Swiss Federal Institute of Technology Lausanne
School of Computer and Communication Sciences
Computer-Human Interaction in Learning and Instruction

École Polytechnique Fédérale de Lausanne

Paprika

A Framework for Paper-based Interaction

Master Thesis Project

Computer Science

Spring 2014

Author

Carlos Sanchez Witt

Supervisors

Dr. Luis P. Prieto

Prof. Pierre Dillenbourg

CONTENTS

INTRODUCTION

In the decades since the inception of Pong, the first video game to achieve widespread success in 1972, games have developed across a multitude of platforms, from arcades and personal computers, to dedicated home consoles and handheld devices. Today, video games are considered an artistic medium on its own merit, characterized by their interactivity. Video games take on surprisingly different forms, being able to deliver immersive storytelling as well as creating innovative interaction mechanisms.

From the conventional gamepads, joysticks and steering wheels, to touch screens and motion capture devices, video games make great use of novel interaction technologies. In the last decade motion control has become an increasingly popular alternative to button-based input in commercial video games, with the unprecedented success of the Nintendo Wii and the introduction of the Wii Remote, followed by Microsoft's controller-less Kinect.

This multitude of input devices has had an important role in the fields of Human-Computer Interaction and Virtual and Augmented Reality, both as tools for and as the results of extensive research, dating back as far as video games themselves. Many studies of what is known now as Tangible User Interfaces (TUIs) have been carried throughout the years, exploring the potential of physical interfaces for computer systems.

A number of these studies have been carried at the Computer-Human Interaction in Learning and Instruction (CHILI) laboratory at EPFL. Several TUIs have been developed at CHILI with a focus on pedagogical applications. Many of them have exploited the flexibility and affordability of paper through the use of fiducial markers in combination with a camera-projector system for augmented reality applications. Both technologies were developed in-house, their last incarnation consisting of the software library Chilitags and the Tinkerlamp hardware system.

However, despite the increasing availability of such powerful technologies, specialized input devices are expensive to produce and difficult to adapt. A lot of work is still required to build concrete tangible applications. Development tools and environments do not offer the ease of use required by application designers as they lack easy to learn abstractions, and require specialized knowledge in computer vision and computer graphics.

In an attempt to develop both affordable and accessible TUIs, a number of researchers have focused their efforts into paper as the base material on which to build tangible interaction, through developer toolkits for object/shape recognition and fiducial markers, abstraction models and interaction frameworks, and specialized augmented reality browsers. Despite these efforts, it remains difficult to design and develop interactive applications and games, even within CHILI: in the latest experiment conducted as part of the Modelling and Integration of Orchestrated Classrooms through Tangible and paper Interfaces (MIOCTI) project, roughly three man-months where required to develop a simple collaborative game. The mechanics at hand were quite simple (placing paper cards in specific areas of the playfield during a preparation phase, and using another card to inform the simulation to compute the new state of the game), and the technologies used where the aforementioned Chilitags and Tinkerlamp system which are well known and continuously used among CHILI researchers.

This project intends to facilitate the development of tangible paper-based applications and games. In chapter 2, we will develop a picture of the research context through an introduction to TUIs, Augmented Reality and related concepts, both in general and as applied to games and paper interfaces. We will also looks at the landscape of existing commercial video games implementing TUIs and AR into their gameplay, as well as a number of development solutions.

Our contributions will be presented and developed in chapter 3. The first one takes the form of a framework for paper-based interactions, an abstraction model which both eases design and development, as well as communication between designers, artists, and users. The second contribution is a concrete implementation of this model into a framework for real developer use, and finally, a number of games were implemented used this framework.

Chapter 4 will summarize our thoughts on our contribution, where it has succeeded and failed. Finally, chapter 5 will conclude with our views on the potential further development of our research.

Related Work and Background

In order to tackle the difficulties of developing tangible paper-based applications and games, we must first look at the concepts behind it. In this chapter we will introduce the theoretical concepts relating to tangible user interfaces, paper-based interaction and related research. We will also review commercial game-oriented solutions in the field of tangible interaction, as well as existing development technologies for creating such games and applications.

2.1. Tangible User Interfaces

Fitzmaurice introduced the concept of Graspable User Interfaces, later to become Tangible User Interfaces (TUIs), in 1995 as an evolution of conventional input methods for Graphical User Interfaces (GUIs). Unlike typical GUIs, where the mouse is almost the sole input method available to the user, a TUI is characterized by the existence of multiple physical handles attached to different functionalities within the application. By offering concurrent input options to the user, a TUI effectively broadens the scope of potential innovative interactions. Concretely, the user can more easily reorganise the workspace or playfield, application commands and data structures can be modelled into a tangible language which can be more easily understood by an untrained user than menus and icons. TUIs are a new take on the implementation of the Direct Manipulation interaction framework which consists of linking virtual objects and actions to real/world metaphors, and originally inspired GUIs.

TUIs have often times been tightly linked to augmented and virtual reality. Augmented reality consists of the projection of virtual objects, images and sounds generated by computer applications on top of the real world, while virtual reality simulates the presence of the user in a virtual world. AR and VR share a number of technological solutions, including TUIs, as we'll see more in detail in section 2.3.

In 2005, Ulmer et al. published a classification of TUIs which separates them in three dominant (non-exclusive) approaches: Interactive Surfaces, Constructive Assembly, and Token+Constraint.

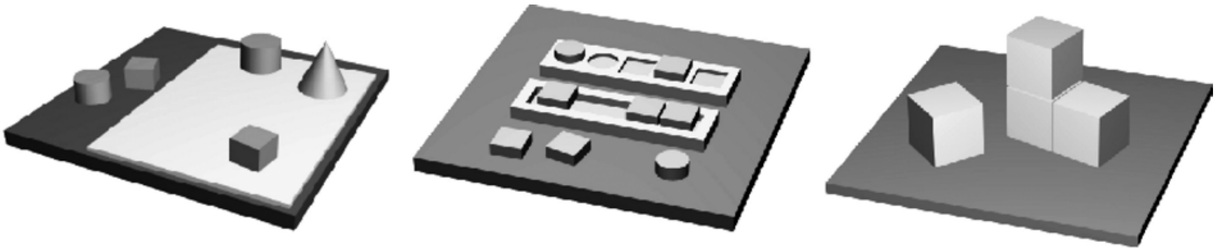


FIGURE 1: ILLUSTRATION OF THE CLASSIFICATION OF TUIS BY ULLMER ET AL.

- *Interactive Surfaces* consider the spatial configuration of objects relative to a reference plane, a working surface. The properties of an object, namely its identity, presence, position, and orientation dictating what type of augmentation is to be applied, and where to project it.

- *Constructive Assembly* considers a modular approach where each physical module is a connectable element. By attaching it to the system, it brings new information and functionality to the simulation. Spatial order usually determines how these modules interact with one another.

- Finally, *Token+Constraint* interface elements take advantage of physical constraints to define a structure in which objects (or tokens) embody different functionalities. By limiting the range of actions the user can apply to tokens, these become interactive elements like knobs or sliders inside a TUI.

Throughout the years, TUIs have taken a number of shapes as research in the fields of interaction and virtual and augmented reality progresses, ranging from expensive haptic gloves to head mounted displays. See Shaer & Hornecker (2010) for a deep survey of existing conceptual frameworks, technologies and their fields of application.

2.2. Paper Interfaces

Paper has been part of human history for more than two millennia, its usage ranging from representative value in banknotes and train tickets, to a means of storing and communicating information in books, newspapers and letters. It has been both a medium of choice to cultivate knowledge, through art and science, as it has been used for entertainment purposes. One of paper’s greatest strength is its malleability, as it can be cut and folded to create a wide range of shapes and sizes.

During the past century, a multitude of technological breakthroughs have led many to believe paper is a thing of the past, a deprecated media to be replaced by computers and touch screens. However, other researchers disagree with this idea, as they believe paper constitutes both a powerful and accessible tool for communication and interaction, the kinds of which modern technology is not quite ready to replace for a number of reasons. Sellen & Harper (2003) argue in “The Myth of the Paperless Office” that computers cannot easily replace all forms of paper. Paper presents a number of affordances over computers in everyday office use (scanning a text, skimming a book, navigating a document, etc.). Paper is also deeply embedded in many well-established practices in society all over the world.

Similarly to how the desktop metaphor proved to be a successful approach to GUIs, paper offers a flexible and familiar solution for TUIs. Wellner embraces paper in what is commonly described as the first augmented paper interface, the DigitalDesk (1993). It consists of an augmented experience where a “digital desk” is projected on top of the traditional desk, allowing interaction with virtual applications through traditional pen and paper manipulations.

Research in the field of paper-based interaction has led to the proposition of multiple classifications for these systems. Bonnard (2012) classifies paper-based systems along a continuum of usages, from situations where content is of most importance, to those where the focus is on tangibility. He further divides this continuum into four, not necessarily mutually exclusive, coarse classes of usages.

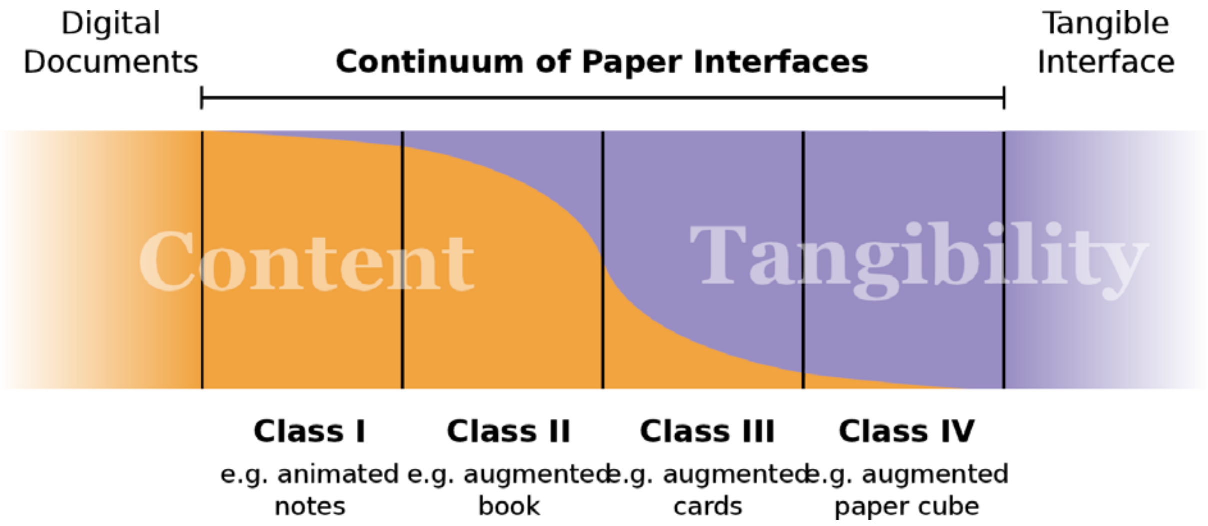


FIGURE 2: THE CONTINUUM OF PAPER INTERFACES BY BONNARD.

- *Class I* defines augmented documents, with typical usages including reading, writing, annotating, printing, filtering, and more.

- *Class II* consists of structured sets of documents, like books, which can be assembled, navigated, dispatched, stacked, etc.

- *Class III* is made of tangible paper, like cards and sheets, which can be flipped, cut, folded, pinched, and more.

- *Class IV* includes all kinds of tangible paper controllers, which can be precisely placed and oriented, rubbed, shaken, grabbed, piled up, etc.

Other researchers have proposed various conceptual frameworks and models for augmented paper applications. For instance, Steimle (2010) presents Letras, a framework for developing pen-and-paper based applications in mobile settings. Interested in the mobility of both the user and the document, this digital pen-based solution falls under Class I and focuses on content.

Cuendet et al. (2011) presents a sequence of pedagogical activities into multiple sheets of paper, each with corresponding printed and augmented information on the exercise. This structured set of documents illustrates Class II.

To Illustrate Class III, we can look at the case of paper cards. The usage of paper cards as input elements for interactive applications date as far back as Perlman's work in 1976, and has been an effective solution throughout the years. More recently Bonnard makes use of cards to manipulate interactive objects in his pedagogical applications for learning geometry (2012).

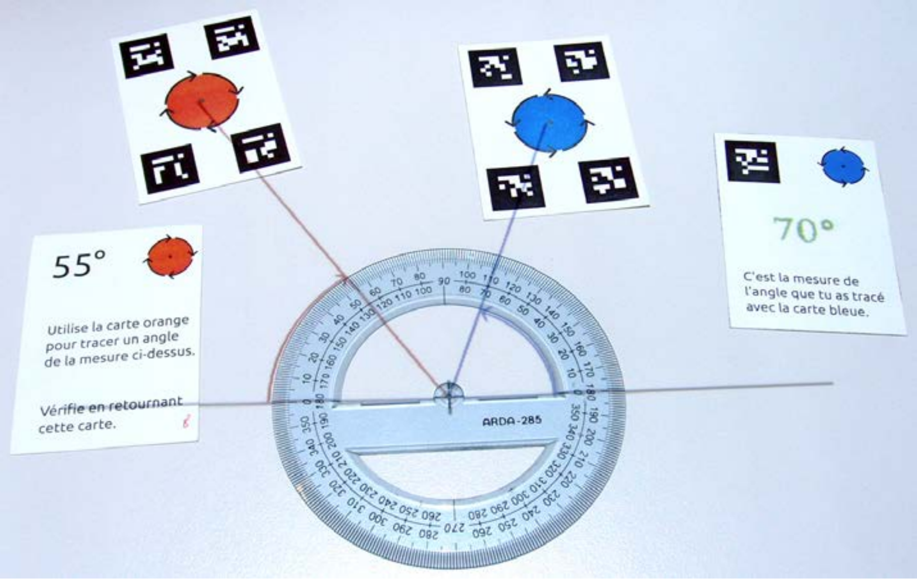

55°
Utilise la carte orange pour tracer un angle de la mesure ci-dessus.
Vérifie en retournant cette carte.
70°
C'est la mesure de l'angle que tu as tracé avec la carte bleue.
ARDA-285

FIGURE 3: ANGLE MEASUREMENT ACTIVITY USING PAPER CARD CONTROLLERS BY BONNARD.

Paper cards thus appear to be a good solution for a large number of tangible applications, as they present a lot potential for engaging gameplay mechanics. Paper cards are ubiquitous and already widely known among the different cultures of the world through countless card games invented throughout history. They shine for their practicality and affordability, they are easy to produce, handle, and store.

2.3. Non Paper-Based Augmented Reality Games

As mentioned earlier, augmented and virtual reality rely to a great extent on tangible user interfaces. As such, commercially available AR games and peripherals inform us of the amount of exposition an average video game player might have to this kind of applications. Augmented reality (AR) consists of the superposition of computer-generated sensory feedback (audio, video, etc.) on top of a live view of the environment. Thus, reality is augmented by the virtual elements projected on top of it. Virtual reality (VR) in turn consists of a computer-generated environment where sensory experiences like sight, sound, and touch care created to simulate the user's presence inside this virtual world.

Only in recent years that technological advances have made it possible to produce mass-market AR applications. Though the video games industry has introduced many input and output AR devices through the years, only a few peripherals have become successful enough to push the medium forward. This section will review some of these technologies and applications.

First Person View Augmentations

AR applications are computer-mediated, presenting the augmentation of reality on a screen, smartphone, or specialized wearable computer system. In order to achieve immersion, they usually adopt a first person view of the world, through peripherals like head-mounted displays (HMDs). These have been tools of research for a long time but haven't met commercial success or widespread adoption as entertainment devices. Nintendo notably launched the famously disastrous Virtual Boy [1]in 1996, at the time marketed as the first portable video game console capable of outputting "true 3D graphics". Due to a series of reasons (not the least of them being the unpleasant experience of a low resolution monochromatic virtual world) the VirtualBoy didn't meet the success expected by Nintendo.

Jump forward to 2012 which saw the rise of the Rift[2], a crowdfunded effort launched by Oculus VR. The Rift intends to be an inexpensive video game-oriented HMD with motion tracking. Equipped with a single full-colour and high-definition display, it uses parallax to emulate depth. By outputting a wide angle 3D image it further increases the level of immersion thanks to peripheral vision. Still under development, the Rift has met wide

[1] http://en.wikipedia.org/wiki/Virtual_Boy
[2] http://www.oculusvr.com/rift/

success. Sony soon followed in the steps of Oculus VR by announcing their own Project Morpheus [3]in 2014, an HMD designed to work with their latest console, the PlayStation 4.

These are virtual reality solutions which can deliver augmented reality when coupled with video input. A more direct approach to AR is also possible with technology like Google's Glass [4](2013), a much less intrusive peripheral presented as eyeglasses and displaying visual information on a transparent screen in a small area.

In the absence of VR head-sets, first-person augmentation can be done with cheaper, less cumbersome and more versatile technology that users might already own. Notable first-person view applications which do not rely in HMDs include a number of games on The Nintendo 3DS handheld console launched in 2010. Equipped with two cameras on its backside and an autostereoscopic display capable of "glasses-free 3D", the device can mediate the user's view of the world while still retaining depth perception. Furthermore, being a handheld device, movement in space and the surroundings become integral gameplay elements. For instance, the Face Raiders[5] game (2010), which comes with the console, has the user fight off an invasion of flying heads by turning in place and shooting at them. The targets are 3D models projected on top of the stereoscopic view of the surroundings, effectively simulating their presence in space. The game Pokémon Dream Radar[6] (2012) uses a similar mechanism to project creatures the player must identify and hunt down.

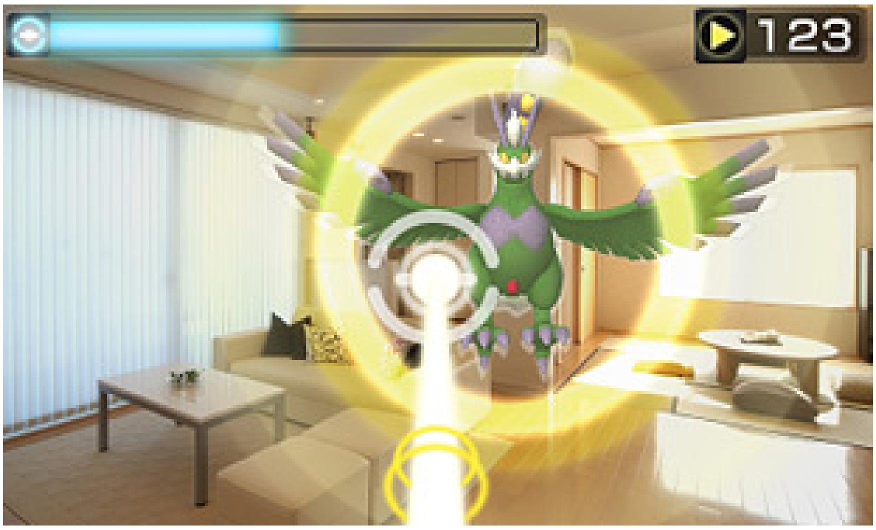


FIGURE 4: FIRST PERSON VIEW AUGMENTATION IN POKÉMON DREAM RADAR[7]

[3]http://www.polygon.com/2014/3/18/5524058/playstation-vr-ps4-virtual-reality
[4] https://www.google.com/glass/start/
[5] http://en.wikipedia.org/wiki/Face_Raiders
[6] http://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_Dream_Radar
[7] http://bulbapedia.bulbagarden.net/wiki/File:Pokemon_Dream_Radar_screenshot_1.jpg

Motion Control Games

The case of non-portable video game consoles is slightly different. Since these rely on televisions for display, they lack the mobility available on handhelds. Augmented Reality games for the home have thus taken a different approach, focusing mainly on full body tracking and motion control, resulting in either a reproduction of their actions on a screen avatar, or mirrored augmented video feedback. These can be related to the representation of a Reality-Virtuality continuum by Milgram et al (1994), and more specifically his concept of augmented virtuality, where reality is projected inside a virtual world (here, the virtual avatar mimics the movement of the real world person, or video of the person itself is projected inside the virtual world.)

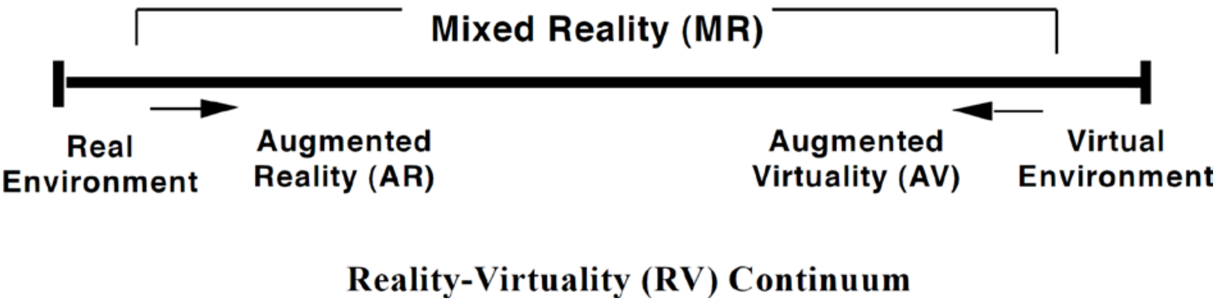


FIGURE 5: MILGRAM'S SIMPLIFIED REALITY-VIRTUALITY CONTINUUM.

In 2003, Sony launched the EyeToy[8], an inexpensive RGB camera peripheral intended to work in tandem with the PlayStation 2 console. A large number of games made use of the EyeToy, all playable by moving one's body to interact with virtual objects. Powered by different computer vision techniques, the EyeToy was succeeded in 2007 by the PlayStation Eye[9] for the PlayStation 3 console, which combines gesture recognition with a built-in microphone array to further increase its functionality. Games developed with these technology project virtual gameplay on top of the real world, accomplishing augmented reality.

In 2006 Nintendo launched the Wii, a revolutionary video game console with replaced the conventional gamepad by the Wii Remote[10], a peripheral capable of tracking motion in 3D space. Wii Sports, a game bundled with the console and consisting of five sports simulations, introduced motion control to the mass market and contributed to the consoles immense commercial success. Microsoft's Kinect [11](2010) for the Xbox 360 made away with the controller altogether by allowing games to be completely controlled by voice command and full-body motion. The device is equipped with an RGB camera, a microphone array, and a depth sensor. Games powered by these two peripherals augment the virtual world by

[8] http://en.wikipedia.org/wiki/EyeToy
[9] http://us.playstation.com/ps3/accessories/playstation-eye-camera-ps3.html
[10] http://en.wikipedia.org/wiki/Wii_Remote
[11] http://www.xbox.com/en-US/kinect

mapping the real motion of the player to that of his avatar, accomplishing augmented virtuality.

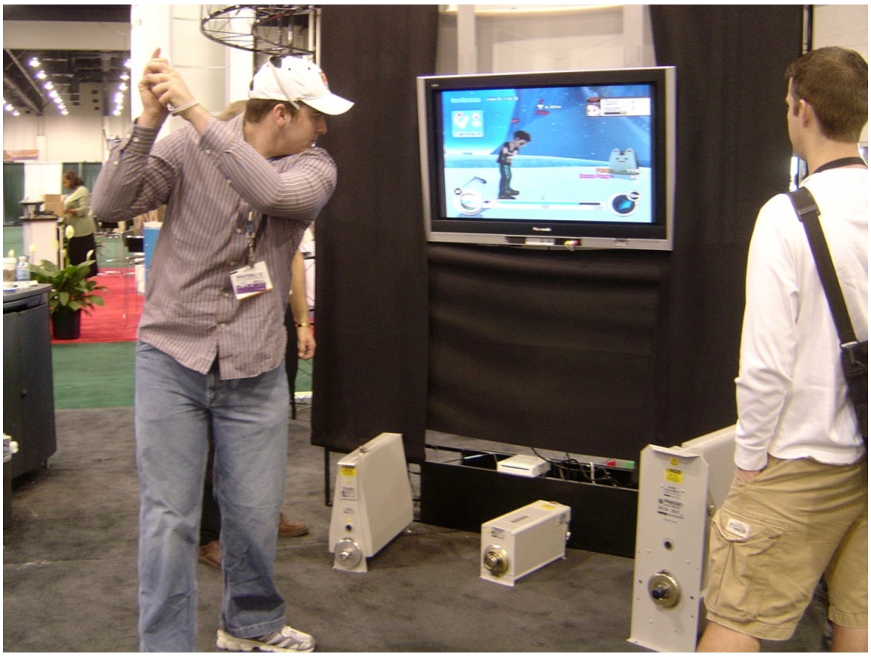


FIGURE 6: MAN PLAYING WII SPORTS, AN ILLUSTRATION OF AUGMENTED VIRTUALITY.[12]

All these technologies show an increasing interest in augmented reality and motion based interaction. Playing sports with the Wii Remote allows one to feel the tennis racket, golf club or baseball bat in the form of the controller, effectively resulting in a tangible interface, albeit while requiring specialized hardware, which comes with a cost and requires considerable learning on the part of the user.

2.4. PAPER-BASED AUGMENTED REALITY GAMES

Paper interfaces have not only proven to be an efficient and inexpensive solution for tangible interaction in research and pedagogy, but they've also been adopted by a number of commercial AR games. Unlike previous examples of augmented reality games, which use some combination of computer vision techniques, depth sensing and motion tracking, paper-based games usually make use of fiducial markers, due to their robustness under non optimal lighting conditions, but also due to ease of design. For instance, using markers is as easy as assigning a marker ID to a controller, rather than having to manually compute the features required to recognise a specific object within an image.

[12] https://flic.kr/p/HuNPD

Markers As Display

A very common usage of fiducial markers is that of a display of sorts. Placing a sheet of paper or card with a marker on it and filming it on the AR game will result in some visual object being projected on top of it. The Nintendo 3DS makes use of such paper cards [13]to project 3D models of famous video game characters on the real word, relaying the 3D visuals to the user. Paper cards don't offer any further interaction in this game, as all actions of the virtual model are controlled with the touch screen and buttons of the handheld device. Another similar application is the Pokédex 3D[14] (2011) also for the Nintendo 3DS, which serves as an encyclopaedia of sorts describing the over 700 different Pokémon creatures. Each Pokémon must be discovered first, and this is accomplished by scanning a unique marker, over which the 3D model of the creature will be projected. Once again, there is no further tangible interaction from that point.

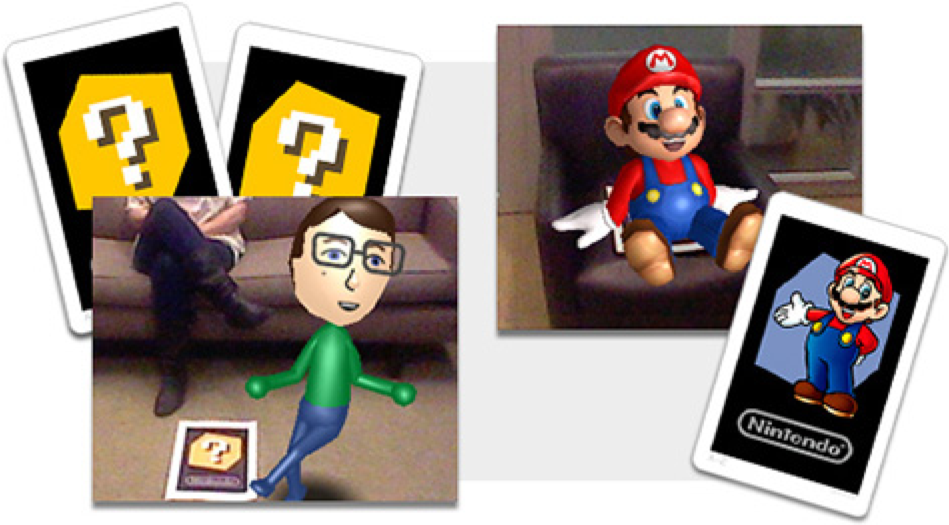


FIGURE 7: AR CARDS ON THE NINTENDO 3DS.[13]

A similar game is AR Defender [15]available on Apple's iPhone. The game uses a printed marker to place a virtual tower, which is under attack by different virtual enemies. The objective of the game is to defend the tower by shooting at the enemies, which is accomplished by moving the iPhone in space to aim, and tapping to shoot. Invizimals[16] (2009) on the PlayStation Portable (PSP) uses the same mechanism to project creatures on the real world and control them as they battle using the conventional buttons of the PSP. Once again in both games the markers serve simply as a reference point for the augmented game to be projected.

[13] http://www.nintendo.com/3ds/ar-cards

[14] http://www.pokedex3d.com/

[15] http://www.ardefender.com/

[16] http://invizimals.eu.playstation.com/

The case of a more tangible game using fiducial markers is that of the Wonderbook[17] (2012), a peripheral for the PlayStation 3 which works in conjunction with the PlayStation Eye and Move peripherals. The Wonderbook consists of a book-like foldable plastic object which, albeit not made of paper, is meant to simulate a physical book. The different games that use the peripheral augment the video image by projecting 3D objects on top of it and displaying the output as a mirrored augmented scene. The book serves little other purpose than that of a reference point for the action, with further tangible interaction being done with the help of the PlayStation Move wand.

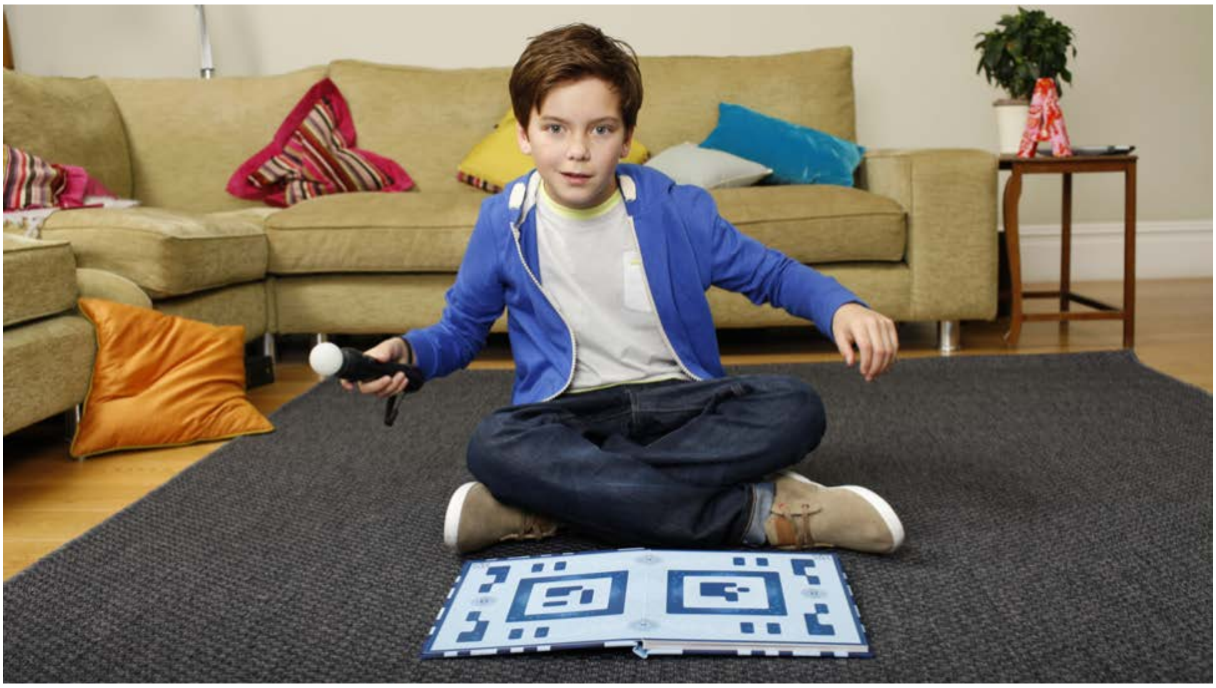


FIGURE 8: THE WONDERBOOK AND PS MOVE WAND.[17]

Markers As Tangible Controllers

Although most applications focus on augmenting through simple projection, some games make better use of paper elements to introduce tangible interaction. A number of commercial video games make use of paper cards for tangible gameplay mechanics.

The PlayStation Vita (2011) handheld console uses fiducial markers and its Wide-Area Augmented Reality[18] technology to track paper cards in space even outside the visible range of the vita camera. Sony's AR Play [19]series of games makes use of this technology in a similar manner to previous examples (by projecting virtual entities over markers), but by including multiple paper cards it allows for a flexible configuration of the playfield, effectively creating a physical level editor system. Cards can be placed at different distances to modify the size

[17] http://wonderbook.eu.playstation.com/
[18] http://youtu.be/wUxSwsy3PZo
[19] http://us.playstation.com/psvita/apps/psvita-app-ar.html

of the augmentation (i.e.: in sports games, the size of the field), and at different positions to modify the placement of different elements (in exploration or platformer games, change the location of trees, platforms, and tracks in 3D space.)

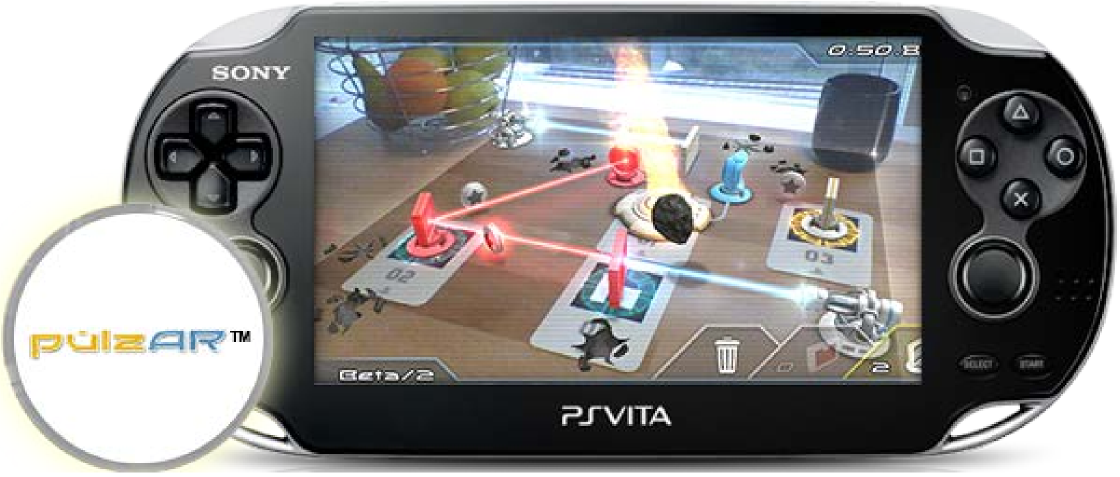


FIGURE 9: PS VITA AR PLAY[19]

In a more traditional use of paper cards, The Eye of Judgement[20] (2006) is a turn-based card battle game for the PlayStation 3 which uses the PlayStation Eye camera. Player actions are similar to those used in trading card games likes Magic: The Gathering or Yu-Gi-Oh! TCG, which each player having their own deck of cards, and playing cards from their hand to the field in predetermined positions. A table mat with a 3x3 grid serves as the playfield where the players place creature cards to summon them as well as other action cards. The Game identifies each card and tracks its position and orientation thanks to 2D markers printed on them and project 3D models of the creatures on top of the camera image.

[20] http://us.playstation.com/games/the-eye-of-judgment-ps3.html

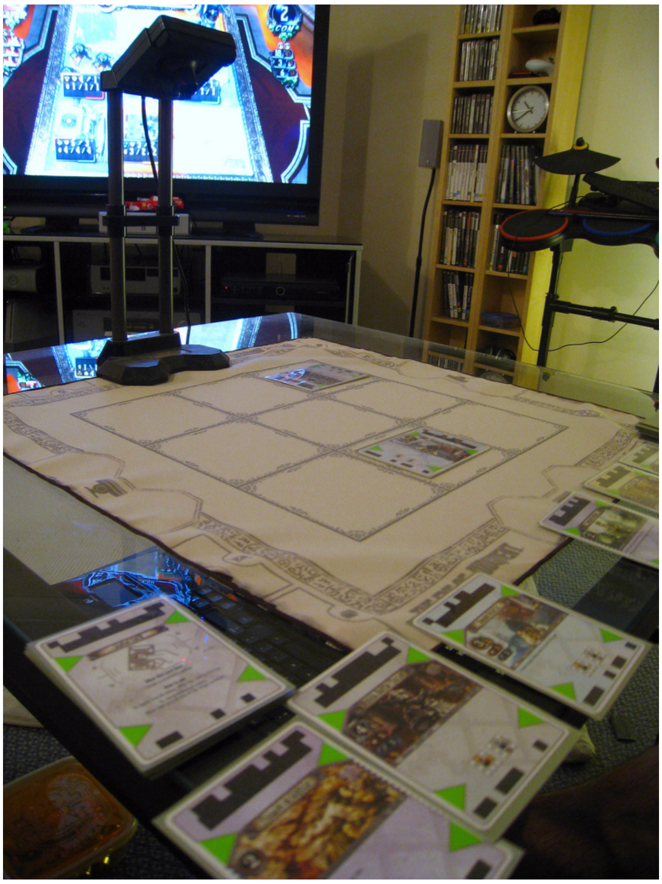


FIGURE 10: THE EYE OF JUDGEMENT PLAYFIELD, CARDS, AND CAMERA. THE GAME IS VISIBLE ON THE TV IN THE BACKGROUND.[21]

This quick review of existing commercial hardware and software already widely available in the market shows us that videogames are broadening their scope towards new interaction experiences. It remains to explore, from a development standpoint, how such interactions are implemented. As such, we now will look at a number of software tools currently available.

2.5. Software Tools Developing Paper-Based Games

Augmented and tangible gameplay mechanics have proved quite successful and will continue to appear as novel proprietary technologies in the future, ensuring the general public is exposed more and more often to tangible interaction. But developing such applications is still quite a challenge. Historically, TUIs and other AR applications have required highly specialized, expensive hardware, usually custom built with the specific purpose of serving a single application, with highly coupled hardware and software

[21] https://flic.kr/p/6nTCn1

components. Even if there are commercially available solutions including HMDs and pinch gloves, these remain expensive and hard to work with for the general enthusiast.

MARKERLESS SOLUTIONS

Tracking can be accomplished in the absence the aforementioned specialized hardware (for which a number of open drivers and frameworks exist). In the last decade, computers have become sufficiently powerful to run expensive algorithms as those required for markerless object tracking. In the last years, it has become possible to run such algorithms along with 3D applications not only natively, but also in the browser, using the technologies like JavaScript and a regular webcam.

A popular solution for developing such software is Open Source Computer Vision Library (OpenCV)[22], an open source computer vision and machine learning software library, capable of running over 2500 optimized algorithms to detect and track objects, faces, extract 3D models, recognize scenery, and more. Being a multi-platform, multi programming language library, it has been widely adopted by researchers, enthusiasts, and well/established names in the industry like Google, Intel, Microsoft, and more.

MARKER-BASED SOLUTIONS

While markerless detection is viable, it is far less reliable and doesn't intrinsically inform the user of what can be used for tangible interaction. By explicitly marking interactive elements with fiducial markers, it is both easier to detect these elements as it is to understand their expected behaviour: the user knows something will occur when the marker is seen by the application. As such fiducial markers appear as a most efficient approach to tangible interfaces and more specifically paper interfaces.

The most popular marker based solution is the Augmented Reality Tool Kit (ARToolKit)[23] which can precisely register and track the 3D position of paper elements through fiducial markers. It can work with any square marker patterns, meaning it can use a variety of square images for detection, albeit precision may vary depending on the choice of pattern. Being an open source solution, it has been adopted and adapted for a multitude of different platforms.

[22] http://opencv.org/about.html
[23] http://artoolkit.sourceforge.net/

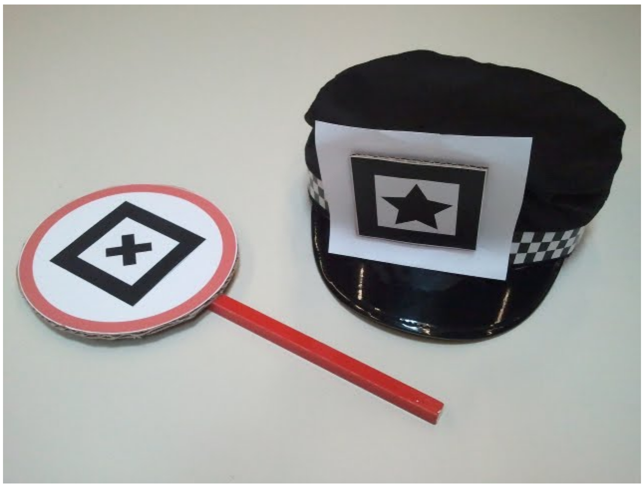


FIGURE 11: ARTOOLKIT COMPATIBLE MARKERS[24]

An alternative to ARToolKit is Chilitags[25]. Developed internally for projects of the CHILI lab, Chilitags offer a robust fiducial markers solution which has been used in a wide number of applications, from paper-based AR to robotics. This cross-platform software library is available both natively and as a JavaScript solution for web applications.

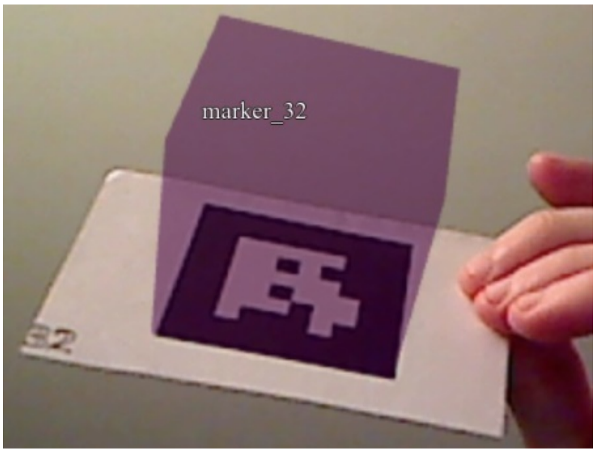


FIGURE 12: CHILITAGS IN USE.[26]

An interesting effort comes in the form of TUIO[27], a protocol for table-top tangible user interfaces. It consists of an open, common protocol and API specification for tangible multi touch screens, allowing for standardised transmission of touch events and tangible object states inside the interaction surface area. TUIO works thus as a general communication interface between tangible elements and underlying applications using them.

[24] http://www.mchablai.com/projects/stopcop

[25] https://github.com/chili-epfl/chilitags#readme

[26] http://quentin.bonnard.eu/portfolio/chilitags/

[27] http://www.tuio.org/?specification

Another more specialized solution comes from DART, a toolkit for rapid design exploration of augmented reality experiences. Built on top of Adobe Director, it uses the movie metaphor of the software as the basis to build applications. An ambitious AR framework, it offers a number of virtual components definitions including actor, events, physics and data storage. Through event based programming, it allows for fast prototyping and early testing.

TUIO and DART have something very important in common: they attempt to answer to the question of how to effectively define augmented and/or tangible experiences by defining a common language framework to start with. TUIO focus on interfacing hardware and low level functionality of controllers with tangible applications, while DART focus on rapid design, prototyping and exploration inside one specific editing environment. Both approaches thus offer new perspectives on the matter, while remaining tightly coupled to specific usages.

2.6. IDENTIFIED CHALLENGES

From our analysis of Tangible User Interfaces, we saw a number of conceptual frameworks have been provided by interaction researchers, while not demonstrating how to concretely develop them. Our review of AR games showed such concepts implemented in the form of highly specialized hard (Wii Remote, Kinect), or limited scope/single-purpose applications (TUIO, DART). Likewise, a number of toolkits for AR and tangible development exist and are well known in the developer community (ARToolKit). However, these are usually tied to specific detection technologies and offer basic models for tangible interaction.

PAPRIKA: A FRAMEWORK FOR PAPER-BASED INTERACTION

3.1. MOTIVATION

Although tracking solutions, in particular those based on fiducial markers, offer efficient methods to identify and track paper controllers, they suffer from a number of limitations. Most tracking technologies remain tied to the low-level information. However, this translation from low-level information to basic manipulations requires a considerable effort, and even more so in order to model more higher-level interactions which are natural and straightforward to understand like flipping, folding, and stacking paper elements, a step most often left to the application developer to implement. This is one of the reasons why it is difficult to create paper-based interfaces.

We have thus identified a gap between existing conceptual frameworks for TUIs and existing toolkits for the development of augmented paper applications. There is a need for an abstraction layer that bridges this gap both conceptually and communicatively, a common language for both researchers and enthusiast developers alike to define tangible paper-based games and applications; but also concretely, as practical solution to ease development in the absence of deep knowledge of computer vision and computer graphics.

3.2. GOALS & CHALLENGES

From a theoretical standpoint, the project aims at sketching an abstraction model for paper interactions. To this avail, we have taken into account both the aforementioned academic research and commercial products from Chapter 2, as well CHILI's research experience through the years seen in a number of augmented paper educative games and activities (as seen in Cuendet's and Bonnard's work in pedagogical applications for carpenter training and geometry teaching respectively). The resulting framework aims to answer to the needs of paper-based interaction, but also guide its design and development through meaningful definitions. Our proposed framework models high level manipulation (e.g. flip) through relatable definitions rather than specialized theoretical knowledge (e.g. computing Euler angles) in computer vision, computer graphics, and mathematics, which is usually required to implement augmented reality applications. The result is an extensible set of paper based interactions derived from common actions in card games, and modelled from basic properties of paper cards as further explained in section 3.3.

To demonstrate the feasibility of our framework, a concrete implementation serves as both a first evaluation and its multiple concepts, as well as a concrete practical solution to illustrate advantages and problems (which can eventually be tested in future evaluations) and lead to eventual evaluations with real subjects (researchers, developers). As discussed in chapter 2, existing solution rely on specialized knowledge in the form of multi-purpose computer vision toolkits, or are tightly coupled with a specific, not widely adopted platform. Our proposed implementation intends to be widely available and easily usable, so the choice of target platform was important. The main challenge in implementing this interaction framework was to ensure that our proposal represents an accessible tool for developers, requiring no advanced theoretical knowledge, and presenting itself as easily deployable and independent from the underlying technology as possible. The resulting implementation is presented in section 3.4. The final practical step was to put the framework to the test in the development of actual paper-based games, a number of which were produced during the project duration, along with demonstrations of the interaction functionality they made available. These example games are presented in section 3.5.

3.3. A Framework for Paper-based Interaction

From a theoretical standpoint, our framework serves as an interaction model for augmented card-based games. As such its main role is to model properties of paper cards and possible interactions at different abstraction levels. At the most basic level, the model considers the three basic properties of a paper card:

- **Presence**: is the card detected by the system?
- **Position**: where is the card located?
- **Rotation**: towards where is the card facing?

These properties define the state of a paper controller at a given time. A number of derived properties correspond to relative values (in time, between separate paper cards) of these properties. They can be combined and expanded upon to define more abstract states and interactions. For instance, by considering the presence of a paper card, we can describe it as appearing and conversely disappearing. Likewise, we can track the position and rotation of a card to see if it has been moved to a given location, or oriented in a specific direction.

Time plays an essential role in modelling interactions from the properties of paper controllers (the appearance of one vs. its absolute presence, its movement vs. its location). We considered two ways of representing this in the model definition. The first one consists of continuously binding (basic and derived) properties, effectively tracking the controller and its gradual changes in state and properties (continuous tracking of position, orientation).

The second representation is more interesting as it considers thresholds for the different properties, effectively defining abstract interactions as bound-crossing events related to the state of the controller cards. These states are defined by a condition on the (often continuous) basic properties mentioned above (e.g.: present/absent, facing towards/away from the view). The change between states corresponds to specific interactions (e.g.: appear/disappear, flip and which side is visible). The result is a number of binary triggers/events with two complementary states (has the card appeared? has it been flipped and to which side?) determined by the detected change and their previous state.

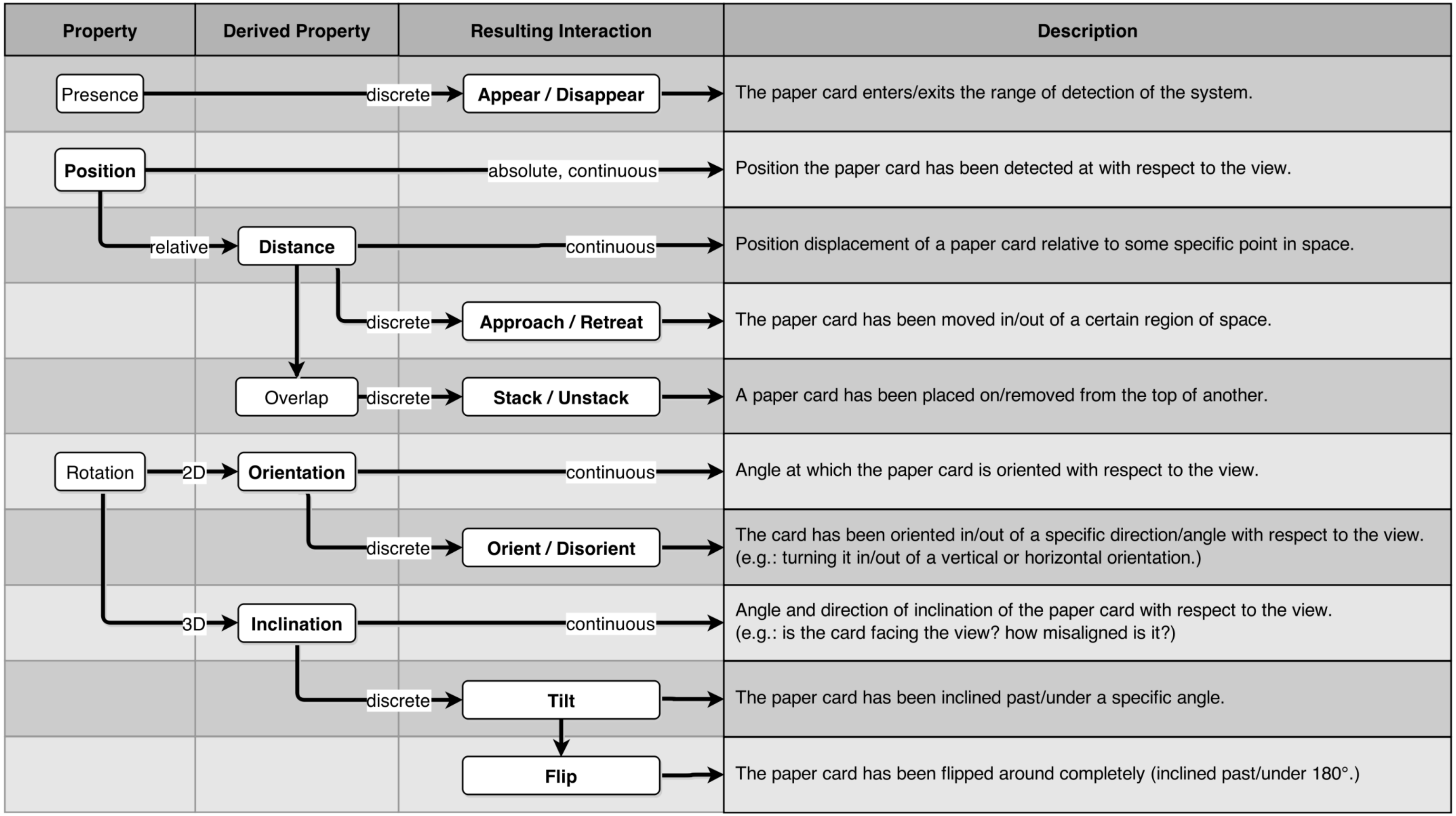
Our first set of paper element interactions includes appearing and disappearing, positioning, orienting, tilting, flipping, approaching, and stacking. These actions were derived from tradition card manipulation as seen on card games and board games using cards alike. More actions which could also be derived from these basic properties are shaking, waving, and derived properties like acceleration and velocity, as well as more destructive actions like folding, cutting/breaking, but these are seldom (if ever) used in such games (though they've proven quite popular in motion based video games in the past).

Figure 13 describes the different considered basic properties, their derived properties, the way these are continuously and discretely bound, as well as the resulting interactions. This set is not exhaustive but represent a first subgroup of interactions used in traditional card games.

Cards Definition

Since we consider paper cards as the basic block of interaction, the model assumes that the paper controller is generally a **flat rectangle**. The description of a card is composed of an identifier, its **height** and its **width**. The model doesn't restrict itself to a particular implementation solution, and could work with a number of alternative methods (like depth or RFID-based detection). It would most likely benefit for a combination of technologies, given than, for instance, RFID-based detection might not yield good results for detecting card orientation, whereas it might yield more reliable results for spatial tracking than fiducial markers, which in turn will rely on visibility of the markers (and thus are more prone to occlusions). Ultimately, the interaction between the proposed model and the implementation technology might inform both developers and designers on the specific requirements of their application, be it choosing a technology which yields good results for the desired interaction, or design this interaction around the limitations of the technology at hand.

FIGURE 13: DESCRIPTION OF PROPERTIES AND INTERACTIONS IN OUR FRAMEWORK.



Property	Derived Property	Resulting Interaction	Description
Presence		discrete **Appear / Disappear**	The paper card enters/exits the range of detection of the system.
Position		absolute, continuous	Position the paper card has been detected at with respect to the view.
relative	**Distance**	continuous	Position displacement of a paper card relative to some specific point in space.
		discrete **Approach / Retreat**	The paper card has been moved in/out of a certain region of space.
	Overlap	discrete **Stack / Unstack**	A paper card has been placed on/removed from the top of another.
Rotation	2D **Orientation**	continuous	Angle at which the paper card is oriented with respect to the view.
		discrete **Orient / Disorient**	The card has been oriented in/out of a specific direction/angle with respect to the view. (e.g.: turning it in/out of a vertical or horizontal orientation.)
	3D **Inclination**	continuous	Angle and direction of inclination of the paper card with respect to the view. (e.g.: is the card facing the view? how misaligned is it?)
		discrete **Tilt**	The paper card has been inclined past/under a specific angle.
		Flip	The paper card has been flipped around completely (inclined past/under 180°.)

3.4. Paprika: A JavaScript Implementation

For our implementation of the framework, we decided to develop a web solution. Today's computers are powerful enough for rich browser applications, so the web became the obvious choice given the high potential user base (who uses a computer but not the Internet?) With web development technologies like HTML5 one can build all sorts of rich and engaging applications and video games with relative ease, so a web solution would be capable of interfacing with already widely adopted creative tools.

This came in the form of Paprika, a JavaScript framework built on top of three.js, and Chilitags, from where it gets its name (paprika being produced from chili peppers). The source code of this implementation is available online at the address: http://chili-epfl.github.io/paprika/

Using Paprika

From the point of view of the application developer, Paprika appears as a global object with a number of functions. Once started, Paprika initializes its dependencies (low level operations) and starts an endless loop charged of interacting with the application (high abstraction interactions). The following figure illustrates the execution path of this loop. It starts by accessing the video feed and sending one frame at a time to Chilitags which in turn returns the estimated 3D transforms of all detected markers. These are then processed with three.js functions to extract the properties of the different objects and infer the higher level interactions present in our model. These will in turn trigger the behaviour defined by the application using Paprika through callback functions.

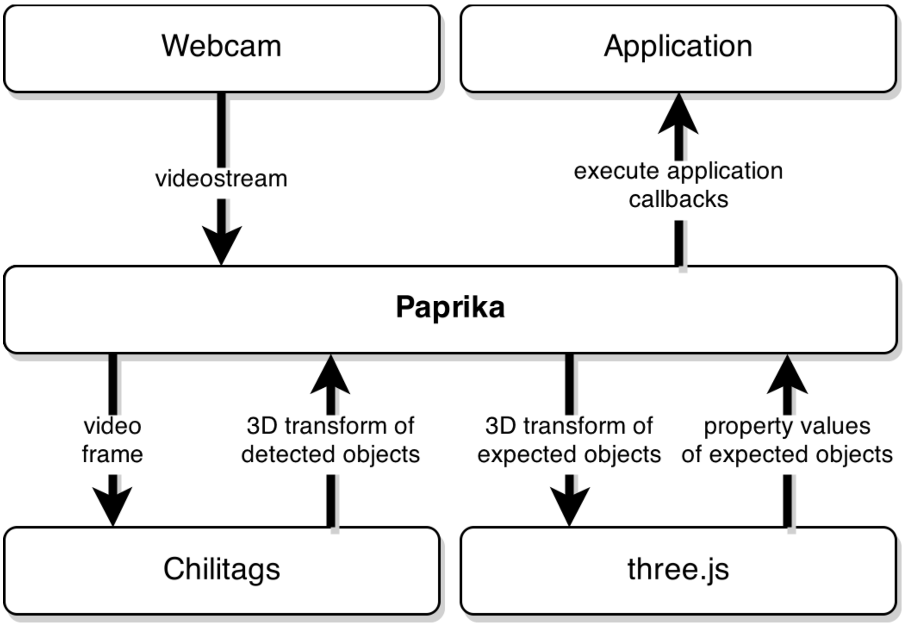

Webcam
Application
videostream
execute application callbacks
Paprika
video frame
3D transform of detected objects
3D transform of expected objects
property values of expected objects
Chilitags
three.js

FIGURE 14: PAPRIKA EXECUTION FLOW.

DEPENDENCIES

Paprika relies on two libraries. The first one handles tracking of paper elements, effectively leveraging the need of computer vision knowledge. The second carries the mathematical operations needed to model our proposed interactions, i.e. to leverage the required computer graphics related operations.

For the purpose identification and tracking or paper cards, it uses fiducial markers in the form of **Chilitags**[25]. The two main features of this library we were interested in are its reliable detection of tags on images, and its ability to estimate the pose of these tags in 3D space.

The second library used is **three.js**[28], a lightweight JavaScript solution for 3D graphics in the browser. Three.js offers a powerful set of objects and methods for 3D games and chances are developers interested in tangible 3D interfaces might already be familiar with it. Nonetheless, three.js remains invisible to the developer, it being accessing directly within Paprika and its usage being limited to geometrical operations, as it contains many useful functions for extracting information from the estimated pose of a marker.

As per in most computer graphic applications, the pose of a marker is encoded by Chilitags as a 4x4 matrix in homogeneous coordinates. From these, we can uses three.js functions to extract the position and rotation of the marker, and further derive our defined properties like orientation and inclination.

CAMERA-CENTRIC APPROACH AND 2D WORLD

In its current implementation, Paprika targets 2D applications. For simplicity's sake, we decided to adopt a camera-centric approach. The camera view serves as the interaction referential. As such, the framework does not discern between different physical layouts (playing in front of a laptop by holding cards in the air, versus playing on a table with the webcam tilted down over it being the most expected ones), which in turns makes it difficult to model physical distances meaningfully. All paper properties and interactions are accordingly defined with respect to the camera view.

Given this assumption, the framework's playfield is flat, parallel to the screen. While a number of manipulations take place in three-dimensional space (flip, tilt, stack) there is no concrete concept of depth in Paprika, and all visible elements are considered to be on the same 2D field. Positions are given as 2D view coordinates, orientation is defined within the 2D view plane, and inclination represents the misaligned of the card with respect to the view.

[28] http://threejs.org/

Card Definition

Chilitags automatically detects each marker as a separate physical controller, but paper cards can be more finely defined. In the current implementation of Paprika, cards are detected as punctual objects with no physical dimensions (as per the camera-centric approach.) Nonetheless, a paper card is considered as a flat, two-dimensional object, as per our proposed model. Each card is recognized by a number of markers printed on it. For each marker the following parameters are specified:

- The marker **identifier**,
- The **displacement** of the marker relative to the centre of the card (in x and y),
- Whether the marker is on the **front** or **back** side of the card.

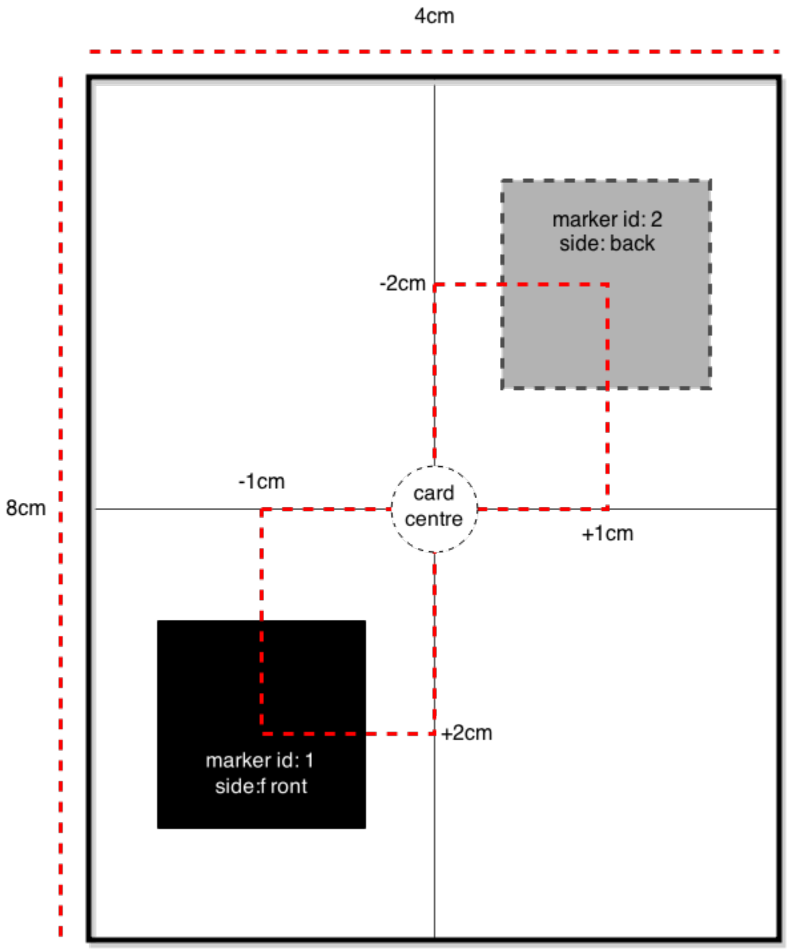
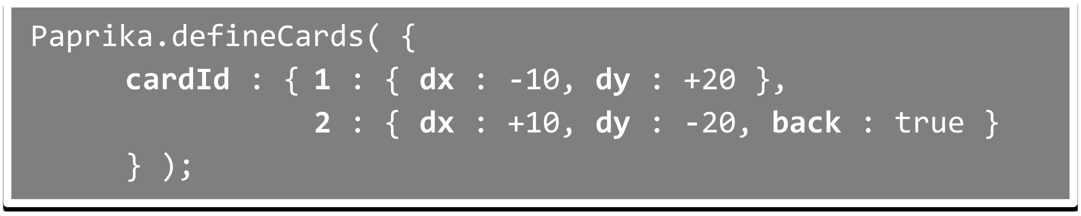


FIGURE 15: A POSSIBLE CARD CONFIGURATION MADE OF A MARKER ON EACH SIDE.

Card definitions must be provided to Paprika before launching the execution loop as, in the current implementation, these can only be passed to Chilitags at initialization.



```
Paprika.defineCards( {
      cardId : { 1 : { dx : -10, dy : +20 },
                 2 : { dx : +10, dy : -20, back : true }
      } );
```

FIGURE 16: THE DEFINITION OF THE CARD IN FIGURE 15. THE CARD WILL BE RECOGNIZED AS "CARDID". DISPLACEMENT FROM THE CENTER ("DX", "DY") IS MEASURED IN MILLIMETRES. THE "BACK" PARAMETER SPECIFIES WHEN A MARKER IS PLACED ON THE BACKSIDE OF THE CARD.

LIST OF FUNCTIONS

As per our model, Paprika can continuously track paper card properties. These can be bound to callback functions defined by the developer to handle the interaction logic of their application or game. More advanced interactions, i.e. discrete events can be detected. These are defined as triggers in an event-driven programming approach, where the developer defines callback functions to interface these events with their application.

The following list presents the different implemented functions currently available in Paprika and briefly explains their usage. Detailed examples of how to use each function are presented in the website of the project.

- **Paprika.defineCards**

Used to define the configuration of tags on paper cards, takes a map of card ids to card definitions. Each card definition is a map of marker ids to marker properties, as per the explained definition and sample code.

- **Paprika.start**

Used to start the execution of the framework. Here the user can decide if they wish to display video feedback and where to embed it inside the web page structure. The video stream is automatically requested by Paprika to the browser without the need of the developer to handle it in their application, Chilitags is started, and the main Paprika loop is launched, triggering the evaluation of the desired properties and interactions at each execution through a number of three.js functions.

PROPERTY BINDINGS

- **Paprika.bindPosition**

Sets a user-defined callback function to which send the computed 2D coordinates of the card on the camera view.

- **Paprika.bindDistance**

Sets a user-defined callback function to which send the 2D distance separating either a paper card and a 2D point, or two separate paper cards.

- **Paprika.bindOrientation**

Sets a user-defined callback function to which bind the angle at which the card is oriented with respect to the camera view. The angle of orientation is 0° for a card oriented normally towards the view (the interpretation of whether “normally” means up, down, or sideways is left to the application developer), and the value increases as it’s turned counter clockwise.

- **Paprika.bindTilt**

Sets a user-defined callback function to which send the inclination (its direction and magnitude) of a card with respect to the camera view. The angle of inclination (its tilt) is 0° for a card parallel to the view, 90° for a perpendicular one (undetectable,) and 180° for a perfectly flipped card. The direction of the inclination is also given, independently of the orientation of the card itself.

EVENT TRIGGERS

- **Paprika.onUpdate**

Sets a user-defined callback function to be executed at each iteration of the main loop. Here the developer can handle direct interaction with any detected marker/car through its transformation matrix.

- **Paprika.onAppear**

Sets a user-defined callback function to be executed whenever the paper card has been detected in view (**appears**). The callback is also triggered when the card **disappears**.

- **Paprika.onApproach**

Sets a user-defined callback function to be executed whenever a paper card has **entered** or **exited** a circle of a given radius around either a 2D point or a second paper card (when the distance between the two has become smaller/larger than a given threshold.)

- **Paprika.onOrient**

Sets a user-defined callback function to be executed whenever the paper card has been **oriented** within a certain range of a specific angle. The callback is also triggered when the card is oriented outside of this range (**disoriented**).

- **Paprika.onTilt**

Sets a user-defined callback function to be executed whenever the paper card has been **tilted** past a specific angle, in either direction. It requires markers to be placed on both sides of the card to work for any angle above 90°.

- **Paprika.onFlip**

Sets a user-defined callback function to be executed whenever the paper card has been **flipped** (tilted past 180° in either direction). It requires markers to be placed on both sides of the card to work.

- **Paprika.onStack**

Sets a user-defined callback to be executed whenever a pair of paper cards has been **stacked** (this requires visible proximity followed by disappearance of either card.) Also triggers when they've been **unstacked** (both cards appear visible again.)

Paprika is an ongoing open project, so function definitions are bound to evolve. The current implementation and its functionalities already offer a lot of possibilities to develop paper-based interactions as will be seen in the next section.

3.5. GAMES

In order to put Paprika to the test and to experiment with practical applications of the framework at an early stage, a few paper-based games were developed. The reader can try them at any time with no installation required, by visiting the project website at **http://chili-epfl.github.io/paprika/**

The games are playable with simple paper cards included as appendix with this document. A short video demonstration of the games can be seen at: **https://vimeo.com/98052994**

PHASER

The games were implemented with Phaser[29], a desktop and mobile HTML5 JavaScript framework for 2D games. Phaser is an open source and community driven project by Photon Storm[30], an independent game developer company. Phaser was chosen as it is a powerful, easy to use, free and open platform with an active community of developers.

[29] https://github.com/photonstorm/phaser#readme
[30] http://www.photonstorm.com/html5

Spatial Configuration of the Interaction

Three games were developed as Paprika took shape and helped test the functionality defined in our theoretical model. Some of the implementation decisions in Paprika, like the camera-centric approach, were motivated by the simpler needs of 2D game development, but also directly influenced the design process of these games. In the traditional desktop setup, the webcam is usually a separate peripheral from the screen, which is not the case when using a laptop, on which the webcam is fixed relative to the screen.

Assuming a desktop setup, one can expect the camera to be in a specific position relative to the playfield, looking down from it opposite the player (see figure 17). In this case, the game elements are projected above the physical ones, as if looked at from above, from the player's point of view. Interactions detected by the camera must thus be rotated by 180°.

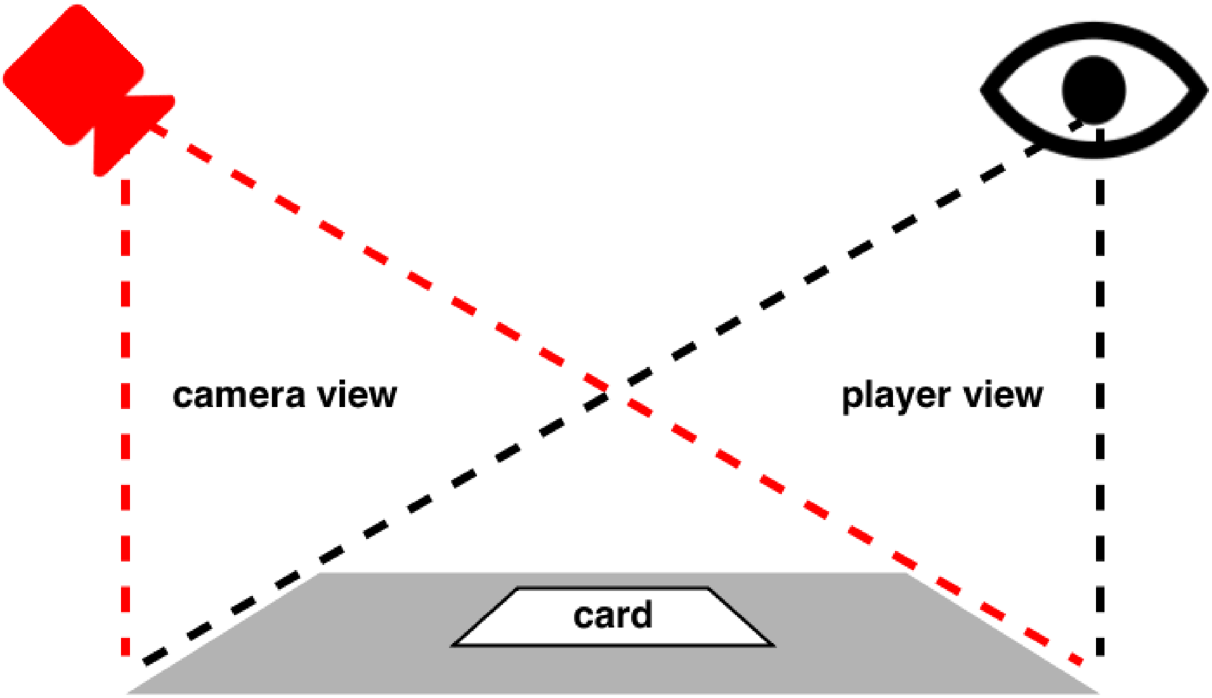


FIGURE 17: THE CAMERA AND THE PLAYER VIEW THE SAME PLAYFIELD FROM OPPOSITE DIRECTIONS.

The second case is usually what will be seen when using a laptop (see figure 18). The camera faces straight away from the screen and towards the user. In this setup, the camera sees the underside of the action, that is, the side of paper cards opposite the one seen by the user. In this setup, detected physical interactions must be considered as mirrored and virtual elements can be presented as either horizontal copies inside the screen, or mirror images, depending on the desired game representation.

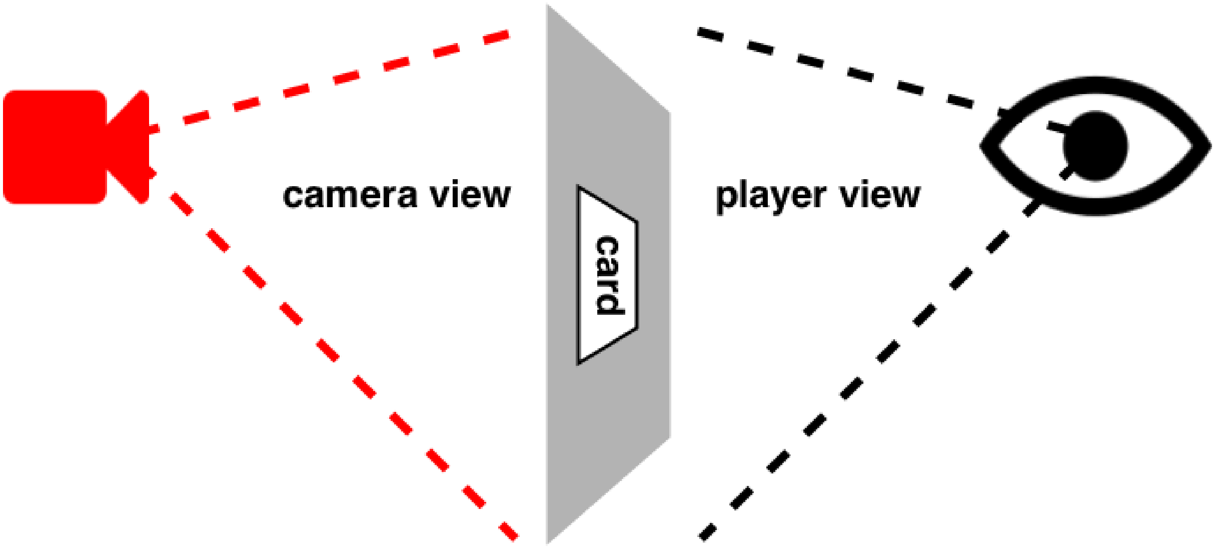


FIGURE 18: THE CAMERA AND PLAYER FACE EACH OTHER AND VIEW OPPOSITE SIDES OF THE CARDS.

Given Paprika's camera-centric implementation, this mechanic was part of the game code, i.e. from the point of view of a game developer using Paprika (as opposed to within Paprika itself), who must decide of the desired game setup and adapt the handling of the detected interactions accordingly. In order to inform the player of the required setup, two pictorial symbols were used on cards and on screen along the games to discern between the two playing conditions.

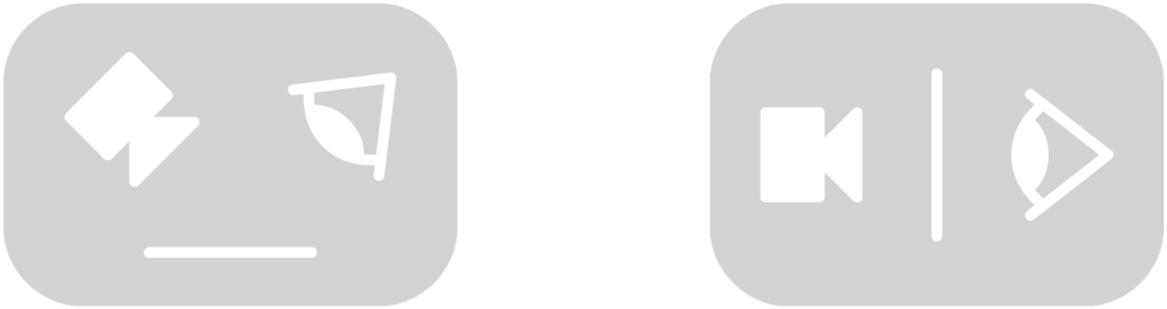


FIGURE 19: THE ICONS USED BY THE GAMES (BOTH ON-SCREEN AND ON THE CARDS) TO SPECIFY THE EXPECTED GAMEPLAY SETUP.

3.5.1. TRAFFIC MANAGEMENT

The game is playable at: **http://chili-epfl.github.io/paprika/phaser_demo_2.html**

The first game was thought of as a tabletop paper activity. It consists of a traffic management simulation, with the traffic light at multiple crossroads being controlled by different paper cards. By orienting each card in one of four directions, the player chooses which lane is active, allowing tracking to flow onto the next road segment.

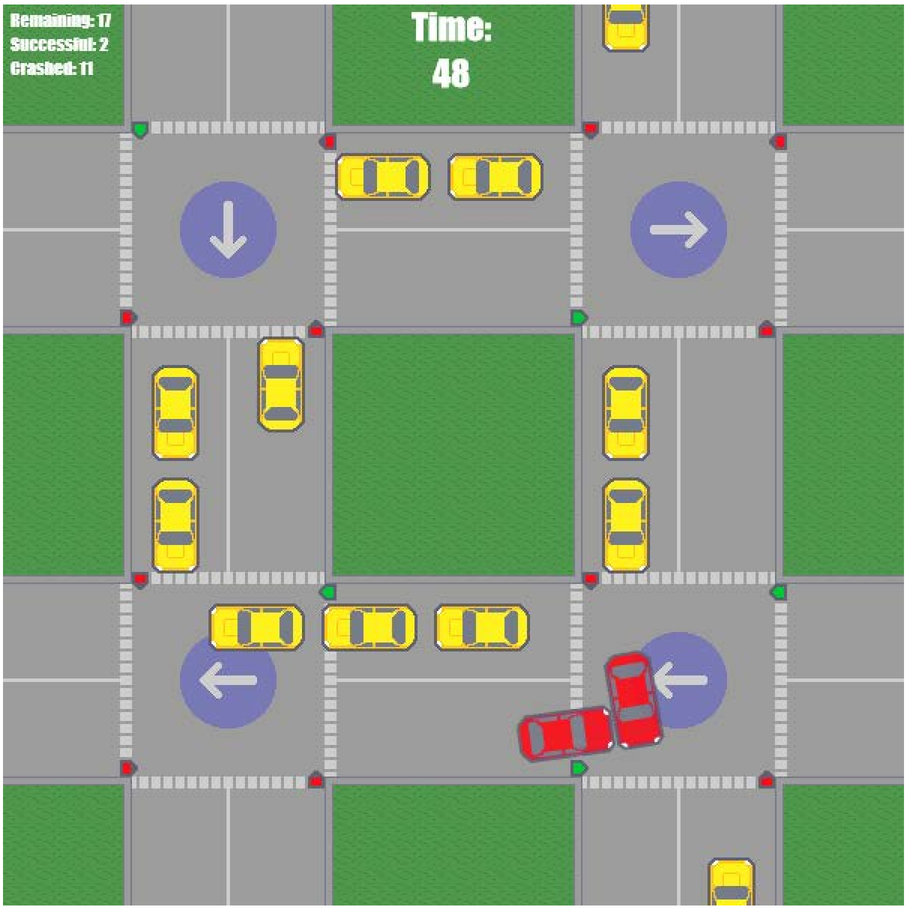


FIGURE 20: TRAFFIC MANANGEMENT GAME.

The game is played on a flat surface, with all cards visible, and player interaction consisting of rotating them. This allowed for multiple actions at the same time, as more than one card can be turned at a time, which wouldn't be the case in a mouse controlled configuration.

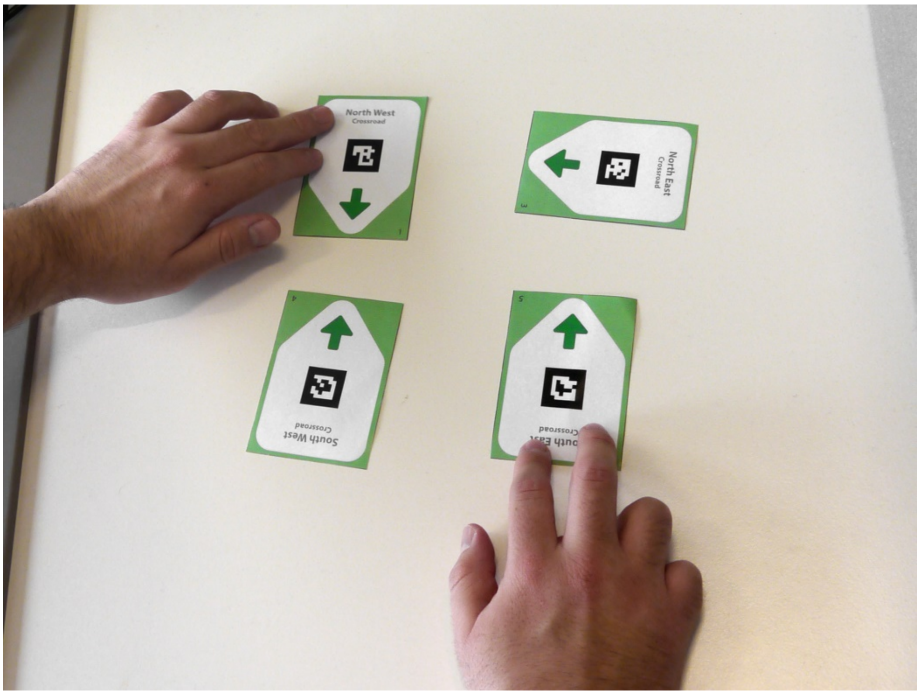


FIGURE 21: PLAYFIELD FROM THE POINT OF VIEW OF THE PLAYER.

As a first development step, a single crossroad collision demo was developed first, and can be played by holding a single card in front of the screen.

USAGE OF PAPRIKA

The game considers the discrete orientation of the cards, making use of the onOrient event definition (see list of event triggers definitions in section 3.4) to trigger the change of traffic flow in a crossroad (e.g., from top to down, from right to left, or vice versa). Like the rest of the games to be described below, this one uses a second card which, when shown, resets the game to its initial state. This interaction is accomplished with the use of the onAppear function.

ADVANTAGE OF PAPER + COMPUTER SETUP

Using paper cards allows multiplayer collaboration, as well as the possibility of controlling multiple cards simultaneously. Using a computer allows a graphical and dynamic simulation of traffic, and leverages keeping track of values like to time, and counting cards (both crashed and not.)

3.5.2. Asteroids

The game is playable at: **http://chili-epfl.github.io/paprika/phaser_demo_3.html**

The second game to be developed was inspired by the popular homonymous arcade game from 1979[31]. The game is a “space shooter” where the player controls the orientation of a spaceship and must destroy asteroids by shooting at them, before they hit and destroy the spaceship. The spaceship is controlled tangibly using a single, two-sided paper card with multiple markers, the orientation of which controls the direction the spaceship is facing. The ship shoots continuously.

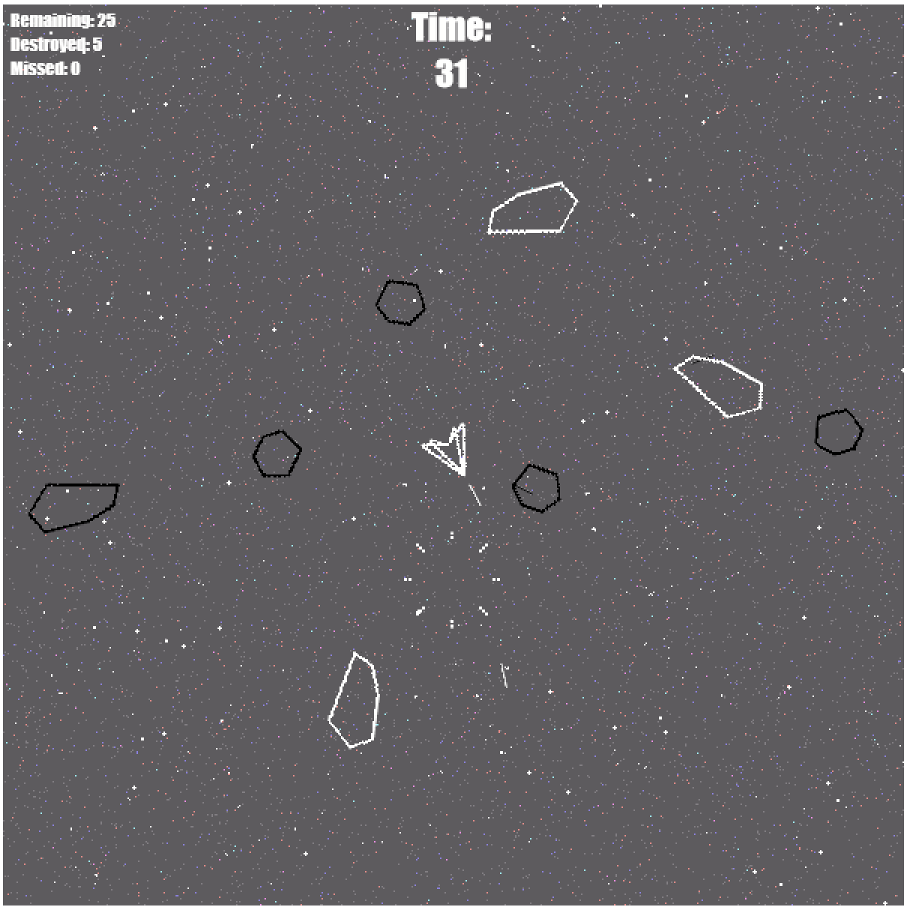


FIGURE 22: ASTEROIDS GAME.

A second mechanic is that of card flipping. Inspired by the polarity mechanic introduced by the “shoot ‘em up” video game Ikaruga[32] in 2001, the spaceship flips to change between black or white modes, making it immune to bullets of the same colour. In our Asteroids

[31] http://www.arcade-history.com/?n=asteroids-upright-model&page=detail&id=126

[32] http://en.wikipedia.org/wiki/Ikaruga

game, the spaceship can be changed between black and white modes by flipping the controller card, so that in each state, it can only destroy asteroids of the same colour.

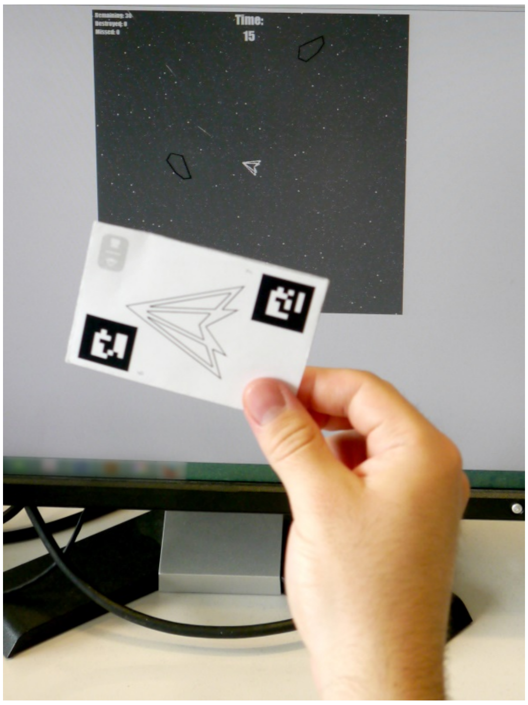


FIGURE 23: PLAYFIELD FROM THE POINT OF VIEW OF THE PLAYER.

Usage of Paprika

The game considers the continuous orientation of the paper card through the bindOrientation function. It uses the onFlip event definition to handle the mode switching mechanic. The spaceship card is defined with four markers, two on each side (for increased robustness to occlusion when holding the card). Placing at least a marker per side is a requirement for the onFlip event, as if no marker is present on the backside of the card, it would be considered absent rather than flipped. Two cards are defined in the same manner, the first to play by holding it in front of the screen (the preferred gameplay setup), and the second one by placing it on a surface in front of the screen.

Advantage of Paper + Computer Setup

Using paper cards allows for natural gestures like turning and flipping the card to be mapped to the same actions within the game. Using a computer once again allows a rich dynamic environment and automates score counting, letting the player focus on interaction.

3.5.3. BALL MAZE

The game is playable at: **http://chili-epfl.github.io/paprika/phaser_demo_4.html**

The third game implementing Paprika functionality is Ball Maze. Inspired by physical and ball-in-a-maze puzzles[33], the objective of the game is to guide a sphere from its starting position to a target virtual hole through a two-dimensional labyrinth. Traditionally built on wood, these are played by tilting the maze itself directly or through knobs. Our version of Ball Maze presents the user with a square labyrinth to cross in the shortest time possible, and also with as few collisions as possible with the labyrinth walls.

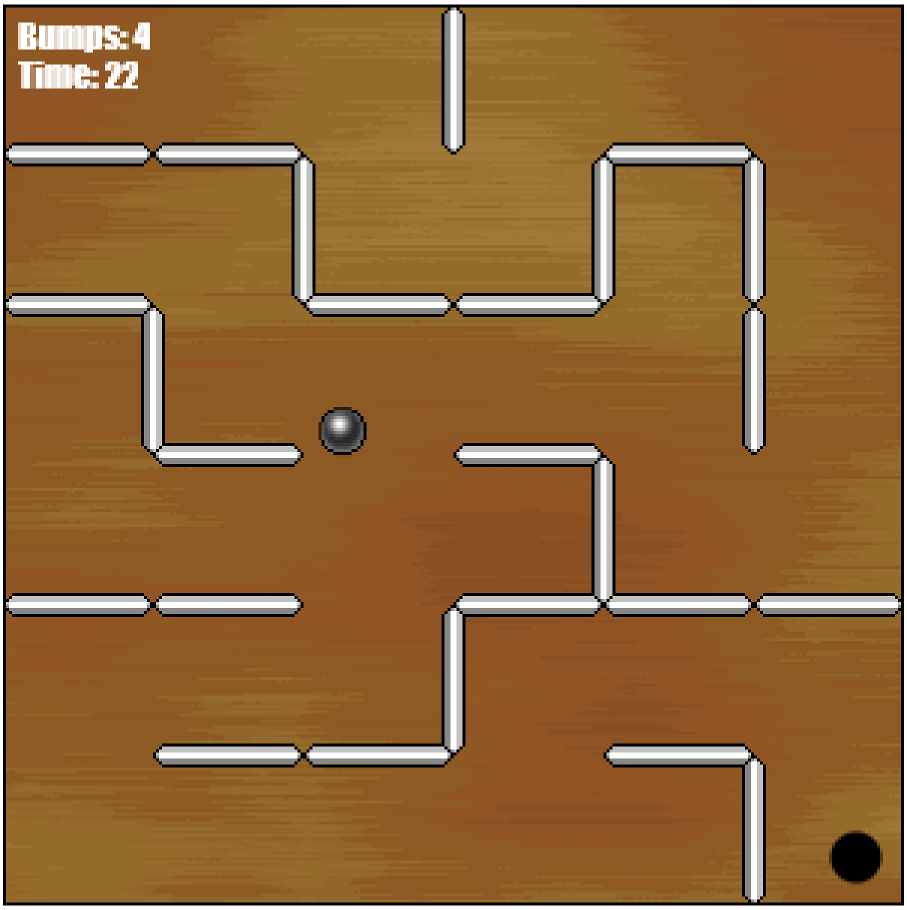


FIGURE 24: TRAFFIC MANANGEMENT GAME.

Ball Maze uses a single-sided multi-marker paper card as a controller which, when tilted, triggers the movement of the virtual sphere, mimicking the effect of tilting a physical maze to allow the ball to roll.

[33] http://en.wikipedia.org/wiki/Ball-in-a-maze_puzzle

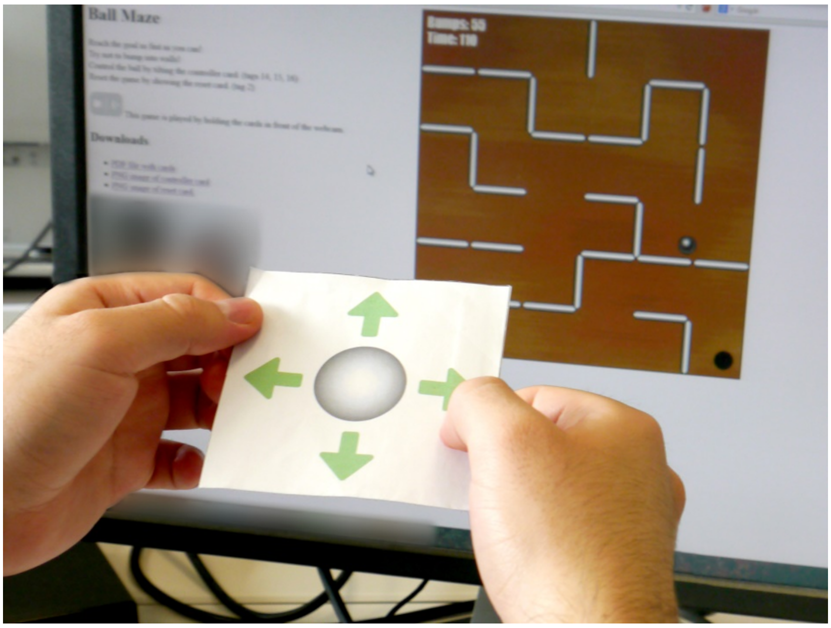


FIGURE 25: PLAYFIELD FROM THE POINT OF VIEW OF THE PLAYER.

USAGE OF PAPRIKA

The game considers the inclination (amount and direction) of the controller card, which it tracks continuously with the bindTilt function, updating the ball speed continuously.

ADVANTAGE OF PAPER + COMPUTER SETUP

Using paper once more allows natural gestures like inclining a physical surface to have a matching impact on its onscreen counterpart. Using a computer makes it simpler and more affordable to create multiple new levels, as opposed to a wooden ball maze where the setup cannot be changed, and where the ball could be lost.

DISCUSSION

In the previous chapter we theorized an interaction framework for paper-card based interaction. We proposed a first implementation in the shape of a framework for browser based applications and games, illustrating the concepts sketched in the abstract model. Finally, we put this framework to the test in the design and development of three browser games powered by a number of high abstraction interaction mechanics. Throughout these stages, a number of challenges appeared and were tackled.

4.1. OUTSTANDING CHALLENGES

From a theoretical standpoint, our framework covers a number of interaction scenarios. The implementation proposed in the form of Paprika addresses the difficulties faced by non-specialist developers and further helped us detect number of limitations and outstanding challenges.

UNTAPPED POTENTIAL FOR INTERACTION

The most obvious limitation of our current model is the unused properties of paper, namely it's malleability (interactions as folding, crushing, breaking) and its ability to both present and receive information to and from the user through ink (interactions and writing and reading, drawing, etc.) We argued the limited usability of such interactions in the case of paper-card games based on our knowledge of physical card-based game and table top games using cards, but the fact remains that these interactions are possible and unaddressed by our framework, at least it its current form.

We believe the current set of interactions to be rather sufficient for most simple scenarios, but it goes without saying that it's not exhaustive. This set of interactions can be reworked and expanded upon by considering other advanced gestures (e.g. pointing), though they can be approximated by a combination of existing interactions (e.g. positioning and orienting a card in a specific manner.) Ideally, we would prefer to have these ubiquitous gestures presented as well-defined atomic interaction.

Narrow Scope of Implementation Technologies

Paprika presents itself as a high level solution for game developers, while relying on low level libraries to handle computer vision and computer graphics operations. While it indeed does not require nor expect the developer to directly access this low level information, it remains limited by the capabilities of these technologies. By using fiducial markers, visibility is a key requirement for detection to take place. This limits interaction to non-occluded actions. This is not so much a problem for Paprika as it will be for game developers, forcing a number of design decisions to be made. In order to be aware of such limitations derived directly from the narrow scope of application of certain solutions, we believe that more empirical evaluations are required.

To truly gauge the power of the proposed abstraction model, we should work with seasoned game developers to both better understand the advantages and limitations of our implementation and to better evaluate the communicative power of the framework. Likewise, more implementations could be created relying on other technologies or combinations of technologies to create more effective and versatile solutions.

Absence of Referential Systems

In our implementation, we chose to implement a camera-centric approach: we consider the playfield of Paprika interactions to be the 2D view of the camera, with no regard for depth or real world dimensions. This forces application designers to address this limitation by coding their games with a physical setup in mind, as was the case of our games and the two possible configurations we presented in section 3.5. This implementation of Paprika limits the ability to design more elaborate games where the camera view does not match the playfield, which could be compensated by the inclusion of a referential system for defining play areas (as they might be more than one) where interaction is relative to their scale and location.

CONCLUSION

We opened by introducing the difficulties of designing and developing paper-based augmented applications. We established the context by reviewing commercial technologies and products, as well academic efforts towards the facilitation of implementing tangible interfaces. We proceeded to propose an abstraction model for high-level paper interaction, which we implemented in the shape of Paprika, a framework for web applications and games. Though suffering of a number of limitations, we believe Paprika is a good first step towards easing communication between interaction designers and application developers by defining a model for common paper-based interactions. Finally, we believe Paprika has the potential to evolve further and tackle on the challenges presented in the previous chapter.

Paprika is an open source project built with free and open technologies. It requires nothing more than a web browser and a webcam to power interactive applications. As such we believe it is worth pursuing development, as it has the potential to become a powerful tool, or at the least, allow for past prototyping and design of interactive applications. Other implementations of our theoretical framework are also worth considering, as other technologies might reveal as efficient or more than the ones found in Paprika. In the long run, such implementations could converge towards a simpler, easier to maintain codebase for augmented paper applications and games.

We have not bridged the gap between existing abstraction models and the limitations of low level development, but we believe we've taken a step in the right direction.

Bibliography

Bonnard, Q. (2012). Paper Interfaces: an HCI Approach to Geometry Education. Ph.D. Thesis, École Polytechnique Fédérale de Lausanne (EPFL). doi:10.5075/epfl-thesis-5579

Cuendet, S., Bonnard, Q., Kaplan, F., & Dillenbourg, P. (2011, May). Paper interface design for classroom orchestration. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems* (pp. 1993-1998). ACM.

Fitzmaurice, G. W., Ishii, H., & Buxton, W. A. (1995, May). Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 442-449). ACM Press/Addison-Wesley Publishing Co..

Heinrichs, F., Steimle, J., Schreiber, D., & Mühlhäuser, M. (2010, June). Letras: an architecture and framework for ubiquitous pen-and-paper interaction. In *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems* (pp. 193-198). ACM.

Kaltenbrunner, M., Bovermann, T., Bencina, R., & Costanza, E. (2005, May). TUIO: A protocol for table-top tangible user interfaces. In *Proc. of the 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation*.

MacIntyre, B., Gandy, M., Dow, S., & Bolter, J. D. (2004, October). DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (pp. 197-206). ACM.

Milgram, P., Takemura, H., Utsumi, A., & Kishino, F. (1994). Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. Proceedings of Telemanipulator and Telepresence Technologies. pp. 2351–34.

Perlman, R. (1976, May). Using computer technology to provide a creative learning environment for preschool children. Technical report, M.I.T.

Sellen, A. J., & Harper, R. H. R. The Myth of the Paperless Office. MIT Press, 2003.

Shaer, O., & Hornecker, E. (2010). Tangible user interfaces: past, present, and future directions. *Foundations and Trends in Human-Computer Interaction*, *3*(1–2), 1-137.

Ullmer, B., Ishii, H., & Jacob, R. J. (2005). Token+ constraint systems for tangible interaction with digital information. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *12*(1), 81-118.

Wellner, P. (1993). Interacting with paper on the DigitalDesk. *Communications of the ACM*, *36*(7), 87-96.