FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Sketch-based Facial Modeling and Animation: an approach based on mobile devices

Ana Luísa de Vila Fernandes Orvalho

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Augusto Sousa (FEUP)

Co-supervisor: Prof. Verónica Orvalho (FCUP)

October 4, 2013

Resumo

Transmitir emoções credíveis à audiência através da animação facial de personagens virtuais requer um nível elevado de realismo. No entanto, este grau de qualidade é muito difícil de atingir dado que o *software* tradicional para modelação e animação, apesar de muito poderoso, é também extremamente complexo, já que oferece ao utilizador uma grande variedade de elementos de interface geralmente referidos como WIMP (*Window*, *Icon*, *Menu*, *Pointer*). Dada a sua longa curva de aprendizagem, o uso deste tipo de *software* está apenas ao alcance de artistas experientes. Mas, com a evolução das interfaces de utilizador, novas abordagens ao modelo de interação humano-máquina provaram ser alternativas válidas ao paradigma WIMP. Estas alternativas, conhecidas como interfaces naturais de utilizador, procuram possibilitar que os utilizadores interajam com os computadores da mesma forma que interagem com o mundo que os rodeia, tirando partido de capacidades como o toque, a fala, a escrita manual e o movimento ocular. Esta dissertação propõe o uso de uma abordagem com recurso a desenhos para manipular a estrutura de controlo de modelos 3D, conhecida como *rig*, através de um *tablet* com capacidade multitoque, que não requer qualquer dispositivo de entrada de dados para além dos dedos do utilizador. Esta abordagem, baseada num algoritmo já existente que recorre a simples desenhos, pretende explorar as novas possibilidades oferecidas pelos dispositivos móveis que permitam simplificar significativamente o processo de animação, tornando-o acessível a utilizadores sem experiência nesta área. A solução proposta foi especialmente orientada para o iPad, da Apple, e desenvolvida usando o motor de jogo Unity3D para que se integrasse de forma simples no protótipo do projeto LIFEisGAME que inspirou este trabalho - um jogo sério que pretende ajudar crianças com Desordens do Espetro Autista (DEA) a reconhecer e expressar emoções através de expressões faciais, de forma interativa e creativa.

Abstract

Realistic facial animation is essential so that virtual characters can convey believable emotions to the audience. However, it is very hard to achieve since traditional modeling and animation software, although very powerful, is also extremely complex, providing to the user a series of interface elements commonly referred as WIMP (Window, Icon, Menu, Pointer). As they require significant learning time, these types of software are only accessible to experienced artists. But with the evolution of user interfaces, new approaches to the human-machine interaction model have proven to be valid alternatives to the WIMP paradigm. These alternatives, know as natural user interfaces, aim to enable users to interact with computers in the same way as they interact with the world, taking advantage of abilities such as touch, speech, handwriting and eye-gazing. This dissertation proposes the use of an hybrid sketching and dragging approach to manipulate a model's control structure, know as rig, through the use of a multi-touch capable tablet that requires no intermediate input device other that the users' fingers. This approach, based on a pre-existing sketching algorithm aims to take advantage and explore new possibilities offered by mobile devices to significantly ease the animation process, making it accessible to non-expert users. The proposed solution was specially oriented for Apple's iPad and developed using Unity3D engine so it seamlessly fits in the LIFEisGAME project prototype, that inspired this work - a serious game that intends to help children with Autism Spectrum Disorder (ASD) to recognize and express emotions through facial expressions in an interactive and creative way.

Agradecimentos

Começo esta lista de agradecimentos, como não poderia deixar de ser, por dirigir uma palavra de apreço ao meu orientador, Professor Doutor Augusto Sousa, pela sua disponibilidade e todo o acompanhamento prestado ao longo do trabalho mas, especialmente pelo incentivo que me deu no momento certo, e à minha co-orientadora, Professora Verónica Orvalho, por toda a ajuda e, em particular, por me motivar e conseguir ver sempre um lado positivo nas situações. Por todo o apoio que recebi, o meu muito obrigada!

De seguida, gostaria de agradecer à equipa do PIC (Porto Interactive Center), que me recebeu tão bem, e a todas as pessoas que participaram nos testes de validação da aplicação desenvolvida, pelo tempo que disponibilizaram e por todos os comentários que deixaram nos questionários e que, mais que uma ajuda, foram um incentivo para continuar.

Já num campo mais pessoal, gostaria de agradecer aos meus colegas de curso, mas em especial àqueles que se tornaram grandes amigos, por ouvirem os meus lamentos e por estarem sempre ao meu lado, com uma palavra de incentivo ou, pelo menos, com uma pequena distração que me fizesse depois voltar com mais ânimo ao trabalho.

Claramente que os agradecimentos mais fortes vão para a minha família. Agradeço aos meus tios e primas Toni, Irene, Sofia e Raquel, por serem os meus segundos pais e irmãs mas obviamente, agradeço em especial aos meus pais, que me abriram o caminho que agora termino e que estiveram sempre ao meu lado para me ajudar a enfrentar todos os obstáculos. Obrigada por todo o amor e carinho que sempre me deram mas, sobretudo, obrigada por estarem sempre aqui nos piores momentos (e houve bastantes!). Agradeço também ao meu irmão, Bé, por sempre ter sido o meu maior exemplo.

No mesmo patamar da família não poderia deixar de agracer ao meu namorado, Tiago, por ser o meu companheiro em todas as situações e por ter aguentado firmemente mesmo quando lhe "fiz a vida negra". Obrigada pelo teu amor e pela tua infinita paciência.

Ana Orvalho

"Animation can explain whatever the mind of man can conceive. This facility makes it the most versatile and explicit means of communication yet devised for quick mass appreciation. "

Walt Disney

Contents

List of Figures

List of Tables

Symbols and Abbreviations

2D	2 Dimensions
3D	3 Dimensions
API	Application Programming Interface
ASD	Autism Spectrum Disorder
AU	Action Unit
AVO	Audiovisual Object
CAD	Computer-Aided Design
CG	Computer Graphics
CLI	Command-Line Interface
CoLab	International Collaboratory for Emerging Technologies
DEA	Desordens do Espetro Autista
FACS	Facial Action Coding System
FAP	Facial Animation Parameters
FAPU	Facial Animation Parameter Unit
FCT	Fundação para a Ciência e a Tecnologia
FFD	Free-Form Deformer
FP	Feature Point
GlovePIE	Glove Programmable Input Emulator
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HD	High-Definition
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
LED	Light-Emitting Diode
LCD	Liquid Crystal Display
LIFEisGAME	LearnIng of Facial Emotions usIng Serious GAMEs
LTE	Long Term Evolution
MERL	Mitsubishi Electric Research Laboratories
MPEG	Moving Picture Experts Group
MoCap	Motion Capture
NUI	Natural User Interface
NURBS	Non Uniform Rational B-Spline
OEM	Original Equipment Manufacturer
PC	Personal Computer
ppi	Pixels per inch
RFFD	Rational Free-Form Deformer
RGB	Red–Green–Blue
RPC	Remote Procedure Call

SBIM	Sketch Based Interfaces for Modeling
SDK	Software Development Kit
SketchFACE	Sketch-based Facial Animation Collaborative Environment
TV	Television
UbiComp	Ubiquitous Computing
UI	User Interface
USB	Universal Serial Bus
UT	University of Texas
WIMP	Window, Icon, Menu, Pointer
XML	Extensible Markup Language

Chapter 1

Introduction

Freehand sketching has always been a crucial part of everyday human interaction. It has been used throughout time to communicate ideas to others and to help the thinking process, clarifying one's thoughts. As it is a natural and intuitive means of communication, sketching is a promising tool to ease the human-computer interaction paradigm. This is particularly interesting for the field of facial animation, once it can help non-expert users to easily model and animate characters. Simple free-hand strokes can replace traditional buttons, sliders and keyboard shortcuts as an alternative of input to control modeling and animation applications.

This document presents an approach for facial deformation and animation based on a hybrid sketching and dragging control system that takes advantage of the intuitive interface provided by mobile devices. The system was developed for the iPad and implemented using Unity3D engine. By simply drawing strokes on a control canvas on the iPad, a non-expert user can create facial expressions on a 3D model on-the-fly and then animate the created pose sequence. The results can be visualized both on a small section of the iPad screen or, if desired, in a much larger external display connected to the mobile device. The possibility of integrating other visualization displays helps the interaction between multiple users that is also supported by the system, creating a simple and intuitive collaborative environment for modeling and animation.

This chapter provides relevant background information on this dissertation. It presents the main challenges addressed, the motivation for the developed work and the objectives and contributions to achieve and briefly describes the proposed solution.

1.1 Motivation

Interactive devices with pen-based interfaces have become popular in recent years and tablets and touch screens are now common input technologies for many applications [8]. Sketching, as an intuitive means of communication, provides a natural alternative to the multi-tool selection paradigm of traditional interfaces, being now used in a variety of domains such as front ends for Computer-Aided Design (CAD) systems, automatic correction or understanding of diagrams for

immediate feedback in educational settings, alternative inputs for small keyboard-less devices, or gestural interfaces. In the context of facial animation, a sketch-based interface can significantly ease the manipulation of the virtual characters, opening new opportunities for non-expert users that usually take a lot of time to master the powerful but complex interfaces of traditional modeling and animation applications.

This dissertation is inspired by the LIFEisGAME project that intends to help children with Autism Spectrum Disorder (ASD) to recognize facial emotions in an interactive and creative way. ASDs are a group of development disabilities whose symptoms include problems with social interaction, communication (verbal and/or non-verbal), restricted activities and interests, both in children and adults. The severity of the symptoms, their combinations and patterns of behaviour significantly varies from patient to patient.

1.1.1 The LIFEisGAME Project

LIFEisGAME (LearnIng of Facial Emotions usIng Serious GAMEs) [9] is a project funded by Fundação para a Ciência e a Tecnologia (FCT) [10] under the program UT Austin|Portugal, also known as the International Collaboratory for Emerging Technologies (CoLab) [11], in partnership with Faculdade de Psicologia e Ciências da Educação da Universidade do Porto, University of Texas in Austin, Instituto de Telecomunicações [12] and Microsoft. It explores a new and innovative approach that uses a serious game to teach children with ASDs to recognize and express facial emotions. The main objective of the LIFEisGAME project is to deploy a low cost real time facial animation system embedded in an experimental game that enables further study of the facial emotion recognition challenges of these children and also allows to analyse if the use of virtual characters in interactive training programs can help the rehabilitation of ASDs patients.

Figure 1.1: LIFEisGAME attempts to show how it is possible to apply a serious game approach to teach people with Autism Spectrum Disorder (ASD) to recognize facial emotions.

The official presentation of the project took place on June 11^{th}, 2013 where a prototype application was shown and explained, including four game modes that correspond to different learning cycles: *Recognize Mee*, encourages children to watch a sequence of random facial expressions and recognize the emotion; *Sketch Mee*, where the user constructs a facial expression on a 3D character to match a defined emotion; *Mimic Mee*, where first an avatar mimics the players facial expressions and then the user must follow the avatar's expressions and *Play Mee*, where the player is required to perform expressions that correspond to a situation depicted in a story.

The work of this dissertation focused on a specific module of the LIFEisGAME project that uses sketch-recognition techniques to allow the player to draw facial expressions on an avatar, modifying its appearance. The core of this module consists on a facial sketching control system developed by Miranda et al. [13] that allows complex 3D deformations with just a freehand drawing on a 3D mesh or on a virtual canvas.

Based on this, a new application was created - SkethFACE (Sketch-based Facial Animation Collaborative Environment) - to allow collaborative work on the modeling and animation processes. SketchFACE explores networking concepts to allow multiple users to simultaneously deform a model in a joint effort to produce an animated sequence of poses and supports multiples devices for better content manipulation and display, increasing the immersiveness of the animation environment.

SketchFACE aims to extend the scope of the LIFEisGAME project to individual without ASDs by providing a simple and intuitive collaborative facial animation system appropriate to non-expert artists and people with no modeling background that can range from young children to multidisciplinary teams where those in charge need a simple tool to transmit their ideas to the ones who will execute them.

1.2 SketchFACE Overview

SketchFACE (Sketch-based Facial Animation Collaborative Environment) is an easy application for modeling and animation of 3D characters controlled with simple strokes on 2D canvases that represent each region of a model face. It provides a collaborative environment where multiple users can, simultaneously, deform a character, in a joint effort to create facial expressions that can be saved to a timeline and animated. It was designed to run on mobile devices with a prototype deployed for Apple's iPad.

The tablet works as the control interface to manipulate model's geometry but the results of the deformations can be seen in a much larger HD TV, if one is available and connected to the iPad via Apple TV.

This way, SketchFACE supports multiple users and multiple visualization displays, resulting a simple but immersive tool to help facial modeling and animation.

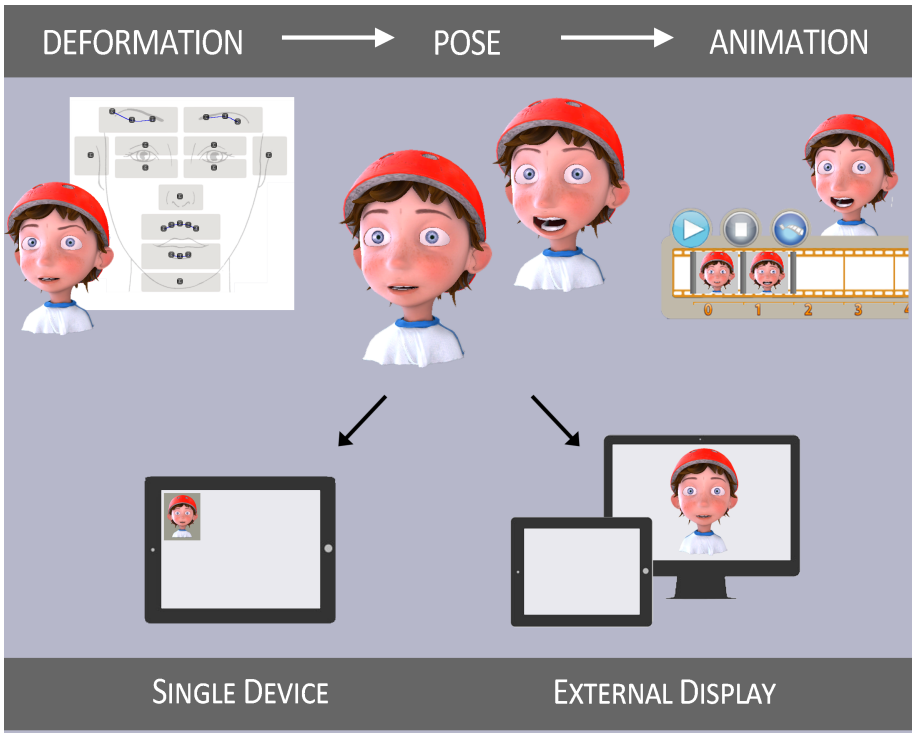

Figure 1.2: SketchFACE Pipeline.

Main Features

The application allows the user to:

- Create different expressions on a 3D character simply by dragging control points or drawing strokes on 2D canvases like sketching on a paper;
- Generate an animated clip with the expressions created and saved;
- Connect an external display to the iPad so that it becomes the control for the content shown in a much wider screen (like a TV or a computer screen). Share with other simultaneous users or with an audience the poses created or the resulting animation;
- Use it alone or team up with some friends to make the animation process more fun and interactive.

1.3 Objectives

This work aimed for the accomplishment of the following objectives:

O1. To carry out research on facial animation techniques with special focus on sketch-based animation interfaces that leads to a timeline-based facial animation application based on the facial

sketching control method developed by Miranda et al. [13];

O2. To carry out research on interactive devices technology and the new trends in user interfaces so that the final application suits the chosen implementation platform: the iPad;

O3. To explore different models to define the most adequate user interface to allow an easy and intuitive manipulation and animation of 3D characters.

O4. To deploy a functional prototype application of the system, for the iPad, that includes a standalone version, for single users, and a collaborative environment, allowing simultaneous alteration of a single model.

O5. To validate the created application with a set of usability experiments.

1.4 Document Outline

The remaining chapters of this dissertation are organized as follows:

Chapter 2: State of the Art - Presents some background content divided into two major sections: *Character Facial Animation* and *Sketching Interaction*. The first one discusses the complexity of facial animation, briefly describes the stages of a traditional animation pipeline and the major techniques used in each one. The second one presents an evolution of user interfaces and a study of emergent technologies related to interactive devices. It also details a new approach in the human-computer interaction model - sketch-based interfaces - and how it can be used to ease the modeling and animation processes.

Chapter 3: SketchFACE: Overview of the Proposed Solution - Defines the proposed solution and details all the preliminary work that led to the development of SketchFACE with special focus on the problem statement and the main system design decisions.

Chapter 4: SketchFACE: Implementation - Describes the implementation of the proposed modeling and animation system based on an hybrid method of dragging and free-hand sketching that allows collaborative work to produce facial expressions.

Chapter 4: Results and Validation - Describes the experiments conducted with users to validate the application and to reach some conclusions about the value of the mobile device based approach and of the collaborative environment.

Chapter 5: Conclusion and Future Work - Summarizes the work developed in this dissertation and presents suggestions for future work related to this topic.

Chapter 2

State of the Art

This chapter, that provides some relevant background content, is divided into two major sections. The first one, *Character Facial Animation*, begins by exposing the challenges raised by realistic facial animation. Then, it describes the traditional animation pipeline and the major techniques used in some stages and the section ends with a brief overview of the major standards used to categorize facial expressions. The *Sketching Interaction* section presents the three phases of interface evolution, focusing on natural user interfaces and addressing some emergent technologies related to interactive devices. The section ends with the description of the main stages of the sketch-based interfaces pipeline, showing how this new human-computer interaction model can be used to ease the modeling and animation processes.

2.1 Character Facial Animation

Facial expressions are essential to convey emotions to a character. But creating realistic facial movements is complex due to the intricacy of the human facial anatomy, with all its subtle textures and small wrinkles, and to people's inherent sensitivity to facial appearance. Convincing animations demand quality models, manipulated by experienced artists through appropriate control interfaces and extensive and time-consuming manual tuning.

2.1.1 Facial Animation

Facial animation began in the 1970s with the pioneer ideas of Frederick Parke that produced the first computer generated animated sequence of a human face changing expressions [14]. It remains one of the most fundamental problems in computer graphics, often separated from regular character animation due to its higher complexity. The difficulty to achieve believable animations arises mainly due to the *morphology* and *behaviour* expected from a facial model. According to Richie et al. [15], facial style can be defined as *hyper-realistic*, if it looks realistic and fantastic at the same time, *photorealistic*, designated by the authors as no-style, and *higly-stylized*, if it can range in size and shape regardless of the constraints of the physical world. The performance of each virtual character must be in tune with its appearance since it is key to achieve believability:

human characters' motion must obey the laws of physics but this may not be valid for super-heroes or imaginary creatures. The human face places a particularly challenging problem since it is an extremely complex geometric form, exhibiting numerous tiny wrinkles and subtle variations in color and texture that are essential to convey emotions. But as people are very sensitive and familiar to facial appearance and expressions, any inconsistency in the shape or movement of a virtual character, when compared to what is expected from a real human face, is easily identified. This is known as the Uncanny Valley (figure 2.1), introduced by Masahiro Mori [16, 17].

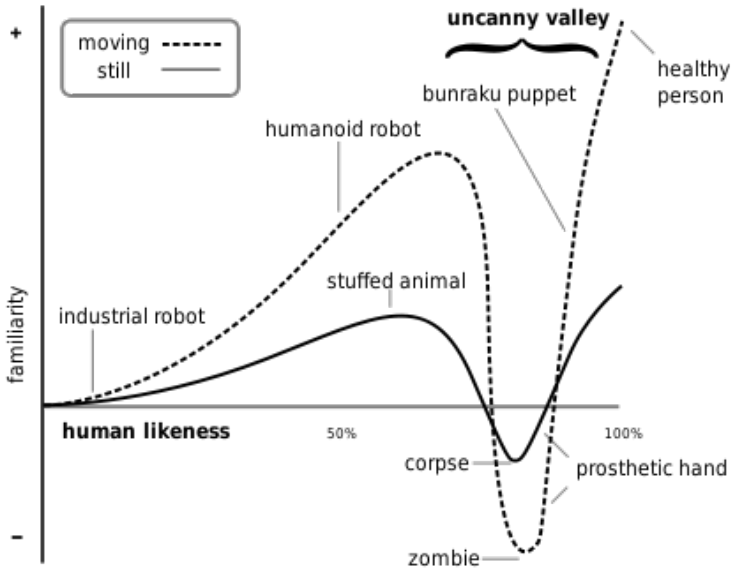

Figure 2.1: The Uncanny Valley (Translated by Karl F. MacDorman and Takashi Minato. Original graph by Dr. Masahiro Mori, 1982). The original hypothesis stated that as the appearance of an entity becomes more human-like, a human observer's emotional response becomes increasingly positive and empathic, until a certain point beyond which it becomes repulsing.

Thus, facial animation uses most methods applied to body animation but with a greater refinement in order to compensate for the higher degree of detail of the face.

Within the entertainment industry, applications can be divided into *off-line systems* and *real-time systems*. Off-line systems require high realism and accuracy to reinforce the audience attention and are mainly used in films, visual effects or TV broadcasting. Real-time systems, also known as interactive systems, such as dialog-based interfaces, virtual reality and videogames, require real-time animations, consequently, a compromise between realism and fast computation.

2.1.1.1 Facial Animation Pipeline

A traditional production environment is divided into the following stages: pre-production, modeling, rigging and animation (figure 2.2) [18].

In the pre-production stage, the concept and requirements of the character are defined in terms of shape, visual style and movement.

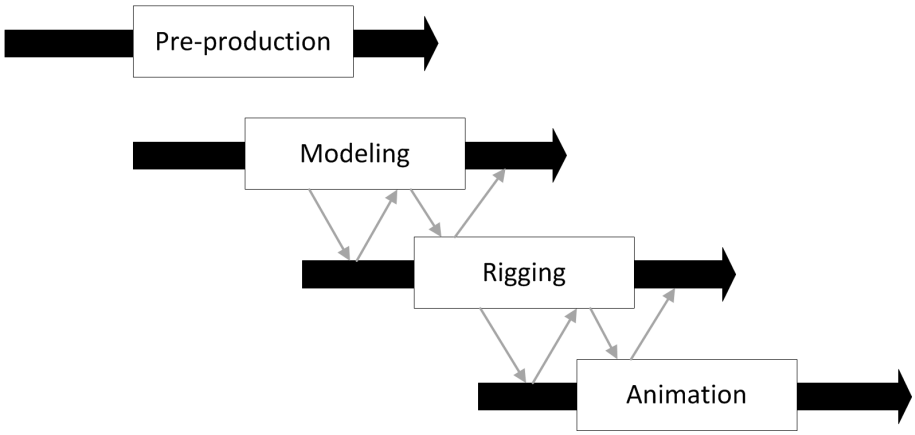

Figure 2.2: Different stages of a traditional animation pipeline

During the modeling stage the geometry of the model is created based on the requirements defined in the previous stage. The techniques to produce quality facial models can be divided into two categories: *generic model individualization*, that generates a facial model for a specific individual through the deformation of a generic model, and *example-based face modeling*, that consists on creating a face model with desired facial features through the linear combinations of an existing face model collection [19].

The next step of the pipeline consists on creating a control structure that allows the manipulation of the model like a virtual puppet [18], - the *rig* - after which the character is ready to be animated.

The responsible for rigging a character, usually known as rigger, needs to understand the behaviours expected for the character and interact with modelers and animators in order to provide an efficient and intuitive interface to control it. Riggers and modelers must reach a balance between putting enough detail into the model, to get the desired control, and not adding too much geometry that leads to cumbersome and slow definition of facial deformers [18].

It is also common that animators ask for new controls after the rig is created, to achieve better deformations or alter its behaviour when does not perform as desired, which usually resets the rigging process and, consequently, delays the production process. Rigging a character becomes an iterative process and a serious bottleneck in a CG production pipeline [20].

Therefore, modeling, rigging and animation stages run in parallel.

The following sections will focus on the rigging and animation stages of the production pipeline, presenting the most relevant techniques used in each one.

2.1.1.2 Facial Rigging

Rigging is a technique for creating a control structure that enables the artist to produce motion on a 3D character by manipulating a 3D model like a puppet [20]. According to McLaughlin and

Sumida, character rigging can be defined as the system engineering process that allows surface deformation, mimicking the effect of bones and muscles moving skin on a biological creature [21].

Every rig must provide easy animation controls that work as expected and should allow small modifications to correct undesired behaviours. In order to avoid strange or impossible moves that do not follow the requirements predefined for a specific character, the rig should include a set of constraints [22].

The rig control points can be attached to selected areas of the model and affect the correspondent area accordingly to the geometric operation (translation, rotation and scaling) applied to them. The rig determines the quality and the number of potential animations. More control points allow smoother animations but also lead to a system that is more complex to animate and maintain. As face animation must preserve the subtleties of facial expressions, a high number of joints is required to achieve realistic results.

The most common approaches to create a facial rig are based on *blend shapes, bone-driven* techniques or a combination of both. To complement these approaches, an additional layer of deformation can be added to a model in areas where neither bones nor shapes successfully reproduce facial features such as wrinkles. These deformers can be divided in two groups: geometrically-based and physically-based methods.

Blend Shapes

Blend shapes, or shape interpolation, is the most intuitive and commonly used technique in facial animation: a set of key facial poses, called *shapes*, are interpolated to generate the character's animation. A blend shape model is the linear weighted sum of a number of topologically conforming shape primitives [1]. Varying the weights of this linear combination allows the representation of a full range of expressions with little computation. However, to express a significant range of highly detailed expressions usually implies the creation of large libraries of blend shapes which can be very time-consuming. For example, in the film The Lord of the Rings: The Two Towers, the rig of the character Gollum required 675 blend shapes [23]. Furthermore, if the topology of the model needs to be changed, all the shapes must be redone. Shapes can be created by deforming a base mesh into the desired canonical expressions or can be directly scanned from a real actor or a clay model [24].

Bones

In bone-driven animation, a character is represented in two parts: a 3D surface that represents the visual features of the model, called *skin* or *mesh*, and a highly articulated set of bones (called the *skeleton* or *rig*) used to deform and animate the mesh. The binding process of the skeleton to the mesh, called skinning, takes into account how each bone influences each part of the surface during deformation. Each vertex is only animated by the bones around it according to a defined weight value so, careful planning is required in the rigging process of each model [25]. A full weight value makes the surface point move exactly like its associated bone. A low weight value only causes a partial influence on the surface point when the associated bone moves. This approach requires

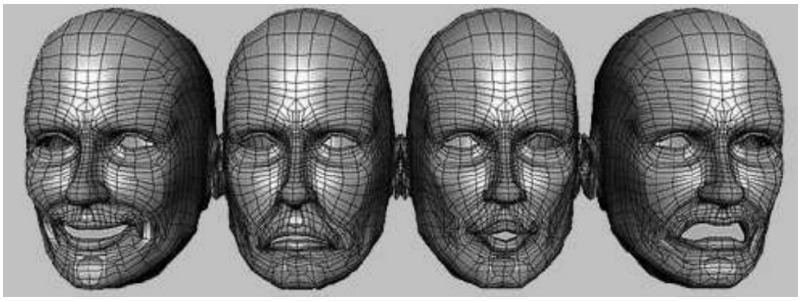

Figure 2.3: Blendshapes of four basic expressions: happy, sad, surprise and angry (Copyright 2004 New Riders Publishing).

more preparation in order to obtain the desired results, specially regarding the correct weight definition, but enables smoother movements comparing to blend shapes and needs no further work when the topology of the character is modified. In videogame productions, bone-driven techniques are often combined with motion capture based on performance of actors with motion sensors placed on the face, each one representing a bone of the rig.

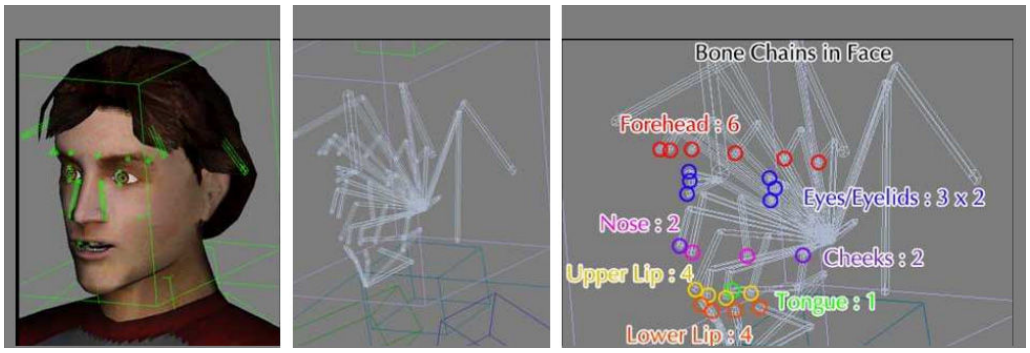

Figure 2.4: A bone-driven rig based on a highly articulated facial skeleton structure (Copyright 2001-2007 Epic Games).

Combined techniques

Exploring hybrid techniques is a common approach in the entertainment industry to achieve better results and minimize time and computational effort. Interpolation between key images used in blend shapes techniques provides a direct and intuitive method for specifying the actions of the character. However, it also leads to incomplete control of motion dynamics translated in smoothness or continuity problems as the weighting factor that controls the amount of change from one frame to the next is a single-valued function of time that is applied to an entire key shape, providing no "spatial weighting" [26]. On the other side, skeletons are simple structures composed of a few points, which provides a high level of interaction and have the advantage of being very compatible with blend shapes techniques. Combining blend shapes with a skeletal approach provides the rig with flexibility and smoothness of a bone-driven system and the expressiveness of

blend shapes [27]. Moreover, skeleton control can be applied selectively to parts of the model that require enhancement.

Geometrically-based

Geometric deformation consists on using an easier and simpler control interface to manipulate a deformable model. Basically, uses a simpler object to modify a more complex one. One of the most commonly used geometric deformation methods is the Free-Form Deformer (FFD), first introduced by Sederberg and Parry [28]. A FFD uses a flexible control box containing a 3D grid of points (figure 2.5) that encompasses the model to be deformed. The manipulation of these points deforms the control box and, at the same time, the embedded object.

Figure 2.5: Free-Form Deformation applied to a spheric surface: controlling box and embedded object; left: neutral position; right: object deformation [1].

Several approaches to this method were presented, varying the shape of the control lattice [29] or the degree of control over the embedded model. For example, Kalra et al. [30] extended the concept of FFDs to Rational Free-Form Deformers (RFFD), in which different weights can be assigned to each point of the control structure allowing better control over the geometry deformation. The authors also proposed a division of the face in regions of interest, allowing a more accurate and independent control of each one and simulating the muscle action on the skin surface of the human face.

Singh and Fiume proposed an alternative approach not directly related to FFDs but that can be used to emulate them, replacing the lattice for a Non Uniform Rational B-Spline (NURBS) curve [31]. A NURBS is a mathematical representation of 3D geometry that can accurately describe any shape from a simple 2D line or curve to the most complex 3D free-form surface. These parametric NURBS curves can be used as wires to control each part of a model, as a rig. By manipulating the parameters, different poses can be created.

Physically-based

Physically-based methods simulate the elastic properties of facial skin and muscles to create expressions and animations, as well as to build facial models. But replicating the behaviour of human tissues is very intricate. The search for realistic results led to two dominant approaches used in physically-based models: mass-springs and finite elements. Depending on the intended simulation, these two techniques can be combined. Mass-spring methods model the skin, and sometimes muscle and bones, as a number of point masses connected by springs in a lattice structure, like a

cloth. Finite elements methods break the continuous system into a regular discrete representation with a finite number of elements using, for example, tetrahedrons. This last technique is more sophisticated, physically accurate and stable than the first, making it more suitable for modelling continuous materials like soft tissue, but is also computationally far more expensive [32].

Platt and Bladler presented the first physically-based facial animation model that used a mass-spring system to simulate muscle fibres [33]. Their work used the Facial Action Coding System (FACS) to determine which muscles to activate in the underlying model (see section 2.1.1.4).

Rig Control Interface

The manipulation of the rig deforms the geometry of a 3D character's model, producing movement. To allow this manipulation, an optional layer of control can be defined - the rig's user interface (UI). There are many approaches to handle the UI for rigging but two major categories can be identified: *window-based* and *viewport-based*, which can also be combined.

Window-based UIs provide direct input of values through traditional sliders, buttons or boxes located in separate windows. Villagrasa and Susin [2] built a slider-based UI based on FACS (see section 2.1.1.4). Bredow et al. [3] configured Maya's channel box to display multiple categorized columns of attributes to animate the characters of *Surf's Up* (Sony Pictures Animation, 2007).

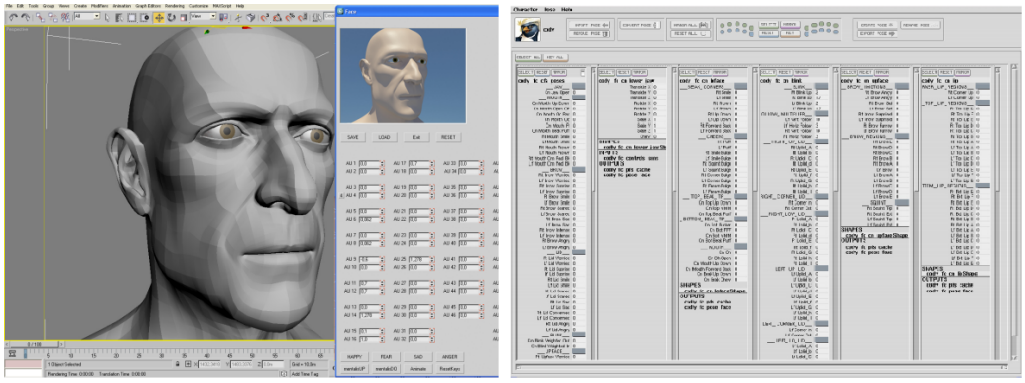

Figure 2.6: Two window-based UI. Left: slider-based UI based on FACS [2]; Right: interface with multiple columns of attributes [3].

Viewport-based UIs use a set of 2D or 3D controls to manipulate the rig that are included in the 3D space where the model is located. An illustrative example of this approach is the one proposed by Jason Osipa, that provided a high level viewport to edit the model and the animations, which only allowed to manipulate four attributes of the rig elements by a bi-dimensional set of controls constrained to a square [34].

The following section discusses the most relevant methods related to facial animation.

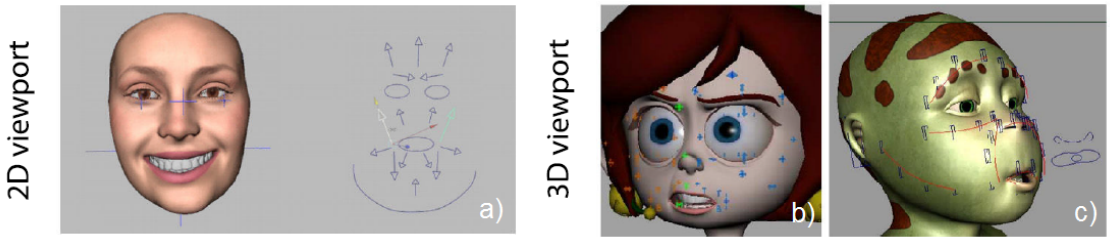

Figure 2.7: Three examples of viewport-based UI. a) 2D controls by Alexander et al. [4]; b) 3D controls by Komorowski et al. [5]; c) 3D controls by Grubb [6]

2.1.1.3 Animation Techniques

Once a model's geometry is defined and the control structure that allows its deformation is created, the character can be animated. Many different techniques have been developed for this process but, in general three major approaches can be identified: *keyframe interpolation*, *motion capture* and *procedural animation*. Since they are rarely used individually, these approaches can be complemented by *physically-based* and *geometrically-based* techniques.

Keyframe Animation

Keyframe animation is the easiest and oldest completely geometric technique that offers an intuitive approach to facial animation: several complete face models with the same topology, called keyframes, are created for a given set of points in time and the in-between frames are obtained by interpolation of these keyframes. Realistic animation requires a good number of keyframes. If not enough are used, the in-betweens will be too unpredictable, the path of action will usually be incorrect and objects may intersect one another, demanding for exhaustive reworking of intermediate poses [35].

The simplest case of keyframe animation to be mentioned corresponds to an interpolation between two keyframes at different positions in time (figure 2.8).

Figure 2.8: Two different poses and the resulting interpolation. Left: Neutral pose, Right: "A" mouth shape, Middle: Interpolated shape. [1]

Due to its simplicity, linear interpolation is commonly used [36], obeying the following formula:

$$pose_{interpolated}(t) = (1-t) \times pose_1 + t \times pose_2 \qquad 0 \leq t \leq 1 \tag{2.1}$$

If t is 0, the current frame will be the same as the first keyframe. If t is 1, then the current frame will match the second keyframe. Different values lead to a weighted combination of both keyframes. Other types of interpolations can also be used: a cosine interpolation function can provide acceleration and deceleration effects at the beginning and end of an animation [37] and bilinear interpolation generates a greater variety of facial expressions when four keyframes are used, instead of two [38].

Although quick and easy, interpolation falls short to achieve smooth and realistic results so it is usually paired with other techniques, such as performance-driven methods [39].

Motion Capture Animation

Motion capture (MoCap) animation, also known as performance-driven animation, emerged from the difficulty of achieving life-like characters in facial animation. It is a data-driven technique since the data obtained from external sources is mapped onto a model to create animated sequences: the majority of these methods trace facial markers placed on a performer and extract the 2D or 3D positions of these markers to animate a 3D face mesh. Accurate tracking of these points is important to maintain a consistent and realistic quality of animation [1].

Performance-driven animation has been used on movies such as *The Polar Express* (2004) where it allowed an actor such as Tom Hanks to drive the facial expressions of several different characters. During the shoots, 80 markers were used for the body and 152 markers were used for the face [40].

Advantages of MoCap systems include the increased speed over manually crafted animations and the potential of producing more realistic facial motion. But most marker-based motion MoCap systems use between 30-160 marker on the face which work reasonably well for capturing the motion of rigid objects but is not very effective at capturing subtle movements of the face [22]. Consequently, the animator needs to spend a significant amount of time tweaking the animation to fit the desired results. This weakness encouraged the development of new markless systems [41] and facial feature tracking from video using complex models.

Procedural Animation

Procedural animation is used to automatically generate animation in real-time to allow more diverse series of actions that could otherwise be created using predefined animations. With this approach, objects are animated based on physical rules, often of the real world, expressed by mathematical equations. The animator specifies these rules and the initial conditions and runs the simulation to see the results. Procedural animation requires significantly more planning time than non-procedural approaches but has the advantage of easing the build and tuning stages: by changing the input parameters it is fast and easy to create new results or to modify previous work.

Several procedural animation systems have been developed to provide a range of useful character behaviours [42, 43]. Most of them aiming to maximize physical realism in highly dynamic tasks such as tumbling. However, in terms of behavioural movements, procedural animation for the face is not a very explored area.

2.1.1.4 Facial Standardization

Broadly speaking, within the field of computer facial animation, two kinds of people can be identified: the researchers and the artists. Researchers are mostly interested in the more technical aspects of the problems, trying to track facial features in real time in unconstrained video without markers or studying intricate methods to realistic animate anatomically correct models according to physic laws. On the other side, artists are concerned with more immediate and practical tasks of producing high quality facial animations, especially for the entertainment industry, so they use the best methods possible provided that they are compatible with the software they already master. Thus, most of the facial animation methods described in scientific articles never reach the pipelines of major movies or TV productions. But even within the research community, different groups often face problems of system interoperability because they do not use the same parameters to detect, control or animate facial movements [44]. In order to bridge these gaps, significant effort has been put on describing the face with a small set of control parameters instead of defining its complete geometry. This parameterization eases the processes of controlling facial movements, acquiring information from video analysis, reconstructing a head or transferring animations between different models.

Research has shown that an ideal parameterization does not exist because it is difficult to satisfy all user demands for a broad range of facial applications. Parke developed the first facial parametric model that allowed direct creation of facial deformation by defining *ad hoc* parameters or by deriving parameters from the structure and anatomy of the face [38]. Since then, other approaches for facial standardization have been studied.

The following sections present two standards that have been used to categorize facial expressions.

FACS - Facial Action Coding System

The Facial Action Coding System (FACS) is the most widely and versatile method for measuring and describing facial behaviours, having become a standard to categorize the physical expressions of emotions. It was originally published by Paul Eckman and Wallace Friesen in 1978 [45] and updated in 2002, with large contributions from Joseph Hager [46]. They determined how the contraction of each facial muscle, singly and in combination with other muscles, changes the appearance of the face by examining videotapes, studying anatomy and palpating their faces. FACS parameterizes facial expressions in terms of Action Units (AU) that are the fundamental actions of individual muscles or groups of muscles like raising left eyebrow. There are 46 AUs that represent

contractions or relaxation of one or more muscles. The scores for a facial expression consist of the list of AUs that produced it.

AU	FACS Name	AU	FACS Name	AU	FACS Name
1	Inner Brow Raiser	12	Lid Corner Puller	2	Outer Brow Raiser
14	Dimpler	4	Brow Lower	15	Lip Corner Depressor
5	Upper Lid Raiser	16	Lower Lip Depressor	6	Check Raiser
17	Chin Raiser	9	Nose Wrinkler	20	Lip Stretcher
23	Lip Tightener	10	Upper Lid Raiser	26	Jaw Drop

Basic Expressions	Involved Action Units
Surprise	AU1, 2, 5, 15, 16, 20, 26
Fear	AU1, 2, 4, 5, 15, 20, 26
Anger	AU2, 4, 7, 9, 10, 20, 26
Happiness	AU1, 6, 12, 14
Sadness	AU1, 4, 15, 23

Figure 2.9: FACS. Upper row: Sample single facial AUs; Lower row: Sets of AUs for basic expressions [1]

Despite its popularity and simplicity, FACS has two major weaknesses [47]: first, the AUs are purely local spatial patterns but real facial motion is rarely completely localized and second, FACS does not offer temporal components to describe the motion. Other limitations of this system include inability to describe fine eye and lip motions, and the inability to describe the co-articulation effects found most commonly in speech.

MPEG-4 Facial Animation

MPEG-4 is an object-based multimedia compression standard that allows encoding of different audiovisual objects (AVO) in the scene independently, developed by the ISO/IEC Moving Picture Experts Group (MPEG). Its initial objective was to achieve low bit-rate video communications but its scope was later expanded to a much broader multimedia context including images, text, graphics, 3D scenes, animation and synthetic audio.

The MPEG-4 Facial Animation standard [44] specifies a face model in its neutral state and a number of Feature Points (FP) that provide spatial reference to specific positions on a human face such as major muscles and bones. It also defines a set of Facial Animation Parameters (FAP), each corresponding to a particular facial action deforming the neutral face. Facial animation sequences are generated by deforming the neutral face model according to some specific FAP values at each time instant.

The standard defines 84 FPs, 66 low-level FAPs and 2 high-level FAPs, visemes and expressions. Viseme is the visual counterpart of phonemes in speech while facial expressions consist of a set of 6 basic emotions: anger, joy, sadness, surprise, disgust and fear. The low-level FAPs are based on the study of minimal facial actions and are closely related to muscle actions, precisely specifying how much a given FPs should move. All low-level FAPs are expressed in terms of the Face Animation Parameter Units (FAPUs) which are fractions of key facial features, such as the

distance between the eyes. FAPUs scale FAPs for fitting any face model, allowing their interpretation in a consistent way. By coding a face model using FPs and FAPUs, developers can exchange face models without concerns about calibrations [7].

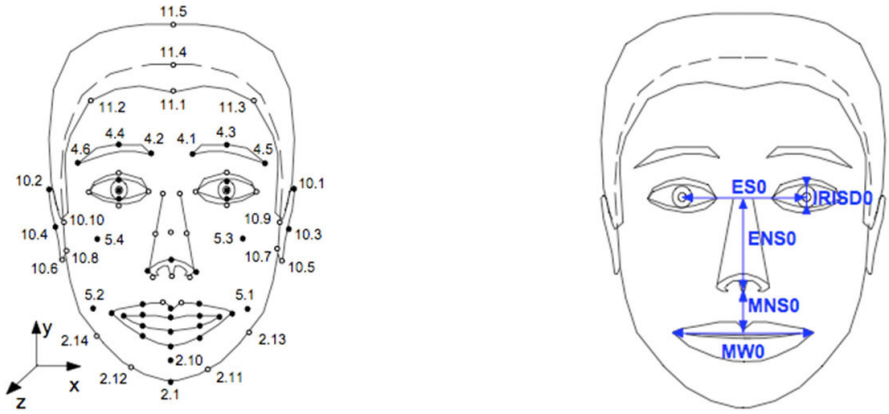

Figure 2.10: MPEG-4 Facial Animation. Left: Some Feature Points (FP); Right: A face model in its neutral state and the FPs used to define FAPUs. Fractions of distances between the marked FPs are used to define FAPU [7].

2.2 Sketching Interaction

The way humans interact with computers has evolved at the same pace as the machines themselves. Today, the barriers between users and devices are fading away with the development of new types of interfaces. As touch-screen devices become more common, this technology can provide an accessible and natural interface for sketches - rapidly executed freehand drawings - to be used in the modeling and animation processes as well as in collaborative systems.

2.2.1 Interactive Devices

Human-Computer Interaction (HCI), originally known as man-machine interaction, is a term known since the early 1980s [48]. It examines the importance of usability and user-oriented interface design meaning that if focus on improving interaction between users and computing devices according to the needs and capabilities of both. Since the first digital computers were programmed, using mechanical switches and plug boards, the ways in which people interact with computers have evolved significantly and as more and more uses for technology came into play, more and more types of interfaces were created to help bridge the barrier between man and computer. A common perspective is that interfaces have passed through three loosely defined phases: *command-line interfaces* (CLI); *graphical user interfaces* (GUI) and, more recently, *natural user interfaces* (NUI).

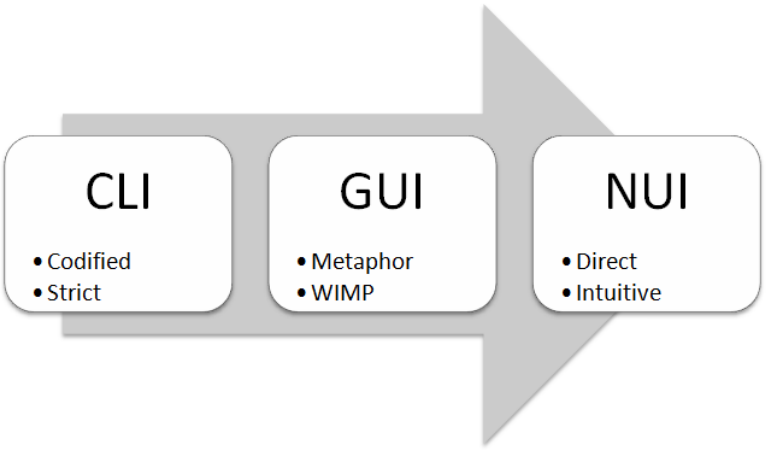

Figure 2.11: The evolution of interfaces.

Command-line interface was the first means of human-computer interaction and the more effective way to control computing devices around the 1960s. It consisted in a kind of interactive dialogue in the form of successive lines of text (command lines) that were understood by both users and computers. The interface was usually implemented with a command line shell, which is a program that accepts commands as text input and converts them to appropriate system functions. Today, CLIs are less used by casual users but often preferred by advanced ones, since they generally provide a more concise and powerful means to control a program or operating system.

With the development of ultra large scale integrated circuits, high-resolution displays and the appearance of the mouse, interfaces turned to a graphical approach. GUIs are based on metaphors, like the desktop metaphor (so called because windows are allowed to overlap, like papers on top of a desk) and rely on a known set of user interface elements, commonly referred as WIMP (Window, Icon, Menu, Pointer) [49].

2.2.1.1 Natural User Interfaces

Today, a new trend is gaining strength: NUI is an emerging paradigm shift that is reducing even more the barriers between users and machines. The term is used to refer a user interface that is invisible or becomes invisible to its users with successive learned interactions. The word "natural" is used because NUIs aim to enable users to interact with computers in the same way as they interact with the world. They rely on the ability of a user to perform relatively natural movements or gestures that they quickly discover to control the computer applications or manipulate the on-screen content. Thus, novice users quickly progress to experts. NUIs take advantage of the power of a much wider range of communication modalities, focusing on human abilities such as touch, speech, handwriting, eye-gazing, motion and higher cognitive functions such as expression, creativity, exploration, as well as combinations of them, forming multimodal interfaces.

But the design and development of the technology that support new NUIs is both conceptually and practically very challenging and may require both novel software and hardware to allow input from multiple and varied sources [50]. Traditional development environments, such as Microsoft Visual Studio/.NET, Adobe Flash or Java, fall short of supporting uncommon input devices as well as handling multi-user applications, for multi-touch interaction and collaborative work. To overcome these issues, over the last few years a broad variety of heterogeneous and very specialized toolkits and frameworks have appeared like Microsoft Surface SDK [51], NUIGroup Touchlib [52], a library for creating multi-touch interaction surfaces or GlovePIE [53], that originally started as a system for emulating joystick and keyboard input using a virtual reality glove peripheral, but now supports many input devices. Few development environments that address the new requirements are available, supporting novel input devices such as physical turntables, mixing desks, multi-touch surfaces and simple vision tracking. Two examples are MAX/MSP [54] and vvvv [55], which are graphical development environments for music and video synthesis that are widely used by artist to create interactive multimedia installations.

The following sections present examples of devices commonly referred as having NUIs.

2.2.1.2 Multi-touch interfaces

Multi-touch devices consist of a sensing surface, like a trackpad or a touch-screen, as well as software that recognizes two or more points of contact with the surface. This plural-point awareness is often used to implement functionalities such as pinch to zoom or activating predefined programs. Multi-touch technologies have a long history. The first documented multi-touch system was developed in 1982 by the University of Toronto's Input Research Group [56]. It consisted of a frosted-glass panel with particular optical properties so that finger pressure produced variable size black spots on an otherwise white background. Using a camera, simple image processing allowed multi-touch input.

In recent years the market witnessed a proliferation of multiple finger tracking products [57], including many tablets, smartphones and digital tables. Bill Buxton presented a good overview of the evolution of the multi-touch technology [58] since its early beginnings. The following sections present some of the most relevant new trends and devices.

Multi-touch Tables

Multi-touch digital tables are tabletop displays that present the characteristics of multi-touch technology: a touch-screen and software to analyse the contact points. It is an innovative user-friendly technology offered in nearly any shape or size to suit any requirement. Depending on its specifications, digital tables may allow users to interact with multimedia content the same way they have interacted with physical objects using their hands, normal gestures or by putting real-world objects on the table.

DiamondTouch

DiamondTouch [59] is a multi-user touch technology for tabletop front-projected displays that supports small group collaboration, originally developed at Mitsubishi Electric Research Laboratories (MERL) in 2001 [60] and later licensed to Circle Twelve Inc., in 2008. It enables several different people to use the same touch-surface simultaneously without interfering with each other, or being affected by foreign objects left on the surface but its most innovative feature is the ability to identify which person is touching where. By distinguishing between different users, the system can track a person's input and behave appropriately, controlling their access to certain functions.

SMART Table

The SMART Table is the first multi-touch, multi-user interactive learning center designed to stimulate collaboration among primary students. It works with both Mac and Windows operating systems and is easy-to-clean, scratch-resistant and highly durable in order to suit the needs of its young users. The SMART Table can be used together with other SMART hardware and comes with an initial set of eight learning applications but many others are available for download. The well-known 230i [61] model, with a blue top child appealing look, presented a 27 inches multi-touch display supporting up to 6 users at one time. The new SMART Table 442i [62] features a 42 inches (106.68 cm) surface with high-definition 1080p LCD display, supporting up to 40 simultaneous touches which enable up to eight students to interact simultaneously and actively collaborate to achieve shared learning goals.

Microsoft PixelSense

Microsoft PixelSense [63], formerly called Microsoft Surface, is an interactive surface computing platform that recognizes fingers, hands and objects placed on the screen to create a natural user interface. It allows one or more people to use touch and real world objects and share digital content at the same time.

Microsoft Surface 1.0, the first version of PixelSense, was announced on May, 2007 and could recognize 52 simultaneous multi-touch points of contact in a 30 inches (76 cm) 4:3 rear projection display (1024x768). Sales of Microsoft Surface 1.0 were discontinued in 2011 in anticipation of the release of the Samsung SUR40 for Microsoft Surface and the Microsoft Surface 2.0 software platform. The current version of PixelSense, the Samsung SUR40 for Microsoft Surface was announced in 2011, and presents a 40 inches (102 cm) 16:9 LED backlit LCD display (1920x1080) with integrated PC and PixelSense technology.

Tablets

A tablet computer, or simply tablet, is a one-piece mobile computer, mainly controlled by touchscreen via finger gestures or a virtual keyboard, removing the need for physical input hardware components. The first commercial tablets appeared at the end of the 20th century and today many models with different sizes and features are available on the market. Tablets are lightweight and

easier to carry than laptops while offer many of the same Web browsing capabilities. They are usually used for consuming multimedia content, like movies, music and books, rather than for creating content.

Apple iPad

The iPad is a line of tablet computers designed and marketed by Apple first released on April, 2010. The wide range of capabilities and increased usability, battery life, simplicity and overall quality of the first model, in comparison with competitor devices, earned the iPad positive reviews which defined a new standard, revolutionizing the tablet industry. However, some aspects, such as the lack of support for the Adobe Flash format were criticized. All devices run on Apple's iOS operating system, have built-in Wi-Fi and, some models present cellular connectivity up to LTE. The most recent models, the new iPad and the iPad Mini were released on November, 2012. The new iPad [64] has a 9.7 inch LED-backlit multi-touch retina display with 2048x1536 resolution at 264 pixels per inch (ppi).

Microsoft Surface

Microsoft Surface was the first name of the interactive surface computing platform now known as Microsoft PixelSense. Today, this name is associated to a series of tablets designed and marketed by Microsoft [65]. The Surface debuted in two models, marketed as Surface and Surface Pro: the first one with Windows RT (a special Microsoft Windows operating system designed to run on mobile devices utilizing the ARM architecture) and the other with Windows 8. Both tablets have high-definition 10.6 inches (27 cm) screens with an anti-fingerprint coating and 16:9 aspect ratio but have different resolutions: 1366x768 pixels for the Windows RT model and 1920x1080 pixels for the Pro model. One of the most useful features of Micrsoft Surface is the built in kickstand on the back of the unit that enables the device to maintain an upright position and become hands-free.

Google Nexus

Google Nexus is a line of mobile devices using the Android operative system produced by Google along with an original equipment manufacturer (OEM) partner that, today, includes smartphones and tablets (it also included a media-streaming entertainment device, Nexus Q, that was unofficial dropped due to bad reviews). The first tablet of the series, the Nexus 7 [66], developed in conjunction with Asus was unveiled in June, 2012 and shipping started the following month, being the first device to run Android version 4.1, nicknamed "Jelly Bean". Even though this is one of the smaller tablets on the market, it includes an HD touch-screen 7 inch (18 cm) display with 1280x800 pixel resolution. The second tablet of the series, the Nexus 10 [67], was manufactured by Samsung and first released in November, 2012. It runs Android 4.2 ("Jelly Bean") operative system and features a 10.1 inch display with 16:10 aspect ratio and 2560x1600 pixel resolution (300ppi), which in 2012, made it the world's highest resolution tablet display.

2.2.1.3 Spatial Interaction

Multi-touch technology can enable natural user interfaces. However, most UI toolkits used to construct interfaces with such technology are traditional GUIs that fail to achieve the "transparency" desired for NUIs: our attention is projected to the multi-touch screen that remains distinguishable from the background. This idea of integrating computers seamlessly into the world is not new. In 1991, Mark Weiser, of Xerox PARC, published an article that outlined a vision of the next generation of computation where he described a model of Ubiquitous Computing, or UbiComp, where technologies "weave themselves into the fabric of everyday life until they are indistinguishable from it" [68]. At the time, there where no appropriate display systems that could work with the full diversity of input and output forms required for this UbiComp approach. Today, we are closer than ever to achieve it, with an emerging type of user interfaces that allow users to interact with computing devices in entirely new ways, such as through the motion of objects and bodies, demoting multi-touch technology to an old draft of natural user interfaces.

Kinect

Kinect [69] is a motion sensing input device by Microsoft for the Xbox 360 video game console and Windows PCs. It was first announced in June, 2010 under the code name "Project Natal" and released in North America on November of the same year. Kinect competed with the Wii Remote Plus and PlayStation Move motion sensor controllers but had a major advantage: this webcam-style add-on peripheral enabled users to control and interact with the Xbox 360 without the need to actually touch a game controller, using gestures and spoken commands. The device features an RGB camera, depth sensor and multi-array microphone that allow full-body 3D motion capture, facial recognition and voice recognition capabilities, changing the way people play games, watch TV and listen to music. On June, 2011 Microsoft released the Kinect SDK, allowing developers to write Kinect-based applications in C++, C# or Visual Basic .NET.

Leap Motion

Leap Motion [70] is a breakthrough technology focused on bringing motion control to the desktop through a small USB peripheral. The inspiration for this new technology came to its creators from the frustration surrounding 3D modeling using a mouse and keyboard versus the simplicity of molding clay in the real world. The Leap Motion controller is designed to rest on a desk in front of a monitor, creating an invisible 3D roughly 1.2 metre-square interaction space inside which the device is advertised to track hands and fingers as well as tools such as pens, pencils, and chopsticks with very high accuracy. Like Microsoft's Kinect, the peripheral tracks human body gestures, and translates this movement into corresponding motions on a video display. According to David Holz and Michael Buckwald, co-founders of the startup Leap Motion, its input device is accurate to within 1/100 of a millimetre and 200 times more sensitive than existing motion-sensing technologies such as Microsoft's

Kinect. The release of the Leap Motion controller was postponed several times but devices are now in the process of shipping to consumers.

2.2.2 Sketch-based Interfaces Pipeline

User interfaces of modeling systems, such as Maya or Blender, have traditionally followed the WIMP (Window, Icon, Menu, Pointer) paradigm [71]. But using these powerful applications can be very difficult for a non-expert user that may need to explore a considerable amount of menus and controls before executing a specific task. Significant learning time and effort is required in order to create complex models and to memorize keyboard shortcuts. In order to simplify this interaction model, recent research in modeling interfaces led to a new trend known as sketch-based interfaces for modeling (SBIM). The idea is to automate the processes of sketch recognition in order for sketching to be an effective means of input in computer modeling, replacing the traditional buttons and menus. Sketching on paper has often been used in the early prototyping stages of the design before its conversion into a 3D model. Automating or assisting this translation could significantly reduce the time and effort needed to create complex models that usually turn out to be a bottleneck in production pipelines. As sketching is natural and intuitive for humans that can imbue so much meaning into a 2D drawing, SBIM can make 3D modeling systems accessible to novice users. The human visual system interprets sketches with little effort, even when they are not faithful representations of real-world objects. But getting a computer to mimic this ability is a very difficult task. The main challenge of SBIM is to interpret the meaning of the input stroke, understanding the user's intention, in order to display the correct result. Based on Olsen et al. [71], the pipeline of a sketch-based system is summarized in figure 2.12. The first step is to obtain a sketch from the user (*Sketch Acquisition*), followed by a filtering stage to clean and transform the sketch (*Sketch Filtering*). The process ends with the extraction of meaning from the sketch (*Sketch Interpretation*).

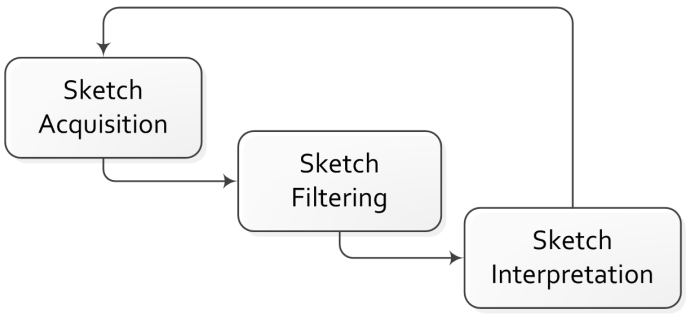

Figure 2.12: The SBIM pipeline.

2.2.2.1 Sketch Acquisition

The process of sketch recognition starts with the acquisition of a sketch from the user. This is done through a sketch-based input device that ideally mimics, as close as possible, the feel

of freehand drawing on paper in order to exploit the user's ability to draw. Although the most common input device is the standard mouse, devices in which the display and the input device are coupled (such as tablet displays) enable a more natural interaction. Despite the degree of immersion provided by the chosen sketch input device, it must, at the bare minimum provide positional information in some 2D coordinate system, usually window coordinates. The sampling rate varies among the devices but in all, the sampled positions represent a linear approximation of continuous movements, varying the space between them according to the drawing speed. The space between samples tends to be smaller in parts drawn more carefully such as corners so this fact can be exploited to identify important parts. [72, 73, 74].

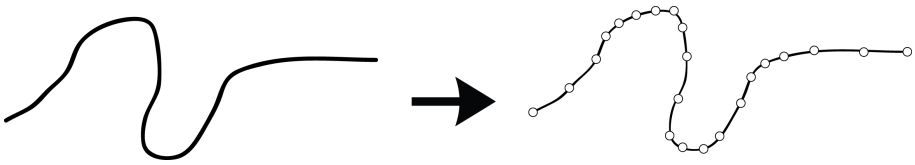

Figure 2.13: The input stroke (left) is acquired as a sequence of point samples spaced irregularly according to drawing speed (right).

A sketch is a set of one or more strokes which correspond to time-ordered sequences of points $S = \{p_1, p_2, ..., p_n\}$ whose beginning and end is defined by a mouse or pen down and up events, respectively. Each point p_i, contains a 2D coordinate and a timestamp: $p_i = [x_i, y_i, t_i]$. Depending on the target application and the available hardware, this basic information can be extended by additional data such as pressure or pen orientation.

2.2.2.2 Sketch Filtering

The filtering stage is important to remove noisy or erroneous samples from the input before attempting to interpret the sketch. Sezgin and Davis [75] identify two main sources of noise: *user* and *device error*. User errors result from poor drawing skills, slight jitter in a user's hand or difficulties in handling the input device. Device errors consist of "digitalization noise" caused by spatial and temporal quantization of the input by the mechanical hardware used and vary from device to device. As a result from this interferences, the input to a sketch-based system is generally considered to be an imperfect representation of user intention, being filtered before interpretation. Different sample rates of the input devices and variations in drawing speed contribute to unevenly spaced samples in the raw input data. *Resampling* allows the reduction of the noise in an input stroke by regularly spacing the samples. This can be done on-the-fly, by discarding or interpolating samples within a threshold distance, or after the stroke is finished. *Polyline* (or polygon) approximation is an extreme case of resampling that reduces the complexity of a stroke to just a few samples. After resampling, a sketch still contains a large number of sample points with little meaning so it is common to fit the sketch to an equivalent representation. *Fitting* simplifies the input data and the future comparison operations. *Curve fitting* is a simplification approach that requires significant computation but produces fewer errors than polyline approximation. Another

option is least-squares polynomial fitting but the most common approach is to use parametric curves like Bézier and B-spline curves. Fitting is more suitable for applications where precision is desirable or assumed, such as engineering drawings. But this approach may inadvertently destroy some important features of the sketch making it unsuitable for applications that support free-form sketching and make few assumptions about the user's intention. *Oversketching* allows the users to sketch what they want and correct any made mistakes by sketching over the error. This system then updates the sketch by cutting the region affected by the secondary stroke and smoothing the transition between the old and the new segments.

2.2.2.3 Sketch Interpretation

The final step of the pipeline is the interpretation of the sketch, in which its meaning is translated to a 3D modeling operation. In tradition systems (WIMP) every button or menu performs a specific and pre-defined task but in sketch-based systems the freehand input is inherently ambiguous and open to multiple interpretations. Olsen et al. propose a categorization of SBIM systems based on the types of modeling operations performed by each one: *creation systems* automatically generate 3D models from 2D input sketches; *augmentation systems* use input strokes to add new details to existing models and *deformation systems* use them to alter existing models with editing operation such as cutting, bending or twisting.

The problem of sketch recognition and interpretation has been solved by two standard approaches: *gesture/feature-based* classification and *geometric-based* classification.

Gesture-based Recognition

The first, and earliest, approach to sketch interpretation typically focus on *how* a sketch was drawn rather than on *what* the final sketch actually looks like. Gesture-based systems require each shape to be drawn in a particular style, making it a gesture, rather than a shape. Users have to learn how to draw each symbol since stroke order, stroke direction and the number of strokes are determining factors for recognition. The goal of these systems is to match the input stroke (a sampling of 3D points in the form of x, y, and time) to one of a pre-defined set of gestures. Recognition is performed based on a number of drawing-style features, such as the speed of the stroke, the start and end direction of the stroke, and the total rotation of the stroke. This approach has the benefit of using mathematically sound classifiers which produce fast and accurate classifications if users draw shapes as defined. However, gesture-based systems are very sensitive to changes in scale and rotation and require user training in order to achieve good results.

The idea of interacting with computers via pen-based input began in the 1960s with Ivan Sutherland's Sketchpad [76]. It proved to be beyond its time as pen-based interfaces would not catch until the 1990s. In 1991, Dean Rubine proposed a gesture recognition toolkit, GRANDMA, which allowed single-stroke gestures to be learned and later recognized through the use of a linear

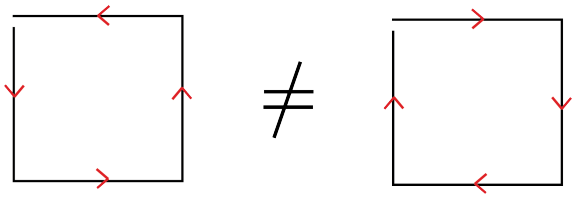

Figure 2.14: Gesture-based recognizers typically focus on how a sketch was drawn rather than on what the final sketch actually looks like so stroke order is relevant.

classifier [77]. Rubine proposed thirteen features which could be used to classify simple gestures with an accuracy of 98% on a fifteen-class gesture set when trained with at least fifteen examples. He also provided two techniques to reject ambiguous or non-gestures. Rubine's work was later extended in 2000 by Long et al. [78], who performed multi-dimensional scaling to identify correlated features and ultimately found an optimal subset that consisted of eleven of Rubine's features along with six of their own. Both of these works proved to perform well in recognizing two-dimensional gestures but their accuracy is not ideal when applied to natural sketch recognition problems because they put constraints on how users draw. A recent popular approach to gesture recognition is known as the $1 recognizer [79]. This easy and cheap recognizer facilitates the incorporation of gestures into user interface prototypes with about 100 lines of code. First, it resamples the input stroke to remove drawing speed variation and aligns it based on an "indicative angle" (that corresponds to the angle formed by the centroid and the gesture's first point), to provide rotation invariance. Then the gesture is scaled, non-uniformly, to a reference square and translated to a reference point. Finally, the gesture results in a set of candidate point that must be matched to a set of previously recorded templates that suffer the same transformations. This method provides highly overall accuracy with minimal training and low computational overhead but also presents some drawbacks as a result of its simplicity: it only recognizes unistroke gestures. To overcome some of this problems, a significant extension to this approach was later presented [80]: $N recognizer identifies gestures comprising multiple strokes and automatically generalizes from one multistroke to all possible multistrokes using alternative stroke orders and direction.

Geometric-based Recognition

Because of the drawing constraints imposed by gesture-based recognition systems, more recent approaches to sketch recognition shifted towards geometric-based techniques. Geometric-based recognizer focus on *what* the sketch looks like and less on *how* it was actually drawn, allowing users to draw as they would naturally. These techniques are considered geometric because they compare a stroke to an ideal representation of pre-defined primitives using geometric formulas. They recognize low-level primitive shapes that can then be combined hierarchically to form more complex shapes using specialized grammars like LADDER [81], a language presented by Tracy Hammond and Randall Davis in 2005, to describe how sketched diagrams in a

domain are drawn, displayed and edited. It consisted of pre-defined shapes, constraints, editing-behaviours and display methods as well as a syntax for specifying a sketch grammar and extending the language, ensuring that shape groups from many domains can be described.

The recognition of high level shapes depends on accurate low-level interpretations so many geometric-based recognizers have been developed. In 2001, Sezgin et al. presented a three phase system - approximation, beautification and basic recognition - that focused on interpreting the pixels generated by the user's strokes on an input device and producing low level geometric descriptions such as lines, ovals and rectangles [82]. It used a novel approach to detect vertices in sketched strokes. In 2003, Yu and Cai built a domain-independent system for sketch recognition that used low-level geometric features [83]. Later, in 2008, Paulson and Hammond presented PaleoSketch, a system that can recognize eight primitive shapes, along with complex shapes, with accuracy rates over 98.5% [84].

The advantage of geometric-based recognition systems is that they are typically more style-independent as they allow users to sketch in a non-constrained manner, requiring no individual training. However, geometric-based recognizers typically use numerous thresholds and heuristic hierarchies which are not mathematically sound. This makes inferences about generalization hard to determine because classification is not statistical. In addition, recognition accuracy is modest, unless tuned for a specific domain.

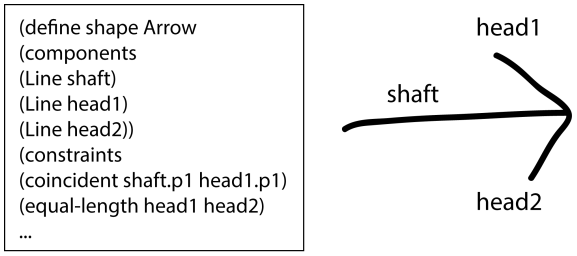

Figure 2.15: Geometric-based recognizers identify low-level primitive that form more complex shapes.

SBIM systems often use interfaces based on gestural recognition: simple strokes are used as input to specify commands and manipulate objects, directly or indirectly. The lack of numerous complex menus may be less intimidating to a novice user but still requires some learning time and effort to memorize what stroke corresponds to each operation.

Chapter 3

SketchFACE: Overview of the Proposed Solution

This chapter provides an overview of the preliminary work that led to the definition of the system design for the application developed during the course of this dissertation. It presents the main problem that inspired the work and details the methodology adopted to achieve the proposed objectives describing, by the end, all the functionalities expected for the final application.

3.1 Problem Statement

Realistic facial animation that fulfils the viewer expectations is very hard to achieve. Traditional modeling and animation software, although very powerful, is complex and requires from the users significant learning time to explore and master multiple menus and functionalities. With the evolution of user interfaces, new approaches to the human-machine interaction model, that replace traditional WIMP paradigm, have proven to be valid alternatives to ease the animation process.

Sketching on paper is often used in early prototyping stages of characters' design. By taking advantage of the fast propagation of natural user interfaces, sketching, as an intuitive means of communication, can now be used further down the animation pipeline. As they allow a more direct manipulation of 3D models, sketch-based interfaces reduce the complexity of otherwise difficult tasks of deforming and animating multiple expressions of a character. And if sketch-based software, that clearly recognizes people's intentions, is combined with the convenience of mobile devices, that require no intermediate input device other than the users fingers, a new range of opportunities becomes available to non-experienced artist. Thus, exploring more "natural interfaces" for facial animation, particularly using mobile devices, would be useful to ease the modeling and animation processes of 3D models, creating new opportunities for novice users and widening the target users of these animation systems.

3.2 SketchFACE Approach

It is the aim of this dissertation to start from a novel version of a facial sketching interface control system created by Miranda et al. [13], that allows to manipulate a model's rig through free-hand drawing, and deploy it for Apple's iPad. This takes advantage of the capabilities of mobile devices and the new ways of human-computer interaction enabled by them and allows to evaluate if a multi-touch capable interface is useful to decrease user's learning curve when using software for facial modeling and animation.

It is also a purpose of this work to create a simple and intuitive facial animation application that can be used in a collaborative environment with multiple users and external devices that display content in a more immersive way. This new application, inspired by the LIFEisGAME project, is called SketchFACE as it provides a *Sketch-based Facial Animation Collaborative Environment*. The application fits within the presented project, following the same technological challenges and interface styles and adding two additional game modes: one that explores a different interaction model to control the 3D avatar, based on 2D control canvases, and other that explores the possibility of a collaborative modeling and animation environment where multiple users can, at the same time, deform a model in a joint effort to produce an animated sequence of poses.

Thus, the major results expected for the end of the dissertation are:

- Deployment of a simple and intuitive sketch-based modeling and animation application for mobile devices, more specifically, the Apple's iPad that takes advantage of the touch capabilities of tablets to implement an hybrid sketching/dragging rigging control system;
- Possibility of integration of multiple players and visualization devices in order to create a collaborative and immersive environment that allows simultaneous alteration of a single model to increase challenge and entertainment;
- Expansion of the scope of the project LIFEisGAME also to individuals without ASD;
- Seamless integration of the new application into the LIFEisGAME prototype as additional game modes.

3.3 Challenges

This research supports the creation of a simple and intuitive application for facial animation that explores a new interface: mobile devices. It also intends to evaluate if these handheld, portable, touch-capable and easily wirelessly interconnectable devices are useful to ease the task of giving life to 3D characters, thus opening new opportunities for non experienced users. However, many challenges are expected being the first ones related with the device chosen for prototyping purposes - Apple's iPad:

- **The iPad's Closed System:** iPad's operating system, iOS, is extremely restrictive. Apple is not very permissive in terms of what contents can be transferred to the iPad and creating applications for the tablet requires licenses, the signature of a a non-disclosure agreement and the payment a developer subscription. All these restrictions may prove an obstacle to new developers, that sometimes may find it very hard to find a solution for their problems that is "Apple approved".

- **Reduced public access:** tablet technology is not yet as widespread as traditional computers. In fact, according to data from 2012 from OberCom (*"Observatório da Comunicação"* - The Observatory for the Media) about the Internet in Portugal, half (50,5%) of the Portuguese adults has laptop computers and desktop computers ownership rate is 35,2%. On the other hand, devices for mobile internet access (21,7%) and tablets (1,5%) are among the types of equipments held by a limited segment of the population [85]. However, mobile devices' ownership rates tend to significantly increase, worldwide, in the near future. Forecasts by the networking giant Cisco say that the fastest growth in device adoption over the next five years will be for tablets, with a prediction of an average of 46% growth year on year, and data growth of 113% annually. This way, mobile-connected tablets will generate more traffic in 2017 (1.3 exabytes per month) that the entire global mobile network in 2012 (885 petabytes per month) [86].

At a more general level, using the multi-touch displays of mobile devices as interface for this facial animation application raises problems at two different levels, that are discussed in the following sections, mostly based on the observations of Bill Buxton's [58]:

- Disadvantages of multi-touch interfaces.

- Interface design concerns.

3.3.1 Disadvantages of multi-touch interfaces

Multi-touch systems have an extraordinary ability of adaptation to almost any imaginable problematic situation. As screen content can be freely modified, being, for example, able to simulate input devices (such as keyboards), touch-screens are very flexible user interfaces that enable the creation of intuitive applications, when correctly designed [87]. But taking advantage of this types of interfaces may involve considering not only their positive aspects but also some of the disadvantages associated to these equipments:

- **Constant visual demand:** it is virtually impossible to manipulate a touch-screen without full visual attention on the display (for example when the lights are out or when our eyes are occupied elsewhere) because there is no tactile feedback that indicates the position of each interface elements. Thus, this technology has little use for visually impaired users and may even be dangerous if used while performing other activities such as driving, unless the device in question also supports speech recognition.

- **Size requirements:** the size of the multi-touch display significantly influences the types of gestures that are suited for the device and the number of fingers/hands that can be used on the surface. This means that small screens may not be appropriate to some actions like taking notes or making detailed drawings.
- **Sunlight:** multi-touch interfaces are usually incorporated in mobile devices that, as the name indicates, are specially designed to be operated outside the traditional desktops, sometimes even while in movement. This may cause exposure to adverse usability conditions, such as bright sunlight, that heavily affect visibility. Unless the device has an outstanding reflective display, it may be unusable in some light conditions.

3.3.2 Interface design concerns

Creating a simple and intuitive user interface is always a major challenge in development of every application. The interface design major focus must be on the user's experience and interaction, allowing simple and effective accomplishment of the goals and tasks at hand. But user interface design must always take into account the specific characteristics of the hardware for which the application was design for. Although often clustered together as computing devices, mobile devices, such as smartphones or tablets, are very different from traditional desktop computers, either in terms of screen size, connectivity reliability, bandwidth, battery life, and so on. Given these many differences, it is clear that designing for mobile devices, often equipped with multi-touch interfaces, is very different that designing for the desktop. As applications must embrace the device characteristics, some aspects should be considered before starting the user interface design:

- **Fat finger problem:** when drawing on a touch-screen (without a stylus), one sometimes notices that lines are not represented on the screen exactly on the desired spot. This issue, know as "fat finger problem", arises from the fact that a finger is usually relatively large when compared to the small touch-screen. To avoid this, interface interaction elements must have a certain minimum size in order to be individually and precisely touched by human fingers.
- **Occlusion problem:** mobile devices are manipulated with hands, and, more particularly, with fingers that touch specific parts of the screen. As fingers are not transparent, they occlude parts of the interface and the smaller the touch-screen or the more fingers are used, the more covered the display will be. This problem can be eased with clever interface design approaches [88] or with the substitution of the finger by an appropriate stylus, that is very skinny and, therefore, does not obscure the screen.
- **Interface interaction principles:** the types of actions performed to interact with a touch technology significantly influence users' assessment of it. The same touch technology, on the same device, can assume a very different character depending on the nature of the touch inputs it accepts or, by other words, depending on whether the interface is designed for discrete or continuous actions. Discrete actions correspond, for example, to pushing a graphical

button or tapping a virtual keyboard key while continuous actions correspond to gestures, such as the lateral stroke commonly used in photo-viewing application to go to the next or previous image. Discrete actions must be accompanied by graphical cues (some kind of feedback that actually works forward and not backwards) that indicate to the users that they can be performed. Some continuous actions share these properties but many do not like the "pinching" gesture to zoom in. Somehow, users must already know that it is possible to execute it, when and where. To ease user interaction, the interaction model chosen must follow known guidelines defined for the device in question.

- **iOS interaction model:** iOS devices (such the iPad used in the case of this dissertation), are operated with specific sets of standard gestures. People become familiar with them because built-in applications use them consistently. So, when operating an iOS device with other application users are already expecting to perform successfully some particular actions to manipulate on-screen content, such as flick or drag to scroll a long list, pinch to zoom in on an image or tap a button to activate it. So, iOS developers must take into account the standards and paradigms people are comfortable with, maintaining a certain consistency encourage by Apple in its *iOS Human Interface Guidelines* [89].

3.4 System Design

This section intends to capture and convey the design decisions which have been made prior to the system's deployment that served as guidelines for the implementation work. It provides a comprehensive overview of all the system's features depicted by an interface prototype.

3.4.1 System Scenarios

The system aims for seamless interaction between many users and devices. It includes two modes: a standalone version, for a single user and a collaborative environment that allows multiple simultaneous users. Each mode supports not only users using individual tablets but also supports the connection of external devices for better content visualization. Each type of device provides a different interaction model due to the capabilities it offers in result of its hardware and software (for example, a TV does not allow multi-touch input as a tablet but a tablet cannot offer a big screen for content visualization). With this in mind, it was decided that the iPad's would support most of the direct interaction with the user (corresponding to the primarily means of input) while other devices (TV, computer screens, projectors) would provide visual feedback for the actions executed on the tablet. This option transforms the iPad into a controller for the content displayed in the external screen. External displays must be connected to the iPad via Apple TV.

The different scenarios of the application are summarized in figure 3.1 and detailed below.

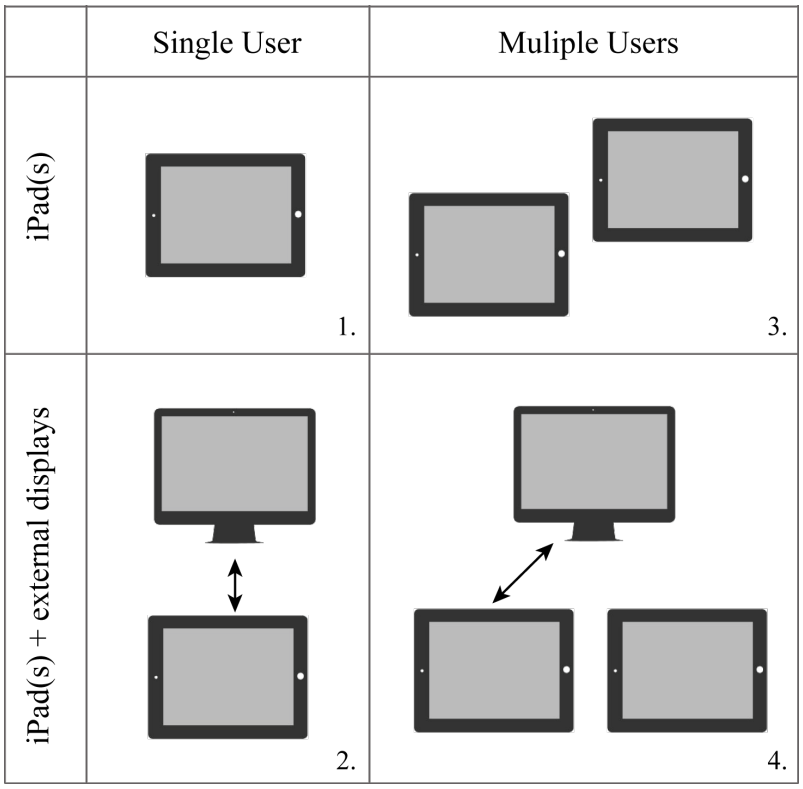

Figure 3.1: Different usage scenarios for the application

- **Scenario 1:** a single user runs the application in the iPad without any external display. Both the controls and results of the modeling and animation processes are displayed in the tablet's screen.

  Being the simplest scenario, and the first to be implemented, it raises the initial problems related to sketching control method chosen, that is the foundation of the entire application. In order to create a simple sketch-based modeling and animation application that allows 3D model deformation controlled by a 2D interface, Miranda et al.'s approach can be adopted: several rig controls can be manipulated at the same time with a single stroke. Input information acquired in the 2D control interface is mapped to the 3D world coordinate system, resulting in the creation of different character poses.

- **Scenario 2:** a single user connects an external display to the iPad. A new icon indicates the availability of the new display and, if the user chooses to use it, the iPad is used to control the 3D model displayed on the external device.

  This scenario adds the problem of multi-screen support, as the iPad must act as a control interface while the results of the modeling and animation processes are shown in a larger display connected to it. This obstacle can be overcome by a new functionality available in the platform chosen for system prototyping (Unity3D game engine), that allows the communication between the iPad and an external screen (of an HD TV, for example) via the digital

media receiver Apple TV, through AirPlay.

- **Scenario 3:** multiple users, in the same room or not, run the application in their own iPads in multiplayer mode and cooperate in the modeling and animation processes.

  Scenario 3 presents the new obstacles of adding networking capabilities to the application, in order to allow cooperative work, and the necessity to define some rules that manage this collaborative intervention, since the actions of one user must not conflict with the actions of the others. Thus, a client-server approach can be used to guarantee synchronization between all the participants. Conflicting situations on the modeling process when several users are simultaneously altering a single character were avoided with the definition of an hierarchy or alteration powers.

- **Scenario 4:** multiple users, in the same room, run the application in multiplayer mode but, as one of the iPads is connected to an external display, all of them can see the results of their cooperative work in a larger display.

  This scenario encompasses all the problems raised by scenarios 1, 2 and 3.

3.4.2 System Use Cases

The user of SketchFACE can perform the following actions:

1. **Activate external display:** if an external screen is detected, a button allows the user to activate it so it begins to display a portion of the visual content of the application;
2. **Deactivate external display:** an active external display can be deactivated so all of the visual content is shown again on the iPad screen;
3. **Create different facial poses:** This can be done on-the-fly by drawing strokes on a separate 2D control canvas. Depending on how a stroke is drawn, it may deform a section of the face or drag certain control points in a determined direction. This can be also done individually or in a collaborative effort with other users;
4. **Save created pose to timeline:** poses in the timeline act as key-frames for the animated sequence. A created pose can be saved to the next available position of the timeline;
5. **Drag timeline:** When the timeline exceeds the screen size, it can be dragged back and forward to allow the access to a specific frame;
6. **Animate the sequence of poses in the timeline:** a "Play" button starts the animation of the poses in the timeline;
7. **Pause timeline animation:** a "Pause" button interrupts the animation of the currently playing sequence of poses in the timeline. This button only appears when a sequence is playing. A paused sequence will start from the same point where it stopped once the "Play" button is selected again;

8. **Stop timeline animation:** a "Stop" button allows the interruption of the animation sequence. The stopping point is not recorded so once the "Play" button is selected again, the animation will start from the first frame of the timeline;

9. **Modify order of the poses in the timeline:** a timeline frame can be selected and dragged to a different position;

10. **Modify length of a pose in the timeline:** timeline frames can be resized so a specific pose lasts more or less in the animated sequence;

11. **Duplicate frame at the end of the timeline:** a frame dragged and dropped at the end of the timeline will be duplicated on this new position;

12. **Duplicate frame at the beginning of the timeline:** a frame dragged and dropped at the beginning of the timeline will be duplicated on this new position;

13. **Delete pose from timeline:** a specific timeline frame dragged to the "Garbage" icon will be permanently deleted;

14. **Delete all poses from timeline:** a "Delete All" button will empty the timeline;

15. **Clear all deformations from the 3D model:** a "Clear" button resets the 3D model to its initial state, without deformations;

16. **Reset a single canvas:** a specific stroke drawn on a 2D canvas to deform model's geometry can be deleted with an "Eraser" button. This will reset that canvas to its original state.

3.4.3 Interface Prototype

SketchFACE aims to be a simple and intuitive sketch-based facial modeling and animation aplication. Its major feature intends to be the translation of information acquired through a 2D interface (that corresponds to strokes drawn by the users) into 3D world coordinates that deform a model's pose. In order to fulfil this goal, while keeping special focus on simplicity, interface design must be a carefully thought issue. Interfaces must, on one side, ensure intuitiveness of the overall system so that the user requires no previous learning time or tutorials to explore all the functionalities available. On the other side, the application must follow the generic guidelines of mobile devices programming, that take maximum advantage of the hardware and software capabilities, and more particularly, follow the specific iOS user experience paradigm that corresponds to some expected interaction and navigation methods usually associated to Apple's devices.

This way, the problems and concerns discussed in sections 3.3.1 and 3.3.2 were taken into account when thinking about interface elements positioning.

Figure 3.2 presents a SketchFACE interface prototype, followed by the explanation of each of its sections.

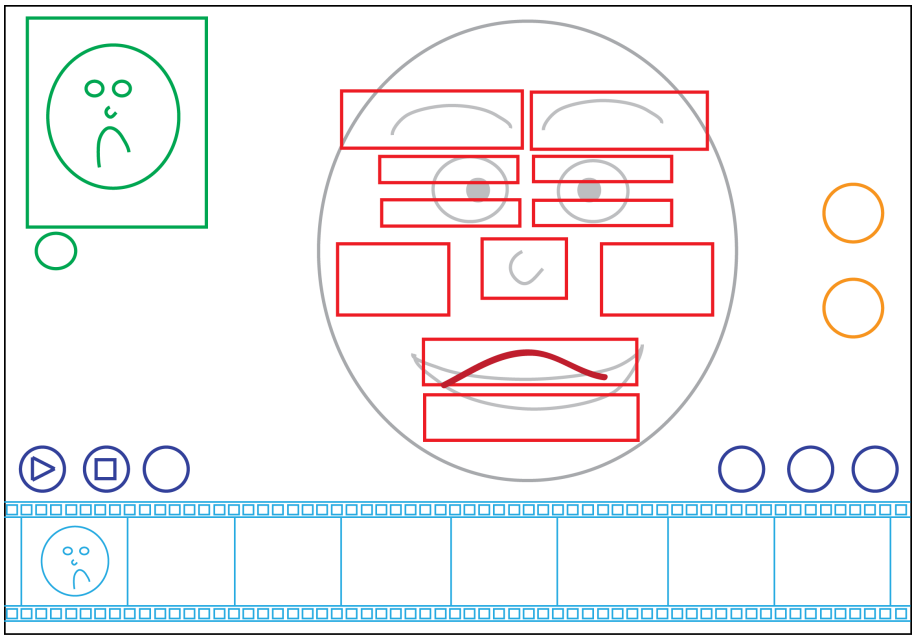

Figure 3.2: SketchFACE interface prototype.

- **Red area:** 2D control canvases over a generic face image where the user draws the strokes that deform the correspondent 3D model parts.

  This section occupies most of the available screen area because it is specially intended to handle the majority of user's interactions that require great precision. Some actions, like pushing an interface button do not require great precision but the actions performed in the sketching canvases, like moving a control point to refine a particular pose, are very sensitive. By making this area as big as possible, the *fat finger problem* is minimized, guaranteeing that input drawing areas have a certain minimum size that is compatible with their function.

  This area processes the events related to Use Case 3.

- **Orange area:** buttons associated with the modeling process: "reset pose" and "reset a single canvas".

  Were placed next to canvases area because they help the modeling process allowed by stroke drawing performed on the canvases.

  This area processes the events related to Use Cases 15 and 16.

- **Green area:** section with the visualization of the final 3D model where users can see results of the deformations, generated by the strokes drawn on the 2D control canvases, and the animated sequence. A button in this area will be available when an external display is detected. If the button is selected, the 3D model will be displayed also on the external screen.

Results area was placed on the left part of the screen to counteract occlusion problems that arise when the user is drawing strokes, with the finger, in the canvases areas. As only an average 10-15% of the population is left-handed [90], it is more likely that users draw strokes and handle other interface controls with the right hand, thus obscuring that side of the screen. Placing the results area in the left side guarantees that, for the majority of the population, they will be visible while handling modeling interactions.

This area processes the events related to Use Cases 1 and 2.

- **Light blue area:** timeline for the keyframe poses that will generate the animated sequence.

  Timeline area is relatively thin in order to leave the majority of screen area to the drawing canvases. A timeline filled with poses can be dragged to the left and to the right, in order to show new available positions, with the familiar iOS scroll action.

  This area processes the events related to Use Cases 5, 9, 10, 11 and 12.

- **Dark blue area:** control buttons associated to the timeline.

  As they correspond to actions related to the timeline, they were placed near it.

  This area processes the events related to Use Cases 4, 6, 7, 8, 13 and 14.

Chapter 4

SketchFACE: Implementation

SketchFACE is a new application specially oriented for Apple's iPad that proposes the use of a hybrid sketching and dragging approach to manipulate a model's control structure, known as rig, in a collaborative environment where users can join efforts to simultaneously deform and animate a 3D character. Taking advantage of the "natural interface" offered by this mobile device, it tries to create new opportunities for non-expert users and wide the target audience of animation systems while exploring the viability of collaborative work in the modeling and animation processes.

The present chapter describes the implementation of the SketchFACE application. By the end of it, the reader should have a clear idea about the main design and architectural decisions made and how the different modules were implemented.

4.1 System Implementation

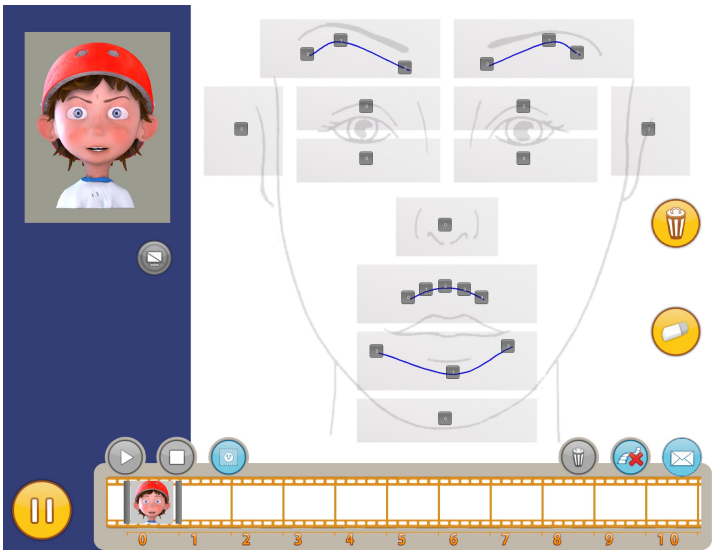

Figure 4.1: SketchFACE interface.

SketchFACE is the result of the attempt to create a sketch-based collaborative facial animation environment. Users interact with a control interface on an iPad to deform a 3D model by drawing strokes on 2D canvases or by dragging control points that manipulate the character's rig. The visual feedback of the performed deformations can be seen on a small corner section of the tablet's screen or, for better visualization, the results can be displayed in a much larger screen (of a HD television) if one is available and connected to the iPad, increasing the immersiveness of the animation environment. The created poses can then be animated, acting as key-frames for an animated clip. All these actions that can be performed individually can also be executed along with other users, in a collaborative animation session similar to a "game room" that allows joint effort in the modeling process. Figure 4.1 shows the application interface.

The SketchFACE system can be divided into three major modules, depicted in figure 4.2. This section starts by describing the development tool chosen to create the application prototype and then presents the system modules and details how each one was implemented.

4.1.1 Development Tool: Unity3D

The first major decision on the path of creating a collaborative facial animation environment was to choose the appropriate tool to create the prototype. The best and chosen candidate was Unity game engine [91] since it fulfilled the major development requirements of the project and had proven to be suitable for prototyping, being used in other modules of LIFEisGAME. Choosing the same development tool made the integration of the new application SketchFACE within that project simple and direct.

Unity, also called Unity3D, is a cross-platform game engine developed by Unity Technologies. It is used to make games for web plugins, desktop platforms, video game consoles and mobile devices that can be deployed to ten global platforms: Windows, Linux, Mac, Android, iOS, Unity Web Player, Adobe Flash Player, PlayStation 3, Xbox 360 and Nintendo Wii. Unity's development ecosystem features a powerful rendering engine with many intuitive tools and a uniquely powerful and flexible animation system. It supports art assets and file formats from 3ds Max, Maya, Softimage, Blender, etc, so models created with other 3D animation software can be easily imported and managed through Unity's graphical user interface. It also offers some platform-specific features that are very useful because building games for devices like the iPhone and iPad requires a different approach than creating desktop PC games. For mobile iOS devices, Unity offers, for example, a number of scripting APIs to access the multi-touch screen, the accelerometer or the device geographical location system and downloadable content that can be used to implement in-app purchases. Unity supports JavaScript, C#, and Boo.

Since the chosen development tool was a game engine and given that the resulting application was integrated in an on-going project whose prototype was a serious game to help children with ASDs, most of SketchFACE's architectural decisions were based on theories also applied to game development and the application was sometimes referred as a game.

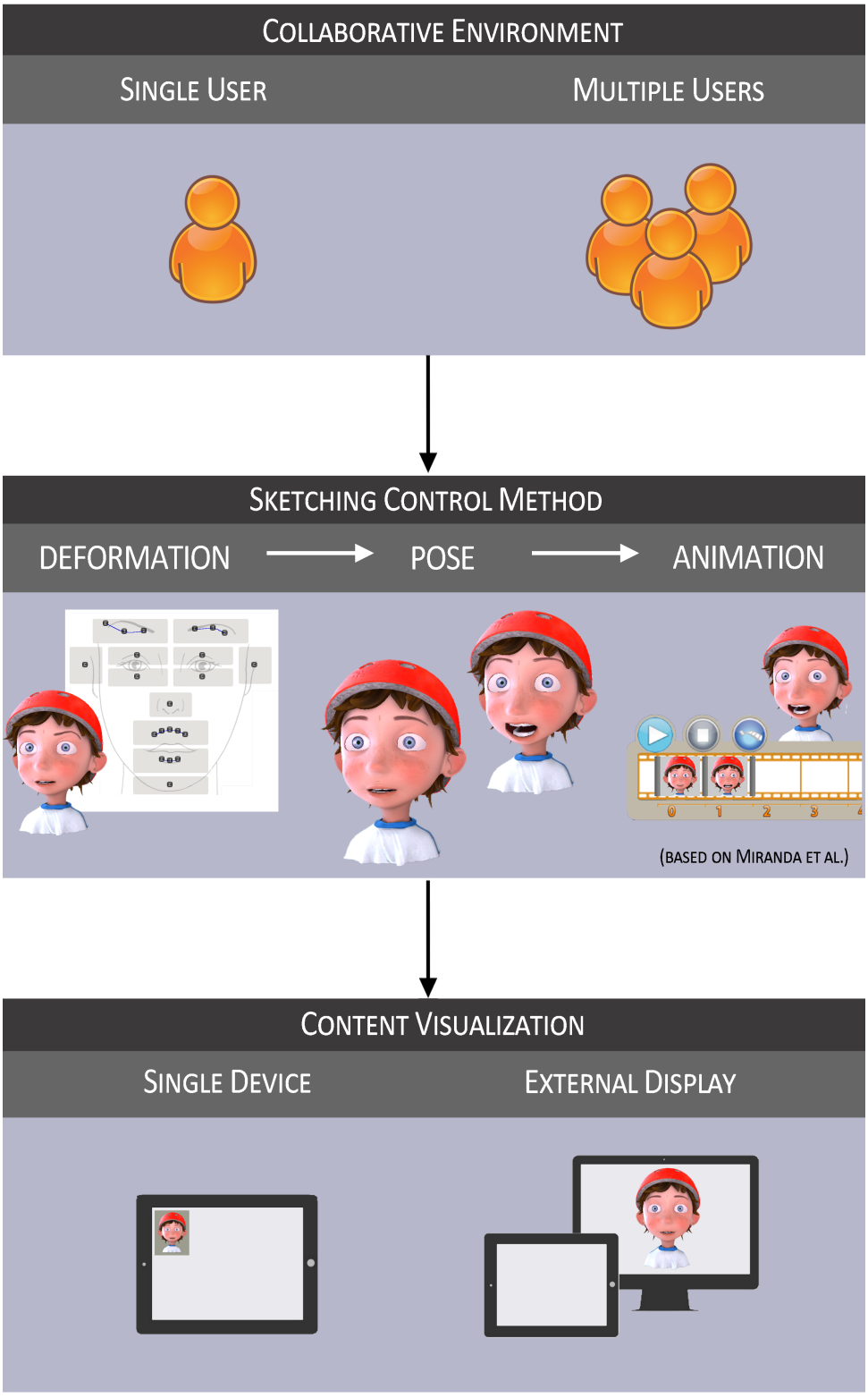

Figure 4.2: The three major modules of SketchFACE.

4.1.2 Module 1: Sketching Control Method

The work of Miranda et al. [13] proposed a paradigm shift in the way a model's rig is manipulated that contrasted with traditional modeling techniques. Instead of handling each control individually, in a discontinuous way, with their approach an artist can now manipulate several rig controls at the same time with a single stroke. They developed a facial sketching control system that allows the artist to draw strokes directly on the 3D mesh or on a virtual canvas and illustrated it with the deployment of a plug-in for Maya, chosen for prototyping purposes.

This dissertation work focused on deploying a sketch-based facial animation application based on Miranda's work, in C#, for the iPad. SketchFACE translates the virtual canvas approach to an environment where a tablet serves as control interface to deform and animate a 3D model, taking advantage of the portability and intuitiveness of such mobile device.

In SketchFACE, the sketching control method is essentially responsible for mapping all the information acquired through the 2D interface, divided into drawing canvases, into the 3D coordinate system of the model geometry, divided into regions. This is done in the following four stages: *initial setup*, *stroke drawing*, *repositioning of the reference curves* and *model deformation*.

Initial Setup: Map the default model pose into the 2D interface

The sketch-based interface provided to the user is divided into 12 drawing canvases, as can be seen of figure 4.1. Each one is responsible to control a particular region of the model. This means that the strokes drawn on a particular canvas will influence the deformation of all the joints of the correspondent region of the model. Consequently, the mesh will be deformed creating a facial expression. Besides being a drawing area, each canvas contains a reference curve that passes through a set of control points that represent, in 2D coordinates, the current position of the joints of the correspondent 3D region of the model.

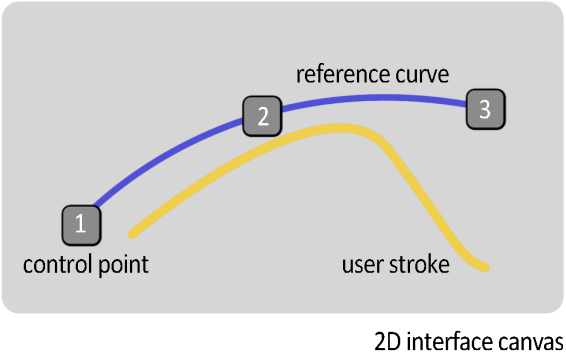

Figure 4.3: Elements of a 2D interface canvas: the control points, the reference curve and the stroke drawn by the user.

When the application starts, the default model position must be translated into the 2D interface, to provide visual feedback to the user of the available controls to deform the character. This involves mapping the 3D position of the model's joints, in world's coordinate system, to a 2D location in the screen's coordinate system.

The world coordinate system is the three dimensional system used for building a 3D scene as a single unified model. Every object placed in the world has its own coordinate system (model coordinate system) but there is only one world coordinate system to define the relative position and orientation of every generated objects. This is why the world coordinate system is always fixed. Every scene has an origin (0,0,0) and the objects in the scene are placed with reference to this origin.

The screen coordinate system is defined as a two dimensional device-dependent coordinate system whose origin is usually located at the lower left corner of the screen. It refers to the physical coordinates of the pixels on the device screen, based on current screen resolution.

To make this mapping between coordinate systems, an orthographic camera looking at the 3D model is needed. With a specific camera function provided by Unity3D, the position of the joints in 3D world space, ji_{world} is transformed into 2D screen space, defined in pixels, as depicted in the left side of figure 4.4. However, this points are not yet located inside the correspondent canvas screen space. They must be repositioned. For regions with only one joint, it is directly repositioned to the center (horizontally and vertically) of the canvas area. For regions with more that one joint, a slightly more complex transformation must be performed. First, the bounding box of each group of joints of a region is calculated, both in world and in screen coordinates, resulting, for each one, in two pairs of points that represent the bottom-left, r0, and the top-right, r1, extremes. Then, the correct position of the control point, cp_i, in screen coordinates is calculated imposing the following relation:

$$r0_{world} - r1_{world} \longrightarrow r0_{screen} - r1_{screen} \tag{4.1}$$

$$middlePointWorld - ji_{world} \longrightarrow middlePointCanvas - cp_i \tag{4.2}$$

where middlePointWorld and middlePointCanvas are the central points of the bounding boxes of the groups of joints in both coordinate systems. This is used both for x and y coordinates. Basically, this imposes that the distance of the point to re-allocate to the center of the bounding box must be proportional to the length/width of the bounding box in both coordinate systems. After all these calculations, each control point is correctly represented inside the appropriate canvas, in screen coordinates, as shown in the right side of figure 4.4. The reference curve that passes trough them can, then, be calculated, according to a following step of the method. The reference curve is not equal but calculated from the strokes drawn on the canvas. It is updated with each new drawn stroke to always depict the position of the joints in the 3D world.

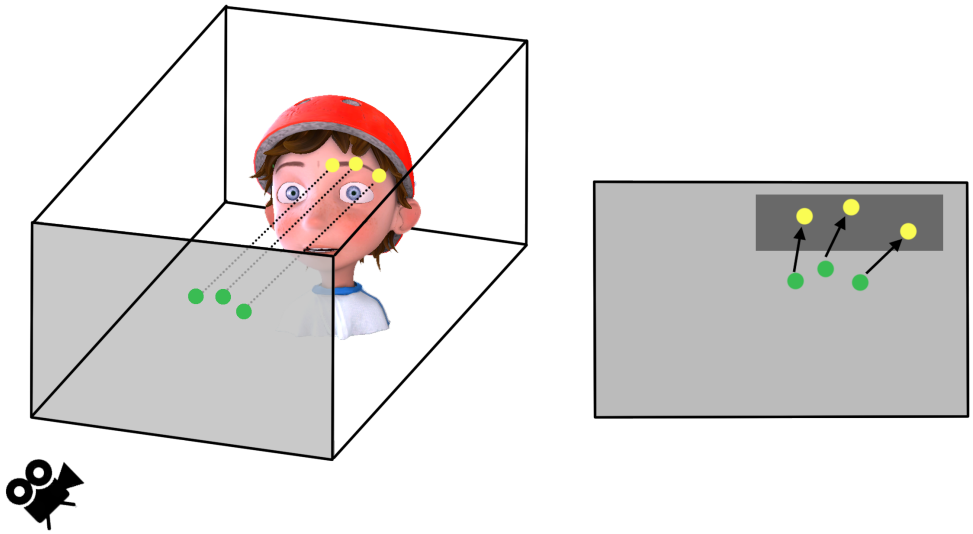

Figure 4.4: Mapping the default model pose into the 2D interface. Left: transformation of joints world coordinates to screen points. Right: repositioning of screen points to fit the appropriate canvas area.

Stroke drawing

Each stroke, S, drawn on the 2D interface is recorded as an ordered set of n points $s_0, s_1, s_2, ...$ In Miranda et al.'s method, each stroke was stored as a parametric NURBS curve N with degree D = 3. Each curve N was parameterized with t edit points, where t corresponds to the total number of joints that belong to the same region of the 3D model. In SketchFACE, due to the absence of a direct representation of a NURBS curve in Unity, a slightly different approach was taken. From the n points of the stroke S, a set of control points, cp_i was chosen. For each stroke drawn on a particular canvas, the number of control points selected is equal to the number of joints of the associated model region. The choice of these points is similar to Miranda et al.'s method but here, each selected point is not an edit point of a NURBS curve. They are called control points because the movement of each one in the 2D canvas allows individual control of one joint of the associated region (they must not be mistaken by the control points of a NURBS curve). The chosen set of points is used to calculate an Hermit curve because it is easier to compute.

The method computes at least 3 control points (or more for regions with more than 3 joints) according to the following formula:

$$cp_i = S\left(\frac{i*(n-1)}{t-1}\right) \qquad i = 0,...,t-1 \qquad n > t \tag{4.3}$$

For regions with only one joint, the chosen point will correspond to the middle control point and in regions with two joints, it will be considered the first and last control points.

Repositioning of the reference curve

The new position of the reference curve of the 2D interface is calculated through interpolation between the control points chosen (or the control points position calculated in the first step of this method). Between each two control points a Hermite curve is calculated. To do this, the following vectors are needed:

- P_1: the start point of the curve;
- T_1: the tangent to how the curve leaves the start point;
- P_2: the end point of the curve;
- T_2: the tangent to how the curve meets the end point.

And the following 4 hermit basis functions are also used:

$$h_1(s) = 2s^3 - 3s^2 + 1 \tag{4.4}$$

$$h_2(s) = -2s^3 + 3s^2 \tag{4.5}$$

$$h_3(s) = s^3 - 2s^2 + s \tag{4.6}$$

$$h_4(s) = s^3 - s^2 \tag{4.7}$$

To build the interpolated point along the curve the 4 vectors are multiplied by the 4 hermite basis functions and added together:

$$p' = h_1 \times P1 + h_2 \times P_2 + h_3 \times T_1 + h_4 \times T_2 \tag{4.8}$$

These calculated points will generate the new 2D reference curve of that canvas that will influence the model deformation in 3D coordinates.

Model deformation

Model deformation is done based on the above translation of the control points relative to their original position. A normalized vector is calculated to represent vertical and horizontal translation of the control point from its default position towards an edge of the canvas so that (-1, -1) corresponds to the translation of the control point to the inferior-right corner of the canvas and (1, 1) corresponds to the translation to the upper-left corner of the canvas. Any smaller translation will correspond to a vector whose coordinates are values of less than 1.

As Unity3D has some problems importing model constraints correctly, in order to restrict the movements of the bones to believable positions an auxiliary XML file is used. This file contains information about the maximum translation and rotation that the joints can perform, in model coordinates.

Model coordinates is the coordinate system local to a specific object. The origin of the object model is usually picked somewhere on the model itself but it does not have to be.

The maximum control point translation transmitted by the vector (1,1) will induce the maximum joint translation/rotation allowed by the XML file. Intermediate values will lead to proportional translations/rotations.

This explains how the *x* and *y* coordinates of the world coordinate system are calculated based on the information acquired from the drawing canvases. However, the world coordinate system is composed by three coordinates: x, y and z. The *z* coordinate is calculated through raycasting techniques similar to the ones described in Miranda et al.'s work [13]. In order to constraint the joint movement to the 3D mesh, the z coordinate chosen for the final position of the joint is always tangent to the model's surface. To compute this, first a normal vector to the mesh is added to the point obtained with the mapping of the x and y coordinates, resulting in an auxiliary point, in from of the mesh. Then, the method casts a ray from the auxiliary point in the inverse normal direction and the intersection between that ray and the mesh is chosen as final position for the joint with xyz coordinates computed.

This concludes the explanation about how the sketch-based method for facial modeling was implemented in SketchFACE.

4.1.3 Module 2: Collaborative Environment

In order to allow multiple users to deform a single model, some networking principles needed to be implemented to create a collaborative environment. Real-time networking is a complex field but Unity3D makes it easy to add networking features to a game. However, it is useful to have some idea of the scope of networking before using it in a game. This section explains the fundamentals of networking applied to the game field, along with the specifics of Unity3D's implementation, details the architectural scenarios evaluated and summarizes the decisions made on the topic of multiplayer support on SketchFACE.

4.1.3.1 Networking Concept

Networking is communication between two or more computers, usually following a model of client-server. The client is the computer that requests the information while the server is the computer responding to the request. The server can either be a dedicated host machine used by all clients, or simply a player machine running the game (client) but also acting as the server for other players. Once a server has been established and a client has connected to it, the two computers can exchange data as demanded by gameplay.

4.1.3.2 Networking Approaches

There are two common approaches for structuring a network game which are known as *Authoritative Server* and *Non-Authoritative Server*. Both approaches rely on a server connecting clients and

passing information between them and both offer privacy for end users since clients never actually connect directly with each other or have their IP addresses revealed to other clients.

- ***Authoritative Server:*** this approach requires the server to perform all world simulation, application of game rules and processing of input from the player clients. Each client sends their input (in the form of keystrokes or requested actions) to the server and continuously receives the current state of the game from the server. The client never makes any changes to the game state itself. Instead, it tells the server what it wants to do, and the server then handles the request and replies to the client to explain what happened as a result.

- ***Non-Authoritative Server:*** does not control the outcome of every user input. The clients themselves process user input and game logic locally, then send the result of any determined actions to the server. The server then synchronizes all actions with the world state. This is easier to implement from a design perspective, as the server really just relays messages between the clients and does no extra processing beyond what the clients do.

After covering the basic architectures of networked games, it is important to explore the lower-levels of how clients and servers can talk to each other. There are two relevant methods: *Remote Procedure Calls* and *State Synchronization*. It is not uncommon to use both methods at different points in any particular game.

- ***State Synchronization:*** is used to share data that is constantly changing. The best example of this would be a player's position in an action game. The player is always moving, running around, jumping, etc. All the other players on the network, even the ones that are not controlling this player locally, need to know where he is and what he is doing. By constantly relaying data about this player's position, the game can accurately represent that position to the other players.
  This kind of data is regularly and frequently sent across the network. Since this data is time-sensitive, and it requires time to travel across the network from one machine to another, it is important to reduce the amount of data that is sent as much as possible. In simpler terms, state synchronization naturally requires a lot of bandwidth, so you should aim to use as little bandwidth as possible.

- ***Remote Procedure Calls (RPCs):*** are used to invoke functions on other computers across the network, although the "network" can also mean the message channel between the client and server when they are both running on the same computer. Clients can send RPCs to the server, and the server can send RPCs to one or more clients. Most commonly, they are used for actions that happen infrequently. For example, if a client flips a switch to open a door, it can send an RPC to the server telling it that the door has been opened. The server can then send another RPC to all clients, invoking their local functions to open that same door. They are used for managing and executing individual events.
  An RPC call can have as many parameters as desired but numerous or large parameters will

have an impact on the network bandwidth involved. Unlike a normal function call, an RPC needs an additional parameter to denote the recipients of the request which can be: only the server, everyone or everyone except the sender. RPC calls can also be buffered. Buffered RPC calls are stored up and executed in the order they were issued for each new client that connects. This can be a useful way to ensure that a late coming player gets all necessary information to start.

4.1.3.3 Master Server

Each "game room" where players meet requires one server that assures communications between them. The server can be a machine that is also running the game, which mean that one of the players may act as server and the other players connected to it act as clients, or it can be a dedicated machine that is not participating in the game. Either case, it is always necessary one *Master Server* used to matchmake servers and clients so servers can be advertised and compatible clients can connect to running games. As such, the Master Server is essentially a basic lobby server, but with some special features. Its purpose is also to hide IP address and port details, and to perform technical tasks around setting up network connections which otherwise would be impossible, like acting as a proxy server to let clients behind a firewall act as a game server.

Unity Technologies has a fully deployed online Master Server available for testing purposes. It is a built-in meeting place for games that are actively seeking clients, and player clients who want to connect to them. Unfortunately, it has no uptime guaranty which makes it an unreliable solution. However, the source code of Unity's Master Server is freely available for anyone to use and the server can be deployed on Windows, Linux and Mac OS [92]. It can also be customized with modifications to the source code.

4.1.3.4 SketchFACE Networking Approach

The chosen networking approach for the collaborative environment of the SketchFACE application relied on a Master Server built and running from the source code made available at Unity3D website since it fulfils all the necessary networking requirements and overcomes the online server unreliability. The communication between users is assured by buffered RPC calls, since the data exchanged is not constantly changing, in order to ensure model, canvas and timeline synchronism between all users, in the following situations:

- **Lock canvas:** when a user starts using a particular canvas by beginning a stroke or moving a control point (Use Case 3), it must be locked for all other users to avoid conflicting model deformations;

- **Unlock canvas:** when a user ends the interaction with a particular canvas by ending the stroke or releasing a control point (end of Use Case 3), it must be unlocked so other user can use it;

- **Send pose information:** when the model's pose is altered by any user (end of Use Case 3), this information must be send to other users so they can recreate the pose in their devices and thus, maintain the synchronism inside the collaborative environment;
- **Reset pose:** when one of the users selects the button responsible for making the model return to the initial neutral neutral pose (Use Case 15) this transformation must be propagated to all other users;
- **Reset a single canvas:** sends the information to reset a particular canvas (control points and reference curve) and the 3D model bones associated, to all the connected users, when one of them performs this action (Use Case 16);
- **Save pose to timeline:** propagates to all the connected users a new frame saved to the timeline (Use Case 4);
- **Delete timeline frame:** when a user deletes one of the timeline frames (Use Case 13), it disappears from the timeline of all the connected users;
- **Drop frame at the end of the timeline:** a frame dragged and dropped at the end of the timeline will be duplicated on this new position (Use Case 11). If a user performs this action it will be replicated on the timeline of all other users;
- **Drop frame at the beginning of the timeline:** this action is similar to the previous one but occurs when a frame is dragged and dropped at the beginning of the timeline (Use Case 12);
- **Reposition frame:** when a user drags and drops a frame to a new timeline position (Use Case 9) it must be repositioned for all other users;
- **Change frame length:** when a user changes the length (that corresponds to the duration) of a frame (Use Case 10), this transformation must be propagated to all other users;
- **Clear timeline:** if one of the users clears the entire timeline, by selecting the correspondent button (Use Case 14), all frames from the timelines of all other connected users will be deleted.

4.1.4 Module 3: Content Visualization

Multiple screen support was always a very desired feature among Unity game developers. Connecting external screens supporting Full HD to iOS devices is an important part of creating immersive user experience in native games and multimedia applications. iOS SDK makes supporting this feature really easy for native applications but with Unity3D engine this is was not an easy task. By the beginning of the development of this dissertation Unity3D did not support rendering to multiple targets which meant it only could create one rendering window. And if we wanted an iPad to be one of the visualization displays, the problem got even more complicated because there

were some options available to mirror the iPad screen to an external display but span it so we could show a different portion of Unity's rendering scene in each display was extremely tricky.

To overcome this problem some solutions were analysed:

- *Render a single really-wide image:* the goal was to create a scene with a custom resolution that corresponded to the sum of the resolution of all connected devices so the image spanned across the available displays. In order to detect the connection/disconnection of external displays this solution had to be integrated with plugins. When developing games in Unity3D it is often necessary to access platform specific features, hardware or anything not available to Unity via the API. Plugins are a way to establish a bridge between the code in Unity (C#, Javascript or Boo) and the native platform. It allows the access to any non-supported features by creating a native binary bundle which can be accessed through an interface in the Unity script. There are some examples of this kind of implementation but displays were always connected to a desktop PC and not to the iPad. A prototype of this solution was deployed but only successfully for multiple desktop screens, not involving an iPad.

- *Run different instances of the application on each screen and make then interact:* in other words, we would have a separate executable assigned to each screen. The communication between them could be implemented via text files or via multiplayer options such as using localhost.

However, none of the above solutions proved to be ideal. Fortunately, during the development of the work, a new version of Unity3D (Unity 4.1) became available and with it, new features, in particular full multi-screen support using AirPlay [93]. AirPlay is a protocol stack developed by Apple Inc. that allows wireless streaming of audio, video and photos, together with related metadata between devices. It allows to wirelessly stream what is on one iOS device to an HD TV and speakers via the digital media receiver Apple TV. This feature is only properly implemented for iOS but this was exactly the solution for the multi-screen support problem in question. A new class "Display" shows an array of connected displays and allows selective rendering for each one so it is possible to render one camera for the iPad and another to an external display connected via Apple TV. The iPad can now be used as a game controller, running and controlling the game from the iOS handheld device whilst the action unfolds on a a bigger screen.

Chapter 5

Results and Validation

A series of experiments was conducted in order to validate the system deployed for this dissertation. It was evaluated if this new approach based on mobile devices was more intuitive that the alternative based on the traditional computer with mouse and it was also tested the relevance and utility of the collaborative environment. This was done by computing the time and effort it took for users to create a particular facial expression in the several experimental scenarios and with questionnaires about subjective usability opinion. The target audience chosen comprised people with no previous experience with modeling and animation tools but with relatively good familiarity with electronic devices in order to validate if the system was appropriate for novice users. This chapter describes the experiments conducted, presents their results and exposes the conclusions that can be drawn from them.

5.1 Experiment Design

During the testing stage, two major experiments were performed: an *interface experiment* and a *collaborative environment experiment*. The interface experiment was designed to evaluate the effectiveness of the interface device chosen to deploy the application. It intended to analyse if mobile touch-capable devices such as tablets and, in this particular case, the iPad, are more natural, intuitive and simpler to use than the traditional computer whose interaction model is based on mouse clicks. Therefore, the following research questions were formulated:

- *Q1: Are mobile device based approaches, with tactile interfaces, as easy and intuitive for creating facial expressions with a sketch-based method as the traditional computer/mouse based approaches?*
- *Q2: Is the necessary time to create facial expressions using the iPad's touch-capable interface and the computer with a mouse input device similar?*

The collaborative environment experiment evaluated if such an environment where multiple users can deform, at the same time, the same model, can benefit the task of creating facial expressions. Therefore, the following research question was formulated:

- *Q3: A collaborative facial animation environment is as useful as an individual approach for the task of creating facial expressions?*

To collect data to answer the research questions, some objective measurements were performed: it was recorded the time and number of strokes, control point movements and canvas resets users made to create a pose. Subjective usability data was collected with a questionnaire, that can be observed in Appendix A, comprising four questions related to user background, three questions about the interface experiment, two questions about the collaborative environment experiment and two open questions about improvements suggestions and additional comments.

5.1.1 Participants

Both experiments were performed with a group of 22 people with good or very good level of familiarity with electronic devices (such as computers, tablets, smartphones, etc) and none or poor experience with modeling and animation tools. The participants had an average age of 29,77 years (SD = 3,74) and relatively balanced sex distribution, with 9 males and 13 females performing the experiments as can be seen on table 5.1.

Table 5.1: Information about the experiments participants.

	Sex		Age	
	Male	Female	Av	SD
Participants	9	13	29,77	3,74

5.1.2 Interface Experiment

The interface experiment began with an introductory description of the application and its development context as well as a brief explanation of the tests to be performed. Then, two tests were conducted, one for each evaluated interface device: the computer and the iPad. Each participant performed the tests individually in sessions of about 10 to 20 minutes.

The computer test was composed of two phases:

- **Training phase:** the first contact of the participants with SketchFACE. They were given no explanation about the application and asked to explore the interface during 2 minutes.
- **Task Phase:** An explanation about every functionality of the application was given to the participants (2 minutes) and then they were asked to reproduce a particular facial expression (Figure 5.1). There was no time limit to conclude the task and the only direction provided was that the users needed to copy, as faithfully as possible, the given expression.

Then, an iPad with the application was presented to the participants. After a short explanation about the new interaction model allowed by this device (based on strokes drawn with the finger) and the specific functionalities featured on the mobile device (multi-display support), the test had also two phases:

- **Training phase:** the participants had 2 minutes to explore and get familiar with the new interface and interaction model;
- **Task Phase:** the participants were were asked to reproduce the same facial expression copied in the previous test (Figure 5.1). There was no time limit to conclude de task and the only direction provided was that the users needed to copy, as faithfully as possible, the given expression.

Figure 5.1: Facial expression that the participants were asked to reproduce using the SketchFACE application, both on the computer and on the iPad.

5.1.3 Collaborative Environment Experiment

The same participants of the previous experiment were now paired up. The experiment was composed by 3 tests where users were asked to reproduce the 3 facial expressions of figure 5.2, being the first one the same already copied in the previous experiment. Users had to work together, deforming at the same time the same model, in the collaborative environment provided by SketchFACE. There was no time limit for neither one of the tests and the only rule was that the user should reproduce, as faithfully as possible, the given expressions.

5.2 Experiment Results

In order to answer question Q1 and to try to understand what type of devices are more intuitive as interfaces for creating facial expressions with a sketch-based method, question 2.1.1 of the questionnaire was analysed:

- *From the following options, choose the one that best matches your opinion:*

  a) I consider the computer interface easier and more intuitive.

Figure 5.2: Facial expressions that the participants were asked to reproduce using the SketchFACE application, in the collaborative environment mode.

b) I consider the iPad interface easier and more intuitive.

The results can be seen in figure 5.3.

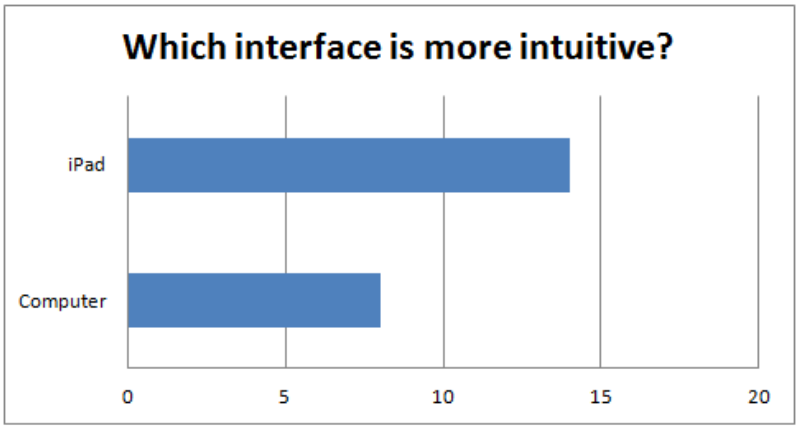

Figure 5.3: Distribution of the answers to the question about what interface device was more intuitive.

The analysis of the chart shows that the participants consider the iPad easier and more intuitive. This way, the null hypothesis is rejected and the alternative hypothesis, that there is significant difference between the level of intuitiveness of both devices is different is accepted, being the iPad considered the more intuitive interface. User's activities, evaluated through the analysis of the experience's log that recorded every stroke created, control point moved, complete reset made or single canvas reset performed, also support this conclusion. Although with a relatively small margin, users had to perform less actions on the iPad to achieve the same results as can be seen in figure 5.4.

In order to answer question Q2 and assess the existence of significant difference between the time spent creating a facial expression on the iPad and on the computer, a difference of means test was performed. Given the relatively small number of cases evaluated (N = 22) and the fact that

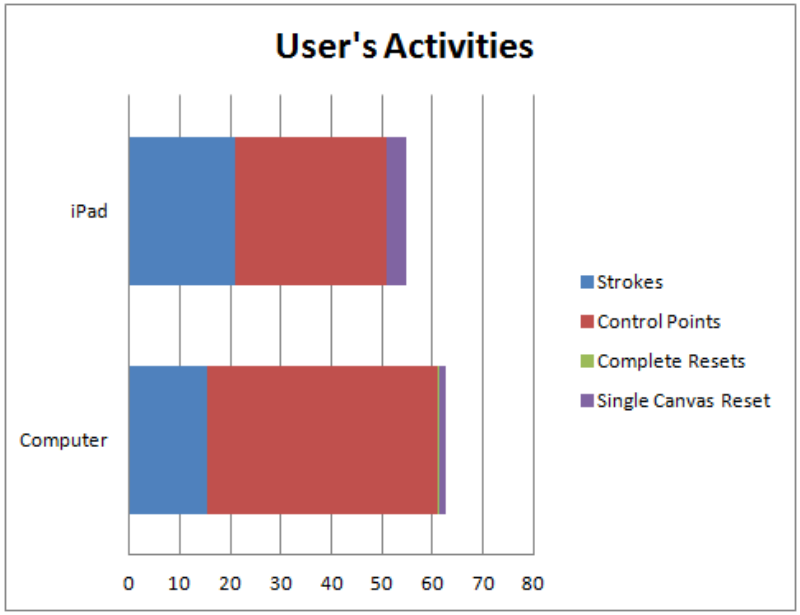

Figure 5.4: Average number of user actions during the interface experiment by category.

a normality test indicated a non-normal distribution of the results, the time comparison between both device interfaces was performed with a non-parametric Mann-Whitney test whose results can be seen on table 5.2.

Table 5.2: Results of the Mann-Whitney test performed.

	Av	SD	Min - Max (seconds)	U	p
Computer	237,41	107,05	112 - 512	188,00	n.s.
iPad	187,91	101,73	55 - 431		

The analysis of table 5.2 indicates that the average time to create a facial pose with SketchFACE on the iPad was smaller than on the computer (as can also be seen on figure 5.5). However, time differences between both interface devices were not statistically significant ($U = 188{,}00$; $p > 0{,}05$). This way, the null hypothesis that states that there are no significant differences between the necessary time to create facial expression using the iPad's touch-capable interface or the computer must be accepted.

Regarding the collaborative environment, in order to evaluate if it made the task of creating a specific facial expression easier, question 2.2.1 from the questionnaire was analysed:

- *From the following options, choose the one that best matches your opinion:*

  a) I consider that the collaborative environment eases the accomplishment of the task.

  b) I consider that the collaborative environment does not ease the accomplishment of the task.

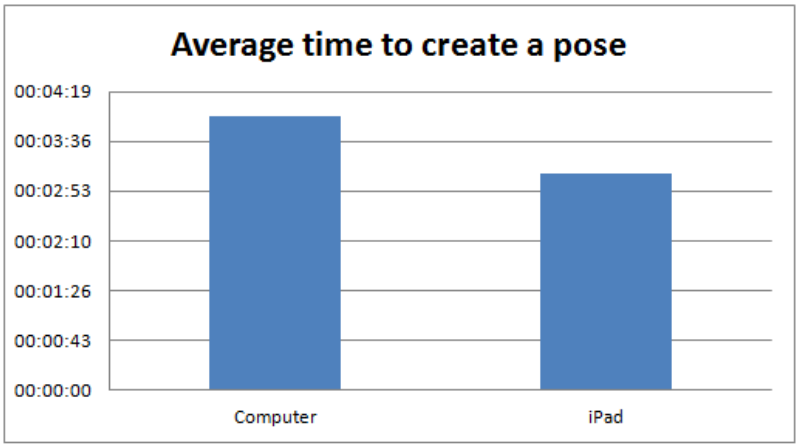

Figure 5.5: Average time necessary to create a given facial pose with SketchFACE on both interface devices.

Results show that 100% of the participants considered that it made the task of creating a particular facial expression easier so the null hypothesis is rejected and the alternative one accepted. The average time users spent to reproduce, on the collaborative environment, the same expression from the interface experiment was slightly smaller than any other previous result (as can be seen on figure 5.6), supporting the same conclusion. This could be due to the fact that participants had now learned how to create that particular expression but when confronted with different facial expressions (poses 2 and 3) time results were even better.

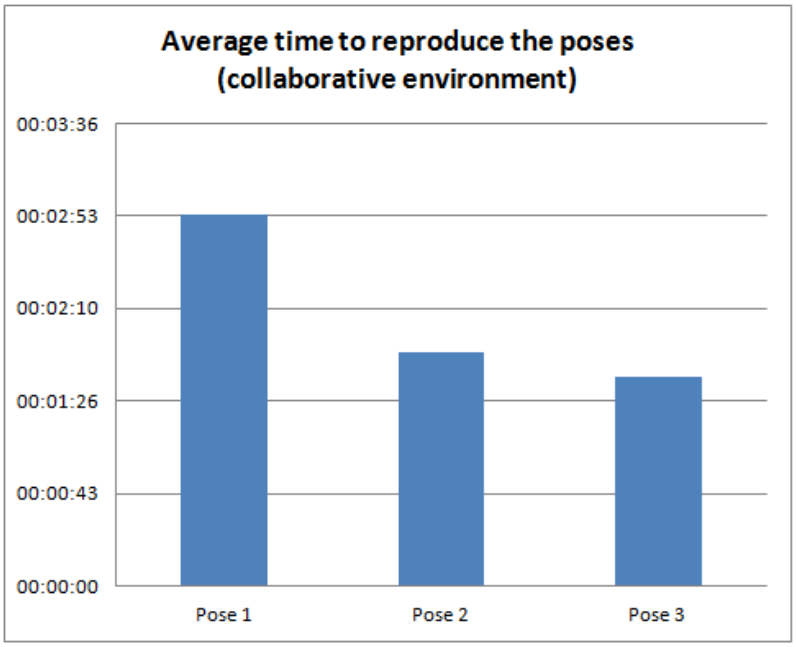

Figure 5.6: Average time necessary to create a given facial pose with SketchFACE on both interface devices.

Regarding the number of actions users had to perform to achieve the target poses, results can be seen on figure 5.7.

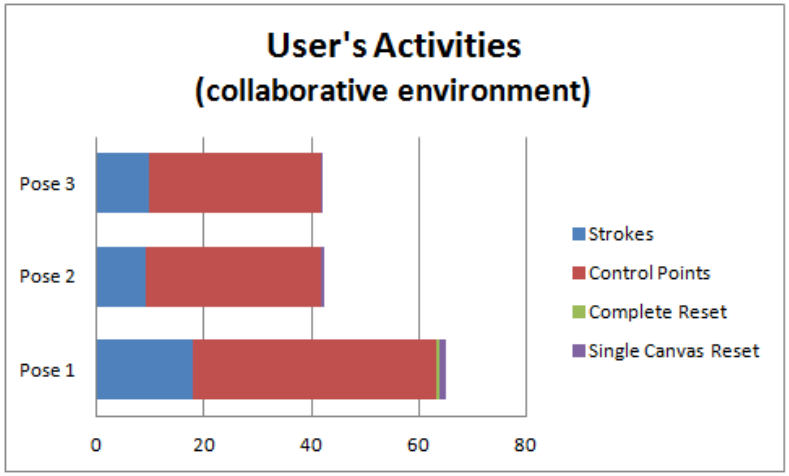

Figure 5.7: Average number of user actions during the collaborative environment experiment, by category, for the three tested expressions.

5.3 Discussion of the Results

The results of the experiments proven to be according to the desired results, even though more general conclusions could not be drawn from them due to statistic insignificance. The majority of the participants chose the iPad as the more intuitive interface probably because the touch-screen mimics more naturally the experience of drawing on paper as many users found it easier to draw curves with a finger than with the mouse. However, the fact that the computer was the first interface presented to the participants could also have influenced their answers, as they were already more familiar with the application by the time the iPad was given to them. Future experiments, besides being performed with a wider number of participants, should use a different group of people for each tested scenario (the iPad and the computer), in order to remove the users learning factor. Other option could also be to change the order in which users test both devices so that a portion of the participants use the iPad first and other use the computer first.

Regarding the absence of significant difference between the necessary time to reproduce facial expressions using SketchFACE on the computer and on the iPad, it could be explained with the poor familiarity of the participants with the touch-interface of the iPad. Tablets are recent devices that have not been as widespread as computers so it is natural that the majority of the population is more used to manipulate the computer's mouse rather than a touch-screen.

Chapter 6

Conclusion and Future Work

This dissertation describes a hybrid sketching and dragging rigging control system that takes advantage of the portability and intuitiveness of modern mobile devices to significantly ease and allow collaborative work on the deformation and animation processes of 3D models. The developed application - SketchFACE - was deployed for Apple's iPad using Unity game engine that allowed simple integration on multiplayer and multi-device functionalities. This chapter summarizes the main conclusions extracted from the research done throughout the course of this dissertation and defines some guidelines for future work.

6.1 Conclusion

Traditional modeling and animation software enable experienced and very trained artists to produce realistic animations but high-quality results are usually beyond the reach of novice users since that deforming and giving life to believable characters requires significant time and effort.

It was the goal of this dissertation to explore a new approach that could open new opportunities for non-expert users in the field of facial animation. By taking advantage of new user interfaces that can be considered more natural, as they aim to enable the user to interact with computers in the same way they interact with the world, an attempt was made to ease the modeling and animation processes of 3D characters. More specifically, it was analysed the possibility of using mobile devices as interfaces to control the deformation and animation of 3D characters.

This way, a new application - SketchFACE (Sketch-based Facial Animation Collaborative System) - was deployed for Apple's iPad. It combines the possibility of controlling a model's deformation, using simple 2D strokes, with the simplicity and intuitiveness of drawing in a touch-capable interface, that mimics more closely the feeling of drawing on paper by allowing direct input using one's finger instead of relying on an external input device such as the mouse. SketchFACE is an application developed in C# using Unity game engine, based on a pre-existing sketch-based algorithm created by Miranda et al. [13], that presents to the user a set of 2D canvases where he can draw in order to control different regions of the model's face. Once a pose is created it can be saved to a timeline and sets of saved poses can be animated in a very simple and intuitive way.

Modeling and animation processes can be done individually or in a collaborative environment where multiple users deform, at the same time, the same model and results can be seen in a small portion of the iPad's screen or in a much wider HD TV connected via Apple TV.

Thus, SketchFACE results in an hybrid sketching and dragging facial animation control system with multiplayer and multi-device functionalities. It was inspired and, at the end, integrated in the LIFEisGAME prototype application - a serious game to help autistic children to recognize and express emotions through facial expressions.

SketchFACE was tested and validated with a group of individuals with no previous experience with modeling and animation tools, that correspond to the target audience of the new application. The performed experiments had a dual focus: evaluating if the new approach based on mobile devices was more intuitive than the traditional computer and mouse based approach and testing the relevance and utility of the collaborative environment. This was done by computing the time and effort it took for users to reproduce given facial expressions, first alone and then in collaboration with other participants and by analysing their answers to a subjective usability questionnaire.

Although some experiments led to non statistically relevant conclusions, the results of the tests show that a majority of the participants consider the iPad a more intuitive interface for modeling tasks when compared to the computer and all the participants considered that the collaborative environment made the task of creating particular facial expressions easier.

To view a video about SketchFACE or to access additional material related to this dissertation please visit the website available at `http://paginas.fe.up.pt/~ee08175`.

6.2 Future Work

This dissertation intended to explore a new approach for facial animation based on mobile devices. It was not possible, nor was under the scope of this work, to implement every feature and test every scenario thought out. Instead, a new door was opened and many improvement paths were created. This document ends with suggestions of potential directions for future work that can be grouped in three major categories: *new features for the application* (some thought out but not implemented due to time constraints and some suggested by the participants that took part on the validation experiments), *improvements to the deployed functionalities* and *hints for further validation experiments*.

- **New features**
  - **Possibility to hide control points:** it would be interesting to have the possibility to show and hide the control points as, sometimes, they interfere with the stroke drawing action;
  - **More control areas:** some facial regions, as the cheeks and the ears, can not be controlled in the current version of SketchFACE. This feature could easily be implemented

by adding more 2D control canvases since the model rig used has already bones associated with these areas. However, increasing the number of control canvases would difficult their placement in the interface making it harder to combine intuitive positioning with acceptable drawing area size.

- **Multi-touch option:** multiple simultaneous input touches to the touch-screen can easily be detected so it would be easy to implement the option of drawing simultaneously with several finger (for example, one participant tried to draw, at the same time, both eyebrows with the two hands but this feature is not currently implemented). It would be interesting to explore the possibility of performing new actions based on multi-touch capabilities;
- **Tutorials and help menus:** although simplicity was a target of the application, to ease even more the usability of SketchFACE some help menus, tutorials or information labels could be added to the application explaining the function of some of the buttons or interface areas. However, this helping content should never be intrusive;
- **Undo option:** many participants of the validation experiments expressed the desire for the possibility to revert the last change done to the model instead of only being able to reset the entire canvas or the entire pose;
- **Skin color alteration:** besides controlling the deformation of the different regions of the face, the application could also include the ability to alter skin pigmentation of some areas to increase faithfulness to real facial expressions.

- **Functionalities improvements**

  - **Constraints:** due to current difficulty to automatically import constraints defined in the 3D model to Unity3D game engine, an approach that used an auxiliary XML file was used. However, this approach is not nearly ideal since it currently imposes rectangular limits for joint movements which doesn't reproduce real bone movement limitations. If constraints remain impossible to import directly, they should be implemented in Unity3D for better results;
  - **Control points:** the control points were a functionality added to the application in order to allow individual rig control point alterations, resulting in more accurate deformations. However, the way they were implemented seemed to difficult its original purpose. As they were relatively small, it was hard to handle them in the touch interface of the iPad causing some unexpected results. For better results, a different interaction model could be used that altered the curve's shape without the need to select a particular control point by "clicking" on it. Small strokes drawn inside the curve's bounding box could deform the existing reference curve instead of starting a new and different stroke. This was the method implemented in Miranda et al.'s work [13] that seems a good further development for SketchFACE.

- **Further experiments**
  - **Larger group of participants:** the performed validation experiments ended up in many non statistically significant results. The outcome could have been different if a wider number of participants had been used and if different groups would have been used for each tested scenario. Further experiments can explore these options;
  - **Different user profiles:** it could have been interesting to widen the experiments to multiple groups of people with different profiles. The performed tests only targeted people with good or very good level of familiarity with electronic devices, none or poor experience with modeling and animation tools and average age of 30 years. Using also participants with different age groups or with prior experience with animation tools can add value to the results obtained from these experiments.
  - **Further evaluation of the collaborative environment:** although the system was implemented to support more than two simultaneous users, no tests were conducted in order to validate other scenarios with more participants. As some aspects of the collaborative environment need further testing, future experiments can address this topic.

Appendix A

Validation Experiments Documents

A.1 Usability Questionnaire

The following pages show the usability questionnaire presented to the participants of the validation experiments performed on SketchFACE.

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

ID: _____

SKETCH-BASED FACIAL MODELING AND ANIMATION: AN APPROACH BASED ON MOBILE DEVICES

Ana Luísa de Vila Fernandes Orvalho
ana.orvalho@gmail.com | ee08175@fe.up.pt

Orientador: Prof. Augusto Sousa (FEUP) | **Co-orientadora:** Prof. Verónica Orvalho (FCUP)

Introdução

- O presente questionário pretende, do ponto de vista do utilizador, avaliar a usabilidade e validar a aplicação SketchFACE, desenvolvida no contexto de uma dissertação do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores.
- As respostas são completamente voluntárias e confidenciais.

Descrição do Projeto

Animação facial que satisfaça as expectativas dos espectadores é muito difícil de se alcançar. O tradicional *software* de modelação e animação de modelos 3D, apesar de muito poderoso, é bastante complexo e exige um tempo de aprendizagem significativo para se conseguir explorar e dominar os seus múltiplos menus e funcionalidades.

Esta dissertação propõe o uso de uma abordagem baseada em desenhos para manipular a estrutura de controlo dos modelos 3D, o *rig*, num ambiente colaborativo, em que vários utilizadores podem unir esforços para simultaneamente deformar e animar uma personagem 3D. Foi desenvolvido um protótipo de uma nova aplicação, designada SketchFACE (*Sketch-based Facial Animation Collaborative Environment*), especialmente orientado para o iPad. Aproveitando a interface tátil "mais natural" deste dispositivo móvel, tentou-se criar novas oportunidades para utilizadores com pouca ou nenhuma experiência, alargando, assim, o público-alvo dos sistemas de animação enquanto se explora a viabilidade do trabalho colaborativo nos processos de modelação e animação.

A solução proposta foi inspirada no projeto LIFEisGAME – um jogo sério que pretende ajudar crianças com problemas do espectro do autismo a reconhecer e expressar emoções através de expressões faciais de forma interativa e creativa.

1) Perfil do Utilizador

1.1) Idade: _____ Sexo: M ☐ F ☐

1.2) Nível de familiarização com dispositivos eletrónicos (computadores, *tablets*, *smartphones*…)

☐ a) Nenhum ☐ c) Bom

☐ b) Pouco ☐ d) Muito Bom

1.3) Nível de experiência com ferramentas de modelação e animação facial.

☐ a) Nenhum ☐ c) Bom

☐ b) Pouco ☐ d) Muito Bom

1.4) Background artístico

☐ a) Nenhum ☐ c) Bom

☐ b) Pouco ☐ d) Muito Bom

2) Sobre o protótipo SketchFACE

2.1) Interface com o utilizador

2.1.1) Das seguintes opções escolha a que melhor corresponde à sua opinião:

☐ a) Considero que a interface do computador é mais fácil e intuitiva de usar.

☐ b) Considero que a interface do iPad é mais fácil e intuitiva de usar.

2.1.2) Considera a interface da aplicação suficientemente clara? Sim ☐ Não ☐

2.1.3) Se não, porquê? _____

_____

_____

2.2) Ambiente colaborativo

2.2.1) Das seguintes opções, escolha a que melhor corresponde à sua opinião:

☐ a) Considero que o ambiente colaborativo facilita a realização da tarefa proposta.

☐ b) Não considero que o ambiente colaborativo facilite a realização da tarefa.

2.2.2) Das seguintes opções, escolha a que melhor corresponde à sua opinião:

☐ a) Considero que esta ferramenta é inovadora.

☐ b) Não considero que esta ferramenta seja inovadora.

3) Registe aqui a sua apreciação global sobre este protótipo SketchFACE

3.1) Sugestões de melhoria: _____

_____

_____

3.2) Comentários gerais: _____

_____

_____

Data: _____ /_____/_____

Obrigada pela sua participação ☺

References

[1] Zhigang Deng and Junyong Noh. Computer facial animation: A survey. In Zhigang Deng and Ulrich Neumann, editors, *Data-Driven 3D Facial Animation*, pages 1–28. Springer London, 2007. URL: http://dx.doi.org/10.1007/978-1-84628-907-1_1.

[2] S. Villagrasa, A. Susín Sánchez, et al. Face! 3d facial animation system based on facs. 2010.

[3] Rob Bredow, David Schaub, Daniel Kramer, Matthew Hausman, Danny Dimian, and R. Stirling Duguid. Surf's up: the making of an animated documentary. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 1–123, New York, NY, USA, 2007. ACM. URL: http://doi.acm.org/10.1145/1281500.1281600, doi:10.1145/1281500.1281600.

[4] O. Alexander, M. Rogers, W. Lambeth, M. Chiang, and P. Debevec. Creating a photoreal digital actor: The digital emily project. In *Visual Media Production, 2009. CVMP'09. Conference for*, pages 176–187. IEEE, 2009.

[5] David Komorowski, Vinod Melapudi, Darren Mortillaro, and Gene S. Lee. A hybrid approach to facial rigging. In *ACM SIGGRAPH ASIA 2010 Sketches*, SA '10, pages 42:1–42:2, New York, NY, USA, 2010. ACM. URL: http://doi.acm.org/10.1145/1899950.1899992, doi:10.1145/1899950.1899992.

[6] W. GRUBB. Facial animation rig for delgo. *Tutorial Notes*, 12, 2009.

[7] K. Balci. Xface: Mpeg-4 based open source toolkit for 3d facial animation. In *Proceedings of the working conference on Advanced visual interfaces*, pages 399–402. ACM, 2004.

[8] B. Paulson, P. Rajan, P. Davalos, R. Gutierrez-Osuna, and T. Hammond. What!?! no rubine features?: using geometric-based features to produce normalized confidence values for sketch recognition. In *HCC Workshop: Sketch Tools for Diagramming*, pages 57–63, 2008.

[9] LIFEisGAME. Learning of facial emotions using serious games, 2010. URL: http://www.portointeractivecenter.org/lifeisgame/ [last accessed 2013-01-27].

[10] FCT. Fundação para a ciência e a tecnologia. URL: http://www.fct.pt/ [last accessed 2013-01-27].

[11] UT Austin|Portugal. Internationa collaboratory for emerging technologies (colab). URL: http://utaustinportugal.org/ [last accessed 2013-01-27].

[12] IT. Instituto de telecomunicações. URL: http://www.it.up.pt/ [last accessed 2013-01-27].

[13] José Carlos Miranda, Xenxo Alvarez, João Orvalho, Diego Gutierrez, A. Augusto Sousa, and Verónica Orvalho. Sketch express: A sketching interface for facial animation. *Computers*

& *Graphics*, 36(6):585 – 595, 2012. URL: http://www.sciencedirect.com/science/article/pii/S0097849312000416, doi:10.1016/j.cag.2012.03.002.

[14] Frederick I. Parke. Computer generated animation of faces. In *Proceedings of the ACM annual conference - Volume 1*, ACM '72, pages 451–457, New York, NY, USA, 1972. ACM. URL: http://doi.acm.org/10.1145/800193.569955, doi:10.1145/800193.569955.

[15] K. Ritchie, J. Callery, and K. Biri. *The Art of Rigging: A definitive guide to character technical direction with Alias Maya*. CG Toolkit, 2005.

[16] M. Mori. The uncanny valley, translated by karl f. macdorman and takashi minato. *Energy*, 7(4):33–35, 1970.

[17] D. Hanson, A. Olney, S. Prilliman, E. Mathews, M. Zielke, D. Hammons, R. Fernandez, and H. Stephanou. Upending the uncanny valley. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1728. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[18] Canetti Y. Piretti M. Schleifer J., Scaduto-Mendola R. Character setup from rig mechanics to skin deformations: A practical approach. SIGGRAPH '02 Course Notes, 2002.

[19] Nikolaos Ersotelos and Feng Dong. Building highly realistic facial modeling and animation: a survey. *The Visual Computer*, 24:13–30, 2008. URL: http://dx.doi.org/10.1007/s00371-007-0175-y, doi:10.1007/s00371-007-0175-y.

[20] V. Orvalho, P. Bastos, F. Parke, B. Oliveira, and X. Alvarez. A facial rigging survey. In *Eurographics 2012-State of the Art Reports*, pages 183–204. The Eurographics Association, 2012.

[21] T. McLaughlin and S.S. Sumida. The morphology of digital creatures. In *ACM SIGGRAPH*, pages 05–09, 2007.

[22] V.C.T. Orvalho. *Reusable facial rigging and animation: Create once, use many*. PhD thesis, Universitat Politècnica de Catalunya, 2007.

[23] J. Fordham. Middle earth strikes back. *Cinefex*, 2003.

[24] Pushkar Joshi, Wen C. Tien, Mathieu Desbrun, and Frédéric Pighin. Learning controls for blend shape based realistic facial animation. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM. URL: http://doi.acm.org/10.1145/1198555.1198588, doi:10.1145/1198555.1198588.

[25] A. Ward. *Game character development with maya*. New Riders, 2005.

[26] N. Burtnyk and M. Wein. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Commun. ACM*, 19(10):564–569, October 1976. URL: http://doi.acm.org/10.1145/360349.360357, doi:10.1145/360349.360357.

[27] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation, 2000.

[28] T.W. Sederberg and S.R. Parry. Free-form deformation of solid geometric models. In *ACM Siggraph Computer Graphics*, volume 20, pages 151–160. ACM, 1986.

[29] S. Coquillart. *Extended free-form deformation: a sculpturing tool for 3D geometric modeling*, volume 24. ACM, 1990.

[30] Prem Kalra, Angelo Mangili, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of facial muscle actions based on rational free form deformations. *Computer Graphics Forum*, 11(3):59–69, 1992. URL: http://dx.doi.org/10.1111/1467-8659.1130059, doi:10.1111/1467-8659.1130059.

[31] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 405–414. ACM, 1998.

[32] M.A. Warburton. Physically-based facial modelling and animation with wrinkles: 12 month report. 2011.

[33] Stephen M. Platt and Norman I. Badler. Animating facial expressions. *SIGGRAPH Comput. Graph.*, 15(3):245–252, August 1981. URL: http://doi.acm.org/10.1145/965161.806812, doi:10.1145/965161.806812.

[34] J. Osipa. *Stop staring: facial modeling and animation done right*. Sybex, 2010.

[35] John Lasseter. Principles of traditional animation applied to 3d computer animation. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 35–44, New York, NY, USA, 1987. ACM. URL: http://doi.acm.org/10.1145/37401.37407, doi:10.1145/37401.37407.

[36] Frederic Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing realistic facial expressions from photographs. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM. URL: http://doi.acm.org/10.1145/1185657.1185859, doi:10.1145/1185657.1185859.

[37] K. Waters and T. Levergood. An automatic lip-synchronization algorithm for synthetic faces. In *Proceedings of The second ACM international conference on Multimedia*, pages 149–156. ACM, 1994.

[38] F.I. Parke. A parametric model for human faces. Technical report, DTIC Document, 1974.

[39] Jun yong Noh and Ulrich Neumann. A Survey of Facial Modeling and Animation Techniques. 1998.

[40] D. Bennett. The faces of the polar express. In *ACM Siggraph 2005 Courses*, page 6. ACM, 2005.

[41] V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. *Computer Graphics Forum*, 22(3):641–650, 2003. URL: http://dx.doi.org/10.1111/1467-8659.t01-1-00712, doi:10.1111/1467-8659.t01-1-00712.

[42] W.L. Wooten and J.K. Hodgins. Simulating leaping, tumbling, landing and balancing humans. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 656–662. IEEE, 2000.

[43] I.D. Horswill. Lightweight procedural animation with believable physical interactions. *Computational Intelligence and AI in Games, IEEE Transactions on*, 1(1):39 –49, march 2009. doi:10.1109/TCIAIG.2009.2019631.

[44] I.S. Pandzic and R. Forchheimer. Mpeg-4 facial animation. *The standard, implementation and applications. Chichester, England: John Wiley&Sons*, 2002.

[45] P. Ekman and W. V. Friesen. Facial action coding system: a technique for the measurement of facial movement. 1978.

[46] P. Ekman, W.V. Friesen, and J.C. Hager. Facial action coding system. *A Human Face*, 2002.

[47] Irfan Essa, Sumit Basu, Trevor Darrell, and Alex Pentland. Modeling, tracking and interactive animation of faces and heads using input from video. In *IN PROCEEDINGS OF COMPUTER ANIMATION CONFERENCE*, pages 68–79. IEEE Computer Society Press, 1996.

[48] Achim Ebert, NahumD. Gershon, and GerritC. Veer. Human-computer interaction. *KI - Künstliche Intelligenz*, 26:121–126, 2012. doi:10.1007/s13218-012-0174-7.

[49] Daniel Wigdor and Dennis Wixon. *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2011.

[50] R. Harper. *Being human: Human-computer interaction in the year 2020*. Microsoft Research, 2008.

[51] Microsoft. Surfice sdk. URL: http://msdn.microsoft.com/en-us/library/ff727815(v=Surface.20).aspx [last accessed 2013-02-04].

[52] NUIGroup. Touchlib. URL: http://nuigroup.com/touchlib/ [last accessed 2013-02-04].

[53] Carl Kenner. Glovepie. URL: http://glovepie.org/glovepie.php [last accessed 2013-02-04].

[54] Cycling '74. Max/msp/jitter. URL: http://www.cycling74.com/ [last accessed 2013-02-01].

[55] vvvv: a multipurpose toolkit. URL: http://vvvv.org/ [last accessed 2013-02-01].

[56] N. Metha. A flexible machine interface. *MA Sc. Thesis, Department of Electrical Engineering, University of Toronto*, 1982.

[57] Daniel Wigdor, Joe Fletcher, and Gerald Morrison. Designing user interfaces for multi-touch and gesture devices. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 2755–2758, New York, NY, USA, 2009. ACM. URL: http://doi.acm.org/10.1145/1520340.1520399, doi:10.1145/1520340.1520399.

[58] B. Buxton. Multi-touch systems that i have known and loved. *Microsoft Research*, 2007.

[59] Circle Twelve Inc. Diamondtouch. URL: http://www.circletwelve.com/products/diamondtouch.html [last accessed 2013-02-04].

[60] P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226. ACM, 2001.

[61] SMART Technologies Inc. Smart table 230i. URL: http://smart.lsk.hu/edu/tamogatas/letoltes/pdf/adatlapok/st230i.pdf [last accessed 2013-02-04].

[62] SMART Technologies Inc. Smart table 442i. URL: http://downloads01.smarttech.com/media/sitecore/en/pdf/products/table/mktg-913-rev05-ed-fs-table_final.pdf [last accessed 2013-02-04].

[63] Microsoft. Pixelsense. URL: http://www.microsoft.com/en-us/pixelsense/default.aspx [last accessed 2013-02-04].

[64] Apple Inc. ipad. URL: http://www.apple.com/ipad/features/ [last accessed 2013-02-04].

[65] Microsoft. Microsoft surface. URL: http://www.microsoft.com/Surface/en-US [last accessed 2013-02-04].

[66] Google. Nexus 7. URL: http://www.google.com/nexus/7/ [last accessed 2013-02-04].

[67] Google. Nexus 10. URL: http://www.google.com/nexus/10/ [last accessed 2013-02-04].

[68] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.

[69] Microsoft. Kinect. URL: http://www.xbox.com/en-US/kinect [last accessed 2013-02-04].

[70] Leap Motion. Leap. URL: https://www.leapmotion.com/ [last accessed 2013-02-04].

[71] L. Olsen, F.F. Samavati, M.C. Sousa, and J.A. Jorge. Sketch-based modeling: A survey. *Computers and Graphics (Pergamon)*, 33(1):85–103, 2009. cited By (since 1996) 48. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-59249095193&partnerID=40&md5=79316233b31b419e83ca721757c11dfa.

[72] S. Saga. A freehand interface for computer aided drawing systems based on the fuzzy spline curve identifier. In *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, volume 3, pages 2754 –2759 vol.3, oct 1995. doi:10.1109/ICSMC.1995.538199.

[73] Tevfik Metin Sezgin, Thomas Stahovich, and Randall Davis. Sketch based interfaces: early processing for sketch understanding. In *Proceedings of the 2001 workshop on Perceptive user interfaces*, PUI '01, pages 1–8, New York, NY, USA, 2001. ACM. URL: http://doi.acm.org/10.1145/971478.971487, doi:10.1145/971478.971487.

[74] Tevfik Metin Sezgin, Thomas Stahovich, and Randall Davis. Sketch based interfaces: early processing for sketch understanding. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM. URL: http://doi.acm.org/10.1145/1185657.1185783, doi:10.1145/1185657.1185783.

[75] Tevfik Metin Sezgin and All Davis. Scale-space based feature point detection for digital ink. In *In Making Pen-Based Interaction Intelligent and Natural, AAAI Spring Symposium*, 2004.

[76] Ivan E. Sutherland. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop*, DAC '64, pages 6.329–6.346, New York, NY, USA, 1964. ACM. URL: http://doi.acm.org/10.1145/800265.810742, doi:10.1145/800265.810742.

[77] Dean Rubine. Specifying gestures by example. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '91, pages 329–337, New York, NY, USA, 1991. ACM. URL: http://doi.acm.org/10.1145/122718.122753, doi:10.1145/122718.122753.

[78] A. Chris Long, Jr., James A. Landay, Lawrence A. Rowe, and Joseph Michiels. Visual similarity of pen gestures. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '00, pages 360–367, New York, NY, USA, 2000. ACM. URL: http://doi.acm.org/10.1145/332040.332458, doi:10.1145/332040.332458.

[79] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM. URL: http://doi.acm.org/10.1145/1294211.1294238, doi:10.1145/1294211.1294238.

[80] L. Anthony and J.O. Wobbrock. A lightweight multistroke recognizer for user interface prototypes. In *Proc. Graphics Interface*, pages 245–252, 2010.

[81] T. Hammond and R. Davis. Ladder, a sketching language for user interface developers. *Computers & Graphics*, 29(4):518–532, 2005.

[82] T.M. Sezgin and R. Davis. Early processing in sketch understanding. *Unpublished Master's Thesis, Massachusetts Institute of Technology*, 2001.

[83] B. Yu and S. Cai. A domain-independent system for sketch recognition. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 141–146. Citeseer, 2003.

[84] B. Paulson and T. Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 1–10. ACM, 2008.

[85] OberCom. A internet em portugal, 2012. URL: http://www.obercom.pt/client/?newsId=548&fileName=sociedadeRede2012.pdf.

[86] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2012–2017. URL: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html [last accessed 2013-08-28].

[87] Patrick Lindemann. A short report on multi-touch user interfaces. *Department of Media Informatics*, 2010.

[88] Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays, 2003.

[89] Apple. ios human interface guidelines. URL: https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html [last accessed 2013-08-14].

[90] Yusra Masud and M Asir Ajmal. Left-handed people in a right-handed world: A phenomenological study. *Pakistan Journal of Social and Clinical Psychology*, 9(2):49–60, 2012.

[91] Unity Technologies. Unity. URL: http://unity3d.com/ [last accessed 2013-01-30].

[92] Unity Technologies. Unity networking servers. URL: http://www.unity3d.com/master-server/index.html [last accessed 2013-02-25].

[93] Apple. Airplay. URL: http://www.apple.com/airplay/ [last accessed 2013-07-04].