
                                                removeCursor.remove(currentTime);
                                                frameCursors.Add(removeCursor);
                                        }
                                }

                        } else if (command=="fseq") {
                                int fseq = (int)args[1];
                                bool lateFrame = false;

                                if (fseq>0) {
                                        if (fseq>currentFrame) currentTime = TuioTime.getSessionTime();
                                        if ((fseq>=currentFrame) || ((currentFrame-fseq)>100)) currentFrame =
fseq;
                                        else lateFrame = true;
                                } else if ((TuioTime.getSessionTime().getTotalMilliseconds()-
currentTime.getTotalMilliseconds())>100) {
                                        currentTime = TuioTime.getSessionTime();
                                }

                                if (!lateFrame) {

                                        IEnumerator<TuioCursor> frameEnum = frameCursors.GetEnumerator();
                                        while(frameEnum.MoveNext()) {
                                                TuioCursor tcur = frameEnum.Current;
                                                switch (tcur.getTuioState()) {
                                                        case TuioCursor.TUIO_REMOVED:
                                                                TuioCursor removeCursor = tcur;
                                                                removeCursor.remove(currentTime);

                                                                for (int i=0;i<listenerList.Count;i++)
{
                                                                        TuioListener listener =
(TuioListener)listenerList[i];
                                                                        if (listener!=null)
listener.removeTuioCursor(removeCursor);
                                                                }
                                                                lock(cursorSync) {

        cursorList.Remove(removeCursor.getSessionID());

                                                                        if
(removeCursor.getCursorID() == maxCursorID) {

        maxCursorID = -1;

                                                                                if
(cursorList.Count > 0) {

        IEnumerator<KeyValuePair<long, TuioCursor>> clist = cursorList.GetEnumerator();

        while (clist.MoveNext()) {

        int f_id = clist.Current.Value.getCursorID();

        if (f_id > maxCursorID) maxCursorID = f_id;

        }


        List<TuioCursor> freeCursorBuffer = new List<TuioCursor>();

        IEnumerator<TuioCursor> flist = freeCursorList.GetEnumerator();

        while (flist.MoveNext()) {

        TuioCursor testCursor = flist.Current;

        if (testCursor.getCursorID() < maxCursorID) freeCursorBuffer.Add(testCursor);

        }
```

```
            freeCursorList = freeCursorBuffer;
                                                                                                                    } else
freeCursorList.Clear();
                                                                                                                } else if
(removeCursor.getCursorID() < maxCursorID) freeCursorList.Add(removeCursor);
                                                                                                            }
                                                                                                            break;

                                                                                        case TuioCursor.TUIO_ADDED:
                                                                                                TuioCursor addCursor;
                                                                                                lock(cursorSync) {
                                                                                                        int c_id = cursorList.Count;
                                                                                                        if
((cursorList.Count<=maxCursorID) && (freeCursorList.Count>0)) {
                                                                                                                TuioCursor
closestCursor = freeCursorList[0];

            IEnumerator<TuioCursor> testList = freeCursorList.GetEnumerator();

            while(testList.MoveNext()) {
                                                                                                                        TuioCursor
testCursor = testList.Current;
                                                                                                                        if
(testCursor.getDistance(tcur)<closestCursor.getDistance(tcur)) closestCursor = testCursor;
                                                                                                                }
                                                                                                                c_id =
closestCursor.getCursorID();

            freeCursorList.Remove(closestCursor);
                                                                                                        } else maxCursorID = c_id;

                                                                                                        addCursor = new
TuioCursor(currentTime,tcur.getSessionID(),c_id,tcur.getX(),tcur.getY());

            cursorList.Add(addCursor.getSessionID(),addCursor);
                                                                                                }

                                                                                                for (int i=0;i<listenerList.Count;i++) {
                                                                                                        TuioListener listener =
(TuioListener)listenerList[i];
                                                                                                        if (listener!=null)
listener.addTuioCursor(addCursor);
                                                                                                }
                                                                                                break;

                                                                                        default:
                                                                                                TuioCursor updateCursor =
getTuioCursor(tcur.getSessionID());
                                                                                                if ( (tcur.getX()!=updateCursor.getX() &&
tcur.getXSpeed()==0) || (tcur.getY()!=updateCursor.getY() && tcur.getYSpeed()==0) )

            updateCursor.update(currentTime,tcur.getX(),tcur.getY());
                                                                                                else

            updateCursor.update(currentTime,tcur.getX(),tcur.getY(),tcur.getXSpeed(),tcur.getYSpeed(),tcur.getMotionAccel());

                                                                                                for (int i=0;i<listenerList.Count;i++) {
                                                                                                        TuioListener listener =
(TuioListener)listenerList[i];
                                                                                                        if (listener!=null)
listener.updateTuioCursor(updateCursor);
                                                                                                }
                                                                                                break;
                                                                                        }
                                                                                }

                                                                                for (int i=0;i<listenerList.Count;i++) {
                                                                                        TuioListener listener = (TuioListener)listenerList[i];
                                                                                        if (listener!=null) listener.refresh(new
TuioTime(currentTime));
                                                                                }

                                                                                List<long> buffer = aliveCursorList;
                                                                                aliveCursorList = newCursorList;
                                                                                // recycling the List
                                                                                newCursorList = buffer;
                                                                        }
                                                                        frameCursors.Clear();
                                                                }

                                                        }
                                                }

                                                /**
                                                 * Adds the provided TuioListener to the list of registered TUIO event listeners
                                                 *
```

```
		 * @param listener the TuioListener to add
		 */
		public void addTuioListener(TuioListener listener) {
			listenerList.Add(listener);
		}

		/**
		 * Removes the provided TuioListener from the list of registered TUIO event listeners
		 *
		 * @param listener the TuioListener to remove
		 */
		public void removeTuioListener(TuioListener listener) {
			listenerList.Remove(listener);
		}

		/**
		 * Removes all TuioListener from the list of registered TUIO event listeners
		 */
		public void removeAllTuioListeners() {
			listenerList.Clear();
		}

		/**
		 * Returns a Vector of all currently active TuioObjects
		 *
		 * @return a Vector of all currently active TuioObjects
		 */
		public List<TuioObject> getTuioObjects() {
			List<TuioObject> listBuffer;
			lock(objectSync) {
				listBuffer = new List<TuioObject>(objectList.Values);
			}
			return listBuffer;
		}

		/**
		 * Returns a Vector of all currently active TuioCursors
		 *
		 * @return a Vector of all currently active TuioCursors
		 */
		public List<TuioCursor> getTuioCursors() {
			List<TuioCursor> listBuffer;
			lock(cursorSync) {
				listBuffer = new List<TuioCursor>(cursorList.Values);
			}
			return listBuffer;
		}

		/**
		 * Returns the TuioObject corresponding to the provided Session ID
		 * or NULL if the Session ID does not refer to an active TuioObject
		 *
		 * @return an active TuioObject corresponding to the provided Session ID or NULL
		 */
		public TuioObject getTuioObject(long s_id) {
			TuioObject tobject = null;
			lock(objectSync) {
				objectList.TryGetValue(s_id,out tobject);
			}
			return tobject;
		}

		/**
		 * Returns the TuioCursor corresponding to the provided Session ID
		 * or NULL if the Session ID does not refer to an active TuioCursor
		 *
		 * @return an active TuioCursor corresponding to the provided Session ID or NULL
		 */
		public TuioCursor getTuioCursor(long s_id) {
			TuioCursor tcursor = null;
			lock(cursorSync) {
				cursorList.TryGetValue(s_id, out tcursor);
			}
			return tcursor;
		}

	}
}
```