I Play, You Play: A Framework for Card Games in a Multi-touch Table

Nuno Maia
Instituto Superior Técnico, University of Lisbon
nunofmaia@tecnico.ulisboa.pt

Abstract. Card games have been around for thousand of years. There are countless variations of games and a large number of different families. Usually, people gather around a table to play card games, but with the appearance of video games the social aspects of the traditional card game have undergone a transformation. Researchers have been trying to blend the traditional and modern facets of card games by extending the traditional game with technology. The proposed framework provides a set of tools for the development of tabletop card games played using tangible playing cards. Moreover, autonomous agents can be integrated into the games and play them. The Sueca card game was built to test the capabilities of the framework. The hardware's limitations prevent a flawless detection and recognition of the cards, using fiducial markers. Nevertheless, the achieved results demonstrate the percentage of incorrect detections can be decreased and have a limited influence during the games.

Keywords: card games, fiducial markers, card games framework, multi-touch table, social robots

1 Introduction

Traditional games have gathered people around tables for centuries. They are a great source of entertainment and joy to everyone who plays them. Throughout the years, some of these games have been ported to the digital world. Recent advances on multi-touch surfaces and object recognition algorithms have allowed these games to keep some of the essence from the traditional version. Tabletop games have proven to enhance the game experience with visual and sound effects, making the game more dynamic [1].

Frameworks for game creation are not a new concept. There are already frameworks to create card games on touch surfaces [6] and projects that have agents interacting with humans while playing cards [3], but they do not explore the usage of physical objects. The few examples of card games using tangible interaction explored only the usage of Radio-frequency Identification (RFID) tags to extend the game and interact with it [2, 8]. Nonetheless, fiducial markers are being tested for building applications for different contexts such as music [4] and learning environments [9].
The PArCEIRO project\textsuperscript{1} focuses on the social interaction between humans and robots, aiming to study and create robots with the capabilities of a social companion. This interaction is sustained by the development of entertaining activities such as card games on tabletop displays. One possible scenario is having elderly people playing card games with a robot, and using real playing cards. The proposed framework pushes the project forward by providing a collection of tools to develop some of its activities.

The challenge of this work is to develop a set of tools that can be used to build tabletop card games. Furthermore, instead of having a digital representation, the framework promotes the usage physical playing cards to play these games. Simultaneously, it offers an easy integration of autonomous agents into the games. As a result, the framework made possible the development of different scenarios and activities for the PArCEIRO project, such as the Sueca card game, which is prepared to be played using physical playing cards and with an embodied agent that interacts with other players and is capable of playing the game.

\section{The Framework}

The framework should be able to represent the fundamental components for the creation of different card games for multi-touch tables. It should also enforce the usage of physical cards to play these games.

The game applications built with this framework follow the structure represented in Figure 1. For these applications, it is important to separate the physical parts, or layers, from the virtual ones.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig1.png}
\caption{Architecture of a game application}
\end{figure}

The physical layer includes all hardware that is capable of providing input to the games. This consists of touch tables, physical objects, and other peripherals that can capture information from the environment and translate it to bits of information that prove to be useful for the developed games, which may include cameras, microphones, or robotic figures. The touch display used to deploy these games was the Multi-Taction Cell 55” Full HD LCD Ultra Thin Bezel Display\textsuperscript{2}, which is able to detect an unlimited number of fingers and hands, and also specially crafted fiducial markers used in the playing cards (see Figure 2).

The virtual layer aggregates all the software and tools to build the games and deals with the events generated by the physical layer and its components. The framework was built on top of Unity3D\textsuperscript{3} due to the compatibility with other libraries already in use, and

\textsuperscript{1} http://gaips.inesc-id.pt/parceiro
\textsuperscript{2} http://www.multitaction.com/
\textsuperscript{3} http://unity3d.com/
also because it facilitates the design and development of the games.

2.1 Building blocks

Most card games have some characteristics in common. On the one hand, these can use the same type of cards, players arranged by teams, or even use tokens. On the other hand, these games can share other properties such as rules, e.g. follow the lead suit. The definition of such entities or directives, and the abstraction of some behaviors that can be reused, are the building blocks of this framework. The implementation of these concepts was divided into two separate libraries: Shuffle and Flow. The first defines the properties of common elements that can be found in many card games, while the second focuses only on managing and structuring behaviors.

As previously mentioned, all card games share some elements with each other. All card games are played using cards and it is not possible to play a game without players. Like players and cards, there may exist other properties that are required in order to represent and play a given card game. Therefore, a representation of these common elements, and also some specific ones, must exist to construct a certain card game.

Every concept is implemented as an interface. Examples of such concepts are the card game, card, and player, which are represented by the interfaces ICardGame, ICard, and IPlayer, respectively. Having these concepts defined as interfaces gives developers the ability of taking advantage of their composition to create more specific and complex primitives. With interfaces such as IScorable and ITeamable, any primitive can have a score property or a team. For instance, a ITeamPlayer, which shares the same properties of a IPlayer, but has information about the team they belong to by implementing the ITeamable interface. Since the support of physical objects is an important feature of this framework, the library also provides the IFiducialObject interface that exposes the properties needed to identify an object with fiducial markers.

In the same way Shuffle tries to break functionality and properties to their own interfaces, and promote the composition of those interfaces in order to build more specific and, perhaps, complex representations, breaking behavior down to their composing units can also be advantageous, in the sense that every unit can be composed to form a different behavior. Furthermore, having a system that simplifies the connection between these reusable units could ease the development of more complex behaviors. That is the purpose of the Flow library. Flow provides the tools for developers to visually create the desired behaviors. Moreover, it can be used for the validation of a certain play, but to control the state of the program and act accordingly. To
accomplish these goals, a graph structure is used to represent the behavior. This graph structure can be composed of three different types of nodes: the Action, Predicate, and Flow nodes (see Figure 3).

![Fig. 3. Flow nodes (action, predicate, and flow)](image)

The Action nodes execute a single action, changing the current state of the application and continuing to the next node. The Predicate nodes have a predicate attached, which returns true or false depending on the current state of the game or application. Unlike the Action nodes, that pass the execution to a single node, they are often connected to two nodes which are chosen depending on the truthfulness of the predicate. Finally, Flow nodes have another flow (graph) attached. More complex behaviors often require a large quantity of nodes and, as the number of nodes grows, laying them out becomes a difficult task. For that reason, the flow nodes offer a way to encapsulate another graph into a single node which can be connected to any other node (see Figure 4).

![Fig. 4. A graph represented as a flow node](image)

Every node has a Validate() method that is called to execute the desired behavior. In order for developers to add new behaviors, they have to extend the Action, Predicate, or Flow classes and implement this method.

These graphs can be created in a simple graph editor, providing a simple interface for interacting with the nodes and build them more easily.

### 2.2 Playing with physical cards

The detection and recognition of the markers are possible using the TUIO protocol [5]. This protocol is used for translating the received input, such as touch and marker events, to a more understandable and manageable format that can be processed to extract the desired information. Every time a fiducial marker is detected, an object representing it is created and added to a list, and removed when it is no longer in contact with the screen. By accessing the contents of that list, one can control which fiducial markers are currently on the touch surface. Many card games have different areas in which players are allowed to place their cards or tokens. Therefore, a single Unity3D component was developed to control in which areas the detection can happen, considering only markers contained inside its boundaries.

### 2.3 Connecting games and agents

Autonomous agents should be able to play the games with the human players. The Thalamus framework [7] provides
all the mechanisms to make a connection between the game and the agents. The game and the agents correspond to different modules that are able to exchange messages through the framework. However, the game itself cannot communicate directly with Thalamus; thus, some workarounds had to be made. The solution was to build a bridge that exchanges information between the game and Thalamus (see Figure 5). When the game wants to send information to Thalamus, it sends that information to the bridge which redirects it to Thalamus. For the game to receive information from other modules through Thalamus, the opposite process happens: the bridge subscribes to the events and sends the information back to the game application.

![Figure 5. Interaction between the game application and the Thalamus framework](image)

Every game is different from the others and therefore the messages defined for the interaction between the game and the agents may also be distinct. The number of events sent from the game to the agents defines the amount of interaction between the two. As the game publishes more information, the agent is more aware of what is happening during the game execution. This allows the agent to react and interact with the game and other players, making it more present.

3 Sueca, the Card Game

The Sueca card game was developed using the tools offered by this framework. Sueca belongs to the family of trick-taking card games. The game is played by two teams, with two players each. The deck is composed of 40 cards, which are distributed to the players, that start with 10 cards each, one for each trick (there are a total of 10 tricks in the game). At the end, the team that scores more than 60 points wins the game. It is relevant to mention players have to always follow the lead suit. Otherwise, they are taking the risk of renouncing, i.e. intentionally not following the lead suit.

3.1 State and rules

From the description previously given, it is possible to extract some of the concepts that are represented in the Shuffle library and that can help building the state of the Sueca game. Examples of these concepts are the trick-taking game, players and teams, and cards. Other concepts that are specific to Sueca are not present and therefore were added to the game representation.

Having the state of the game constructed, the next step was to determine how it could be handled in order to validate the plays and to be updated. The graph structure, made using Flow and illustrated by Figure 6 was made to validate plays and control the game state. The nodes that constitute this graph were made for the specific case of Sueca but some of them can also be used to build a graph for other card games. The NextTrick node is an example of a node that can be reused in other trick-taking games.
since its sole purpose is to save the current trick and prepare the new one. On the other hand, the \textit{IsPlayerCheating} is exclusive to this game, for dealing with renounces.

![Fig. 6. The flow graph for the Sueca card game](image)

### 3.2 Building the UI

The next natural step of the development process was to build the game from the state representation and rules. Building the game consisted in connecting the state and the behaviors with a simple User Interface (UI) that presents the players with relevant information and feedback of their actions.

A \textit{Sueca} card game requires some intermediate steps before actually starting, for instance shuffling, cutting, and dealing. These steps follow some conventions which the interface respects by indicating which player will perform the corresponding action. Another important phase of the pre-game is the selection of the trump suit.

The main view of the game displays a very simple arrangement of elements. The central area detects when cards are inside its boundaries, accepting only cards that were not yet played. When a card is detected, it triggers an event to process that card and start its validation. Additionally, to provide a more visual feedback to the players, a white glow appears beneath the card so that they know the card was well detected and recognized. The center area component is also responsible of triggering other events to control the UI and to make the game-play feel more natural, \textit{e.g.,} go to the next trick only when players remove the cards from the table.

### 3.3 Integrating an agent

Integrating an agent into a game requires some changes both in behavior and in the UI. As mentioned in Section 2.3, the communication between the game and the agents is made through a bridge that defines the actions to subscribe and publish.

As for the UI, the adjustments made considered only the existence of a single agent. Since an agent is not capable of holding their cards, some solutions needed to be encountered to overcome that limitation. There is a view where a human player is responsible for placing the agent’s cards on the display to be detected and recognized. Once the cards are recognized, there is no need to use the agent’s physical cards anymore because they are digitally represented (see Figure 7).

Finally, at the end of a game, the score view appears. Even though the score is presented to the players, they can easily count the points won by reviewing the won tricks. Nevertheless, the tricks will be a card short when human players are playing with an agent. Therefore, to provide a way for players to count the points on their own, the view shows which cards belonging to the agent were won by each team (see Figure 8).
Fig. 7. A digital card appears in the center when the agent plays

Fig. 8. The agent cards won by each team are displayed in the scoreboard

4 Evaluation

A framework is composed of several tools, each with its own purpose. For that reason, each tool can be evaluated individually or, alternatively, as a whole. This framework was already used to build a card game from scratch, which validates its purpose. Therefore, the only component that needs a more thorough evaluation is the detection and recognition of the physical playing cards and the overall experience of using them to play the games.

The idea of using physical playing cards to play card games on a multi-touch display is entirely new, which means there is no information available on how these cards should be in order to have a flawless detection and recognition. For that reason, some different designs were tested to achieve the best possible performance. The original intention for the cards was to change the design as little as possible comparing to standard cards, meaning that the card would be white, with the fiducial markers replacing the illustrations, and with the rank and suit information. Additionally, the size of the playing cards should be the same so that players feel comfortable playing with them. After trying this design on the touch display, some of its limitations are immediately revealed. The touch display uses a set of cameras to capture the display input and the quality of the captured black-and-white image is poor. Given the constraints on the markers’ size, this fact is not surprising, however the white of the card is so bright that prevents the marker to be detected, as the marker blends with the background. Consequently, the background color of the cards had to change and, after trying different color options such as red and different shades of gray, the best option was to make the cards black (see Figure 9). Whereas the white background had a bad impact on the marker’s detection, the dark background extends the markers’ borders and isolates them, making their detection much easier.

With the final design of the cards, two sets of tests were arranged. The first consisted in placing all the available cards on the touch display, one at a time, and see if the marker is correctly detected and also if the behavior changes during a certain interval of time. The time was measured in updates of the game loop: 10, 50, and 100 updates. For every interval, each card was thrown 5 times and the percent-
ages of correct and incorrect recognition were recorded. The results of this experiment can be seen in Figure 10.

The achieved values are too much alike to conclude that, after a certain interval, the chances of recognizing the correct marker code increases. However, the amount of incorrect values can be justified by the unsatisfactory quality of the captured image, which is not able to distinguish the markers with precision due to the similarities among them. In addition, in this test, the markers were compared with all 4096 available markers which increases the probability of getting an incorrect value.

The second set of test results gathered information of 100 games of Sueca with students. In order to declare that the usage of physical cards is viable, the majority of the played games must end successfully, meaning that all cards were correctly recognized. Results show that 96 of the total number of games ended without any problem, corresponding to a percentage of 96% of correct identifications. Comparing this result with the results of the test mentioned before, the improvement is noticeable. First, it is of great relevance to mention that the number of card detections is much higher than the previous, i.e. 4000 versus 600. Second, instead of comparing the markers with all possible values, they are only compared with markers attached to cards that are still left to play. By reducing the amount of possible values, the probability of an incorrect recognition decreases significantly.

Testing the system in the university domain is useful, but sometimes it is more important trying to test it with people that are accustomed to play in a day-to-day basis and that, perhaps, are not as tech-savvy as the students. Therefore, in an attempt to perform tests outside the university, the whole system was taken to an event featuring a Sueca tournament. The scenario was installed in a hall to invite anyone curious to play the game. The participants, after playing several games, were asked to fill a questionnaire to provide their insights about the experience. The questionnaire had two questions that focused more on the interaction with the touch display and the physical cards:

1. Did you enjoy playing Sueca on the multi-touch display?
2. Does the experience of playing Sueca on a multi-touch display have any issues?

The answers to these questions are highlighted by Figure 11. For the first question, the majority of the participants (approximately 65%) loved the experience of playing Sueca with physical cards and an agent on a multi-touch table, whereas 35% of the participants thought that it felt strange to play in this conditions. As for the second question, most of the participants thought that there were some problems with the interaction (almost 75%). Over 41% of the participants concluded the game does not flow in a natural fashion because of all the rules that are imposed in order to keep a correct game state. Moreover, more than 35% feels that the surface takes an excessive amount of time to detect the cards, which also contributes for a unnatural game play, and almost 12% of the participants has difficulties to identify the cards. Despite all the problems identified, close to 30% of the players thought the experience was good and that it did not have any issues.

5 Conclusions

The creation of a framework with the purpose of building card games for multi-touch table displays is an opportunity to develop a set of tools that encourages the development of new games and scenarios, but also to explore new approaches for the interaction with those games.

The development of this framework led to the creation of libraries and components that allow developers to build card games for multi-touch displays, using physical playing cards, and the possibility to play with an autonomous agent. This framework was used to make the Sueca card game, showing that it is possible to build card game scenarios using the developed tools. However, more scenarios have to be produced to confidently demonstrate that this framework is able to cover a larger number of card games.

The limitations of the hardware are evident, causing the recognition of the cards to be incorrect. Nevertheless, results show that, limiting the range of codes available to the game, the percentage of incorrect values decreases, reducing their impact during the game.
Bibliography


