Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Game Design Patterns in Digital Card Games

A browser engine to an abstract card game model

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplomingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Matthias Steinböck
Matrikelnummer 0527943

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Wien, 07.12.2017

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Game Design Patterns in Digital Card Games

A browser engine to an abstract card game model

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering & Internet Computing

by

Matthias Steinböck
Registration Number 0527943

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Vienna, 07.12.2017

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Matthias Steinböck
Am Modenapark 8-9/9/14, 1130 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasser)

Danksagung

An dieser Stelle möchte ich meinen aufrichtigen Dank an all jene aussprechen, die durch ihre fachliche und vor allem persönliche Unterstützung zum Gelingen dieser Diplomarbeit beigetragen haben.

Mein besonderer Dank gilt Herr Ao.Univ.Prof Dipl.-Ing. Dr.techn. Peter Purgathofer für das fortwährende Verständnis meiner außeruniversitären Tätigkeiten, die stetige Unterstützung bei richtungsweisenden Fragen und Hilfsbereitschaft im Sinne der Diplomarbeit.

Besonders bedanke ich mich auch bei meiner Familie die mich zu jeder Zeit unterstützt und dieses Studium ermöglicht hat.

Weiters danke ich meiner Schwester Christina Steinböck, für die tatkräftige Unterstützung beim Erstellen der Bilder der Pattern-Karten.

Weiters möchte ich mich bei Frieder Ulm für seine Unterstützung bei der Endkorrektur der Arbeit bedanken.

Ebenso danke ich meiner Lebensgefährtin, die mich stets mit Verständnis und Rückhalt in jedem Sinne durch den Abschluss meines Studiums begleitet hat.

Abstract

Digital card games are - as a subcategory of games in general - a subject of research in gamedesign. Ludologists use different approaches to describe, differentiate and classify games. One of those approaches are Design Patterns, commonalities that can be found analyzing different card games.

This work has two main goals and two main target audiences that are often combined in the same person. For gamedesigners I created a catalog of Design Patterns that can be used to analyze existing games as well as designing new games. For developers I designed an abstract model of card games that can be used as a data structure for new games.

Additionally this work contains:

1. an introduction to the world of (digital) card games (see 2)
2. classification-dimensions, which should allow better differentiation and classification of card games (see 2.4)
3. interaction patterns with real cards and an analysis of interaction-aesthetics (see 3)
4. a general abstract model for digital card games (see 4)
5. a catalog of design patterns for digital card games (see 5)
6. a description of the developed prototype (see 6)

Kurzfassung

Digitale Kartenspiele sind als Untergruppe der Spiele Gegenstand des Forschungsbereiches Gamedesign. Ludologen verwenden verschiedene Ansätze zum Beschreiben, Differenzieren und Klassifizieren von Spielen. Einer der Ansätze sind Design Patterns, also Gemeinsamkeiten welche sich bei der Analyse vieler verschiedener Spiele wiederfinden.

Die vorliegende Arbeit verfolgt zwei zentrale Themen. Zum einen wurde ein Katalog von Designmustern, mit welchen existierende Spiele analysiert und neue Spiele entwickelt werden können erstellt. Dies bietet sich insbesondere für Gamedesigner als systematisches Werkzeug für die kreative Spielentwicklung an. Zum anderen beinhaltet diese Arbeit ein generelles abstraktes Modell der Kartenspiele, welches als Datenstruktur neuer Spiele dienen kann. Dieses Modell eignet sich wiederum als Basis für die Arbeit von Softwareentwicklern. Eine prototypische Implementierung zeigt die Verwendung des Modells und die Auswirkung der Verwendung verschiedener Gamedesignmuster.

Des weiteres enthält die Arbeit:

1. eine Einführung in die Welt der (digitalen) Kartenspiele (siehe 2)
2. Klassifikationsdimensionen, welche es ermöglichen sollen, Kartenspiele besser differenzieren zu können (siehe 2.4)
3. Interaktionsmuster mit Karten und deren Interaktionsästhetik (siehe 3)
4. ein allgemeines abstraktes Modell für digitale Kartenspiele (siehe 4)
5. einen Katalog von Gamedesignmustern für digitale Kartenspiele (siehe 5)
6. eine Beschreibung des entwickelten Prototyps (siehe 6)

Contents

CHAPTER 1
Introduction

1.1 Motivation

In the late middle ages early card games began to spread across Europe. Since then a wide variety of card games have been developed all over the world [57]. Card games vary in complexity in multiple dimensions and it is not only luck and strategy that determine if a game is won. There are card games that are played in most simple ways, like with domino stones, or games with small amount of cards. More complex card games can include trading the cards and people hold tournaments to compete. People form clubs or associations for particular card games like Bingo, Bridge or Poker. One of the bigger being "The International Playing-Card Society".

The earliest digital card game I was able to find was Bingo. It was programmed by Larry Bethurum, released on 23rd January 1966 and was written in BASIC [8].

Most card games are known for a long time beginning with the game *Karnöffel* [43], which is first mentioned in 1426. It was designed to be heretical as the trump Knave wins every trick. Even the Pope or the King loses against the Knave. Because of its controversial design it quickly found many players all over Europe and even got forbidden in some places. Another big step in the history of card games was Whist [22]. After the rules for the simple trick-taking game were first published by Edmond Hoyle, it spread all over the world in multiple variants. It was so revolutionary and in demand that some people even stole the small pamphlet stating the rules.

The evolution of card games and their design however was slow and started flourishing with the advent of modern game design. It was 1991 when Richard Garfield designed *Mana Clash* - the first ancestor of collectible card games, which is also the year David Parlett published his book "A History of Card Games". As such, playing card games was and is a part of human society.

The techniques used to invent and create all those games and variants changed over time. Today we call the act of creating a game "designing a game". Concerning computer sciences, game design is an inherent part of game development. Every software designer creating a game has to make design decisions. Deciding what is the best way to solve a certain problem is often complicated because problems are cross-connected. Not only in technical terms, but also in questions of user experience, game aesthetics and user interface design. Therefore designers searched for a way to ease game creation and came up with design patterns. Patterns are systematic and abstract descriptions of best practice examples. They describe how something can be implemented.

The big questions that are tackled with this work are: Do card games have special design patterns compared to games in general? Is possible to find an abstract model describing card games in a general way? Can this model be derived from a catalog of design patterns? And if so, can this model be used to generalize digital card game development?

To test the results of my findings I will develop a prototype game that is based on the aforementioned model and can be played in web browsers. Designing and implementing a framework on which card based software can be built upon helps all developers who have to find a solution for this use case. Unfortunately, development processes and best practice examples for web developers nowadays change on a yearly basis.

1.2 Problem Statement

With the rise of the internet and the constant demand for mobile gaming and entertainment, many different card games have been digitalized to be playable using the web browser or alternative distributed technologies. Early implementations of card games involved complicated keyboard or mouse interaction and evolved with better input methods, getting more and more intuitive. Today, most online card games played by the growing amount of players, are implemented using Adobe Flash or Java Applets. With the ongoing development and implementation of the HTML5 standard in most modern web browsers, some games became available on all systems supporting those browsers, including mobile devices. This eases development a lot, as less environmental influences have to be considered.

Digitalizing card games can be done in many ways. Most implementations are based on 2D engines and are specialized implementations. They do not try to offer a reusable solution to the domain of digital card games in general. So most card software development frameworks for web browsers are cut to their needs, or are implemented rudimentarily. More generic approaches are rare and complicated to use.

The real problem software engineers and game designers face when it comes to digital card games is that there is no general model available to derive best practices for design, interaction and implementation.

1.3 Scientific Method

Game Design is a field of computer science that provides the theoretical background to the problem area of this work. As game designers use different approaches to generalize games, it is necessary to explain the method I used to describe digital card games. Inspired by Schell [66], who uses so called “lenses”, my idea was to create a catalog of specific card game design patterns from which a general model could possibly be derived. Designing the model I used ideas from the Mechanics-Dynamics-Aesthetics-Framework ([37], [77]). I learned that another problem area is the comparison between digital and non-digital card games, and that story telling is a significant part of card game design. Most card games do not differ in their basic interaction mechanics. They have different scoring schemes and winning strategies which are based on the games story or setting. Distinguishing storytelling and mechanics in game design is a problem pointed out by Jenkins and Murray [41, 53]. Finally, Crawford [17] inspired me to keep a sharp focus on this problem area whilst generalizing card games.

Today many game designers use game design patterns to describe commonly occurring problems and possible solutions. Although pattern catalogs exist, it is difficult to use them systematically, due to multiple reasons. I.e. different taxonomies are used. Patterns created by different scientists are based on different methods and sometimes are not comparable. These problems do not only exist in the digital world, but also in general game design. Most ludologists - as some game designers call themselves - have no common language that enables exchange or the creation of a common knowledge base. Björk and Holopainen [9] [10] also use design patterns, but keep those problems in mind. Their strategy is exploratory, though very systematic, and I used it to generate the List of Digital Card Game Design Patterns 5:

First I "brute force" analyzed existing card games (see List of Card Games A) by examining them one by one, as done by Björk and Holopainen [10]. In this process I extrapolated person-to-person or person-to-environment interactions done in five steps: recognize, analyze, describe, test and evaluate. As very little scientific literature about digital card games can be found, the findings are documented in Card Game Commons 2.

Secondly, the patterns identified during this process have been documented using five attributes: name, description, consequences, using the pattern, relations. While evaluating the patterns for digital card games, I built an index of verbs, actions and rules that are directly bound to the mechanics of digital card games. The verbs and actions are also be explained in the dimensions of usability aesthetics (pliability, fluency, etc.) as described by Löwgren [47] in Interaction 3.1. The patterns have been exported as a card game, and every card has been included in this work (see List of Digital Card Game Design Patterns 5).

Thirdly, I created the game *Uno* based on the developed model based on a library written in JavaScript for creating digital card games. The library is object oriented, easily extensible, and works on mobile devices, as well as desktop computers, using different input methods (touch and mouse). Some of the identified patterns have been applied using the described verbs/actions.

1.4 Structure

Following the introduction the second chapter "2. Card Game Commons" will describe the state of the art, some card game commonalities, like its classification, and define dimensions in which they can be located.

The third chapter "3. Interaction" describes player interaction (aesthetics) in card games and what considerations influence the attempt to create digital representations thereof.

The next chapter "4. Model" explains the model developed in course of my research in detail.

Design patterns derived whilst developing the model and analyzing multiple card games are described in detail in the fifth chapter "5. Card Game Design Patterns".

Concluding the last chapter “6. Implementation” will describe the web browser implementation and the technical challenges that had to be overcome.

CHAPTER 2

Card Game Commons

2.1 State of the Art

There are two relevant areas of research concerning this thesis: The implementation of digital games, and the game design-perspective. Following some examples using different approaches are listed.

Concerning implementation, card games are created using very different interfaces. One of the more extraordinary examples is a Multi-Touch Table for poker playing supported by mobile devices [67] developed by The Pervasive Computing Group, University of Duisburg-Essen. This example not only shows that digital card gaming is not limited to simple mouse and screen interaction, it also shows distant displays interaction as described by Dachselt [19]. Another similar example is an implementation based on Microsoft surface [2]. The application of game design patterns using pervasive strategies conduce a deeper feeling of immersion [48] and create a richer user experience. Tangible devices create an environment that feels more natural and real to the player. Touch-only interfaces, on the other hand, are not sufficient to create a similar experience.

Concerning design, besides general approaches to game design by Schell ("lenses" [66]), Crawford [17], Jenkins ("narrative architecture" [41]), and Hunicke (the Mechanics-Dynamics-Actions framework [37]), I found one interesting approach to the formalization of card game rules by Jose M. Font and Tobias Mahlmann [24]. This approach is exciting, because it formalizes card game play. It focuses on *Turn Based* 5.2 point-scoring games.

Buur and Soendergaard tried to improve design processes by playing a card game that connects cards to video playbacks [12]. Normally, cards contain text that is interpreted by the players. In a design process interpretation of text can cause misunderstanding. In their example, Buur and Soendergaard link a card to a video everybody can take a look at. Interpretation is different because more information is available for a conclusion. They found out, Videos are remembered and interpreted more easily than written ideas. In this context, cards are used as a tangible object connecting the idea represented by the video with the card. Therefore, this example is not a card game, strictly speaking. The problem of what is a card game and what is not is discussed in the following sections. This example shows how complex the domain of interaction with cards can get.

The Department of Computer Science and Engineering of The Chinese University of Hong Kong implemented an Augmented Reality Table. In their work they combine the strengths of the virtual and real world [45] to create an enriched playing experience and describe how this combination can be applied to different use cases e.g. in a design process. Players play an existing trading card game on a virtual surface that does automatic counting and enhances the game experience with visuals and audio playback. The game experience has a multi medial character.

Römer and Domnitcheva of the Department of Computer Science, ETH Zurich developed a "Smart Playing Cards" application, by attaching RFID tags to real cards. The game automatically counts the scores and even has a cheating alarm. [63]

In 2013 two scientists from Waseda University, Japan implemented this idea using the popular trading card game *Yu-Gi-Oh* [65].

What should be shown by those examples is that game designers already have many ways to implement their design goals or design patterns. However, most user interfaces nowadays are two dimensional and limit our visualization abilities to screens and beamers. Perhaps this will change when holographic projection and interaction becomes more pliable. This problem - as shown in the examples - is circumvented by using objects that are registered by the computer and resemble digital objects called tangible objects [40].

My goal is to create a 2D-based prototype for the web browser. Following, some state of the art alternatives for implementing the prototype game are listed.

The web - currently (2017) a fast developing area of computer science - has brought up many implementations of JavaScript 2D-graphics engines [29, 36]. Some of them are based on SVG or WebGL as described by Williams in [78]. At the time of writing this work, WebGL is not widely supported by modern web browsers (Internet Explorer, Safari, Opera and Firefox do only support version 1 [13] whereas Chrome and Firefox already support version 2 [31]). It is based on OpenGL and is maintained by the non-profit Khronos Group [49], not by the W3C. The canvas-element is supported more widely. It was separated from the HTML5-standard as own "module" in 2010 [15] but still is listed as "W3C Working Draft". SVG [76] is supported by all major browsers, and is vector based, not pixel based like canvas. Another alternative would be DHTML and CSS3 transformations as shown by a fun to play implementation of the child-friendly card game "Go Fish" [30]. However, support for WebGL improves and may become the best choice in the future. The JavaScript based gaming engine phaser [59] for example has a renderer for canvas and for WebGL.

2.2 Classification

To reach our goal, it is important to understand the structure of card games. It could be possible to derive a model from this structure. Besides numerous sub genres of card games (most of them named after the key mechanic), the classification of card games is difficult as well and no common scientific classification exists that I was able to find.

The most commonly used classification is based on the cards used. Not only abstract aspects (like rank and amount of suits), but also aesthetic properties are used for the classification, like the painting used to depict the rank. Thus card games can be classified by families of resembling card decks normally named after their believed origin:

- *French-suited*
- *German and Swiss-suited*
- *Latin-suited*
- *Abstract*

David Parlett [57] suggests a classification by game mechanics subdivided by game objectives. He distinguishes:

Mechanics

1. *Null:* games in which cards themselves are not the instruments of play
2. *Exchange:* exchange cards between players or with the deck
3. *Matching outplay:* play cards until a preferred layout is achieved
4. *Competitive outplay (trick-taking):* specialized sequence, a player wins the trick

Objectives

1. *Null:* gambling games
2. *Penalty Avoidance:* avoid holding a penalty card
3. *Card elimination:* be the first to play out all cards
4. *Card combination:* form sets of matching cards
5. *Card capture:* trick playing

This approach however does not cover modern abstract card games where most of the objectives and mechanics are mixed, making a classification again very difficult. Trying to classify modern games by this system, I realized that most of them fall into multiple (or all) non-null categories. Because of this fact, I tried using the concept of ludemes [56], David Parlett identifies three types in card games:

- *mechanical:* physical interaction
- *purposive:* aim or objective
- *decorative:* nonessential cultural motifs

More generally speaking and trying a classification by purposive elements, card games fall into three types [51]:

- *Fun games*: Casual games that do not require a deep thought, ie. games for children or drinking games.

- *Veteran games*: Games that involve a lot of counting, knowing tricks, bidding and complex strategies.
- *Unorthodox/Abstract card games*: Games which are not trivial nor complex enough to be appealing to card game fans.

As there are many approaches to card game classification, it is difficult to use the classification to create an abstract model. So I tried to look at the instruments of play in card games - the cards themselves - and realized that this approach is again futile, as they are also used in other contexts:

- *In board games*: Some board games (eg. The Settlers of Catan) use cards as a supplement to the gameplay on the board.
- *For learning*: Cards are a useful tool for associative learning. It is possible to combine images and words for vocabulary learning. Another example are colors to group animals for learning animal classification. Game based learning is a connected field of research. It was discovered by Richard Van Eck that card games are useful for learning "the ability to match concepts, manipulate numbers, and recognize patterns" [74].
- *Knowledge references*: Reference cards are most commonly cards with lots of text. A complex topic is described as short as possible but as extensive as adequate. Reference cards are a compressed categorized representation of knowledge.
- *User Experience Design*: A method called "Card Sorting" can be used to organize the information structure of a website [54]
- *Project management processes*: Cards are also excessively used in project management processes like Scrum or Kanban. They represent a task or a bug that has to move on the board over time.

Understanding the basic structure of card games, I returned to games played in real life from which I proceeded to the virtual world by continuing using this knowledge. Digital card games can be described as a subgroup of *software using cards as interaction element*. As such, a more general approach for implementing digital card games should be considered. My approach was to only concentrate on the mechanical *ludemes* 2.3.

2.3 Taxonomy

Before I can outline the model itself, I need to declare some common taxonomy. The need for a common language for games and game design in general, as described by Björk, Lundgren and Holopainen [10], is also applicable to the domain of card games. Many synonyms exist, used by players on game conventions, or in reviews. Some games have special names for basic objects in order to enhance player immersion. This complicates communication between game designers. As there exists no common agreement on the ontology of card games known to me, my approach is shown in Figure 2.1.

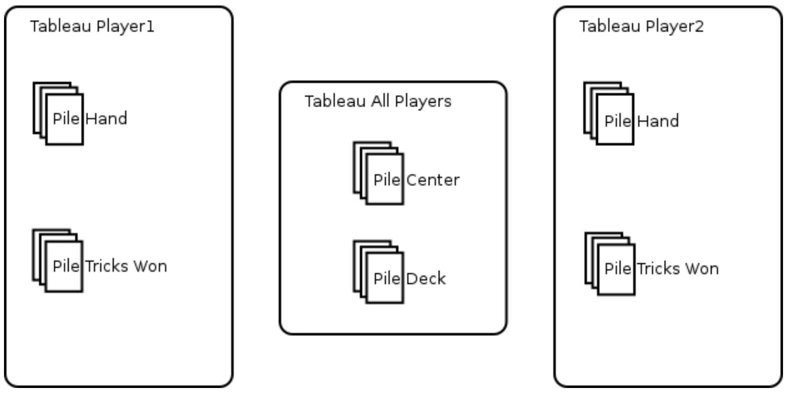


Figure 2.1: The basic virtual objects in card games are the tableau, the pile and the cards

One could be confused by the fact that the elements described in figure 2.1 do not contain a player, a deck or a hand. In our taxonomy, the hand and the deck is the same as a pile, and every player can interact with every object.

Ludemes

Besides these three basic elements, the essence of a card game is the description of how players can interact with those objects and how they can be manipulated. In this context, *manipulating* means adding or removing objects, changing their relative positions, changing their attributes. While analyzing multiple card games (A), I extended the basic taxonomy by the following elements and interactions that can be commonly found with some superficial description:

- **Attribute:** a named expression on any object holding a literal or dynamic value
- **Card:** a card is a collection of attributes that vary depending on the game, like name, a picture, card-type, suit, rank or creature-strength
- **Pile:** a collection of cards. Attributes:
  - name
  - order strategy: sorted, mixed, ...
  - visibility policy: first card visible, all cards hidden, ...
  - privacy policy: private, shared between all players, ...
  - drawing strategy: 1 card per turn, event based, ...
  - card layout: stacked, aufgefächert, ...
- **Tableau:** a collection of piles. Attributes: name, which player the tableau belongs to (or none or all), layout of piles

- **Move:** moving an object from one place to another, normally referred to as *drawing* or *playing*. Attributes: secrecy mode
- **Action:** the act of modifying the game state directly without the usage of a card
- **Rule:** a condition that allows or prohibits a move or an action

More details about the difference between *Move* 3.12 and *Action* can be found in *Move versus Action* 4.2.

2.4 Dimensions of Classification

I found that some games use cards as an additional instrument of gameplay like *Settlers of Catan*, *Activity* or *Nobody is Perfect* as described in *Classification* 2.2. It turned out that I need to draw a line between card-assisted games and card games per se. Furthermore, I needed a basis to compare existing digital card games. Based on my research, I defined the following dimensions to quantify those differences. Those dimensions help to categorize card games because the approach is based on the players experience, which correlates to how we perceive card games. I use those dimensions to gain a better understanding what kind of card game we are talking about. They should not get mixed up with *Interaction Design Dimensions* described in 3.2.

- ***dimensions specific for card games***
  - intensity of card usage as instrument of gameplay
  - freedom of interactions vs. restrictions by gameplay and environment
  - card attribute and deck complexity
  - amount of different interactions a player can choose from (draw, discard, take a trick, ...)
- ***more general dimensions but still strongly connected to card games***
  - importance of luck as winning strategy
  - rule complexity and granularity

Following I will describe every dimension in more detail.

Intensity of card usage as an Instrument of Gameplay

To play a classic card game such as *Solitaire* or *Skat*, players require the cards and a flat area - most commonly a table - sometimes covered with a card suitable green fabric. Additionally, some card games make use of tokens, scoreboards, dice, etc. that are also required.

The significance of using cards in a game can be quantified answering the following questions:

- Can the cards themselves be replaced by any other means?

- Do players mainly use cards or other objects?
- If there are other objects used, could those be replaced by other means?

Answering those questions, the *Dimension of Card-Usage-Intensity* can be quantified relatively. An absolute metric could be difficult to find without further empirical comparative studies.

Freedom of Interaction

The *Dimension of Freedom of Interaction* is influenced by device specific attributes and interface design. I will describe how this dimension is restricted by explaining how a card game can be digitized assuming that total freedom can only be found in real life. The restrictions described below are cumulative.

Playing games in real life means absolute freedom of interaction because theoretically every player can do everything. The player is primarily only restricted by code of conduct because even rules can be ignored or changed. Playing card games with the help of a computer restricts interaction and defines this dimension.

Tangible objects or other means of interacting with the digital system do restrict freedom of interaction indirectly as they link the real and the virtual world.

More restrictive to the Dimension of Interaction is the feedback provided by the digital system connected to those objects. The system can only react in a meaningful way if it fully understands the interaction correctly. Since it can only recognize interactions it is programmed to identify and the selection of recognizable behaviors determine the feedback, freedom of interaction is restricted by:

- accuracy of measurement: object position, ...
- foreknown interaction types
- latency of expected feedback

Digitizing card games using touch interfaces could be seen as the next step. In such environments interaction is limited to visual feedback and touching. It is very important to weigh automatism (animations) against direct control of virtual objects. A touch interface can enable a pliable user experience.

Playing games on a desktop computer with a mouse and keyboard is the ultimate physical restriction considering card games. The interaction is not only limited by the interface but also because the feedback loop is interrupted by a general purpose instrument like a mouse or keyboard.

The virtual realm (interfaces) of this dimension can be grouped into three regions. Interfaces that:

- allow every manipulation and solely provide information
- disallow certain interactions

- allow only a set of given interactions

The *Dimension of Freedom of Interaction* is difficult to measure in exact quantities because it depends on multiple aspects of game interaction:

- the device and interface used
- the code of conduct agreed to
- the rules of the game

However one can observe tendencies, compare and discuss them to order the games accordingly.

Object Attribute Complexity

Most classic card games use so called *standard decks*. Standard decks are normalized subsets of the full set of standard cards, like the *Standard Deck* 5.8. There are a lot of different subsets using different painting or design, but their attributes stay the same. Those attributes are only two:

- rank - from ace to king including numbers from 2 to 10
- suit - normally there are four of them

Modern card games, however, do not only add more ranks or suits. They extend the list of attributes. One of the more complex games included in my research, *Magic: The Gathering*, even defines card types before defining their attributes. Depending on the type (super- and subtypes also exist), there are a lot of different attributes.

To the end of clarity and simplicity children's card games have less attributes. E.g. in the game *Domino*, every card only has two numbers on it. *Memory* has only one painting on every card. But there exist a lot of children's card games (like *Go Fish*) played with a standard 52 cards deck, where the meaning of suit and rank are used in a simplified way.

The amount of distinguishable attributes is countable and the dimension of *Object Attribute Complexity* can be built on an absolute metric.

Amount of Possible Interactions

In most card games, the player is required to hide cards from the others. Every time the player is allowed or required to, the set of cards in the hand are exchanged according to the rules of the game. The cards from the hand are moved to piles other players own or can also interact with.

I compiled a list of possible interactions with cards and other common game objects in card games in the next chapter 3.

The fact that most card games include all of those physical interactions complicate ordering games on this dimension. That is why the *Dimension of possible interactions* is mainly influenced by the rules constraining or disallowing some interactions in some situations.

So calculating the Interactions-Amount-Value for a game has to be an average value: Sum all possible interactions in every state a player can reach and divide it by the number of different states.

Luck as a Winning Strategy

Most card games have randomness, and therefore some amount of luck is required to win a game. Shuffling the cards before starting a game already adds the first element of uncertainty. Shuffling is done to balance chances between players.

Some games do use cards, but the main interaction is not done by playing and exchanging the cards. Gambling card games like *Poker* fall into this category.

Games like *Chess* or *Go* are perfect information games. It is theoretically possible to calculate all possible moves, to evaluate them, and to choose the best. Those games have no element of luck at all. *Set* is an example for a card game having almost no element of luck.

It is difficult to measure the influence of luck in a card game. But it is certainly possible to compare and order the games on a *Dimension of Luck-Impact on Winning-Odds*.

Rule Complexity and Granularity

As Funes points out in [28], complexity should be an intermediate property between perfect uniformity and total randomness. Given the vast amount of card games played all over the planet, almost every degree of complexity of rule systems exist.

While *Go Fish* and *Old Maid* are some of the simpler games, *Android Netrunner* or *Magic: The Gathering* belong to the complex ones.

It is again difficult to calculate a complexity value for a card game. Simply counting the base rules, the exceptions and irregularities is a good place to start. From that measurement, one can indirectly see how complex or simple a game can get given the right circumstances. The problem measuring the complexity of card games lies within finding a symbolic encoding for their rules. The model I developed might allow to calculate a game's *Algorithmic Information Content* (also known as Kolmogorov Complexity) and use that value for comparison.

A more general approach to measure complexity of card games, similar to Lloyd's idea [46], is to try to find answers to the following questions:

- How hard is it to describe the game using rules and exceptions?
- How hard is it to play the game?
- What is its degree of organization?

CHAPTER 3

Interaction

3.1 Introduction

Interaction in card games has a distinct social aspect. A good example many people know and play is *Uno*. Comparing how the game is played by different groups shows that there are sometimes minor, sometimes major differences to the basic rule set, and thus interaction is perceived differently. For each group of people, the emotional focus, the main goal, the relation and behavior between opponents tends to differ.

Regarding our interaction elements, the *Move* is simple in comparison to the *Action*, which had to be explored in more detail. I built an index of verbs, actions and interaction elements from the games researched that are directly bond to the mechanics of digital card games. Towards achieving my goal of an abstract model for digital systems commonly available, I restricted my research to two dimensional screen-interaction. From the dimensions of usability aesthetics, as described by Löwgren [47], I can learn how those interactions should be implemented in a digital environment.

In this chapter, I will look into the physical and abstract interactions with card games and their game state. For each interaction, I extracted a sentence that summarizes what I learned from my research. This helped me a lot to identify patterns in the digital games analyzed later.

3.2 Dimensions of Interaction

The idea of using dimensions in interaction as a basis for interaction design communication came up in the late 2010s. Gillian Crampton Smith stated that there are four dimensions to an interaction design language [52]. Shortly after, Kevin Silver added a fifth dimension to the original idea [69]:

- 1D: words - which are interactions
- 2D: visual representation - graphics with which users interact

- 3D: physical objects or space - with which or within which users interact
- 4D: time - within which users interact
- 5D: behavior - including action, or operation, and presentation, or reaction

The fact that the second dimension is multidimensional itself does not change anything.

3.3 Handling rules

Breaching a rule in a game is of course a no go. However, this is also a source of innovation and new games can be created. I found that family-rules are an important part of card gaming. Of course there is a booklet outlining the rules for the original game. A card game that can not be modified can be perceived as “not interesting”. This is not the case for classic games like Chess or Tetris. Concerning this aspect, card games are different.

If a digital card game recognizes a breach of rules, the game should certainly inform the players about it. Automatic *score counting should stop* as a result because the game state is undefined. Now it could be possible to undo the action or *to continue* the game. It is true that this does not make any sense in a tournament situation. But it makes a difference if technology restrains the gaming experience or the *behavioral contract* between players decides how to deal with the situation.

Learning: Evaluate how strict rule breaches are handled and if it is the system or the players who decide.

3.4 Appearance and Interaction with a Card

When displaying a card, its properties must be clearly recognizable. A card should never be up-sampled automatically unless the player can recognize all properties when required to know them. If the screen is too small to recognize the cards, one could consider using a popover, or a window, to display the cards in original game size. However, adding an additional interaction element to enable the user to recognize a card should be avoided.

For a pliable experience the player should be able to *rotate* the card and *toss* it around.

Concerning children-friendly games, it could be allowed to *steal* or *hide* cards to open the experience to cheating and confrontation.

Learning: Evaluate how “real” a card should feel.

3.5 Appearance and Interaction with a Pile

For a fluent experience, game objects should resemble relevant properties from real life. Taking a sneak peek on a real pile informs the player not only about if there are cards left, it is also possible to get a feeling about *how many cards* are left. This information should have a digital resemblance, such as a bigger shadow, or cards sticking out from below the top most card.

Another suggestion would be *not to enforce card rotation*. If the cards stack up the same rotation as they are moved to the pile, the pile feels like a journal to the game. It is fluent that players have to interact with a badly stacked pile because it resembles reality. Such behavior could be configurable.

If a card is moved to a pile, it is also possible to show an animation how the card is aligned to the pile. That way the player does not feel interrupted and does not need to come back to the pile and clean it up.

Learning: Decide how "real" a pile should look like.

3.6 Shuffle

Shuffling is the act of randomizing the deck before a game starts. Shuffling normally takes time and in some games there exist rules about how to shuffle and who has to shuffle but most games just require a randomized stack of cards. It heavily depends on the interface used to create the digital card game to decide how this interaction should be implemented.

The games rhythm is not broken by a shuffling animation. In fact, randomizing the cards is an important part of the dramaturgical structure of a card game. It might even lead to mistrust between players. Therefore, the whole process of shuffling, cutting, and dealing should be visualized.

Learning: Evaluate how much time should be used for a shuffling animation.

3.7 Cut

Cutting a pile is normally done to decide which player starts, or which position the player is playing in. Also, it is done to further randomize the cards. However there is a big difference between automatic cutting and player controlled cutting. If players can cut a pile, they should have some means of defining *where to cut* it.

Learning: Choose if decisions are simply announced, or a cut-animation is used.

3.8 Deal

Because dealing is an interaction that is often more disputed than the game's rules, it should be *done by the computer*. Most players are not very good at dealing, and there is a reason to why tons of youtube videos exist on this topic.

Rhythm is disturbed if a bad dealing interface slows the process. If the dealer has to deal manually, the dealer feels more aroused with every card dealt while the players impatiently wait for the dealer to finish. If not done purposely, the dealing process should be handled by the computer and should be animated so that players get the feeling the dealing is done correctly.

Learning: Evaluate how much time should be used for the dealing animation.

3.9 Draw or Choose

When a player selects a card from a pile, it should be done the way the game allows/expects it. This means there should be more than one ways to draw a card, e.g. showing the pile (see 3.10) in full so that the player can choose exactly what is being searched for.

If the pile should remain hidden while the player draws a card, the cards should be displayed in an impartial way. What has to be considered in this case is that in real games, it is another player, or the dealer, who is presenting the cards to draw from. It is considered bad manner if the cards are presented in a biased way.

If the amount of cards that can be drawn is more than one card, the system should somehow show the player how many more cards are allowed to draw. However, if the player continues to draw cards, the system should warn but not deny drawing more cards.

Drawing a card from an opponents hand has a very emotional component that gets lost completely if the interaction with the element refuses to work (which breaks pliability). Hence, a very short and non-distracting animation should be used to imply that the player is choosing but has not yet decided. This has to be addressed differently on touch devices where no hover-state is available.

Learning: Evaluate for every drawing/choosing scenario if it can be automated, and how it can be visualized.

3.10 Flip or Show

There are two different objects that can be shown or flipped: a single card or a pile.

Some games have piles where only the top card is shown. In this case flipping the top card of a pile should be done automatically whenever the top most card is removed.

If a player has to show one or more cards of one of his hidden piles (e.g. the *hand*), the process of choosing which cards to show should not be visible to the other players. After the player has decided which cards to show, it should be in the players control how long the others can peek at them. So there has to be a mechanism of drawing back openly visible cards. Cards shown from a pile should form its own pile until the player or the system returns them.

Showing the contents of a pile should consider the screen's size. Fluency is broken if cards are resized when displaying them in a layout that fits a small screen. Resizing cards automatically would break fluency because you cannot resize real cards.

Learning: Evaluate which flipping/showing-interactions are automated and which are done manually.

3.11 Mark

Marking cards can be done in two ways: Changing its visual appearance (changing the card itself or adding something to it), or by moving it (changing its relative position or rotation). In either case, marking should not be automated unless it is totally clear to the player why it had to happen.

If a token is used to mark a card, the token can be treated as part of the card or the same way as any other game object. That includes the means of interaction: the token must be modifiable.

An example for marking a card by rotation is contained in the game *Magic: The Gathering*, where marking a card is called *tap*.

Learning: Decide how big tokens are in relation to other game objects and if direct interaction with it is allowed.

3.12 Move or Play

Moving a card from one pile to another[1] can take time in some games. However, most digital card games heavily automate playing a card. The system can automatically detect which piles allow incoming moves, and then distribute the card(s) accordingly. This does not break rhythm or pliability. A good example is *Hanafuda Koi Koi* where a lot of drawing, stacking and collecting goes on.

Some games might require thinking about the possibility of undoing or withdrawing a move. Normally this is considered bad manner and does not have to be addressed.

Learning: Drag&Drop, Click and Touch are only three possible ways to implement a move. Evaluate how important the move is in the real game and decide how to implement it upon those findings.

3.13 Sort or Order

For faster card recognition, players order the cards in their hand. Most games do the sorting automatically, some do not sort the cards at all and it is difficult to find the correct card.

Learning: Decide if you want the players to follow their own sorting method or an algorithm should do that.

3.14 Modify

Some games require the player to modify game object attributes. Other games strictly disallow it.

The types of attributes differ a lot from game to game. The recommended way to modify them depends on this type. The following attribute classes have been identified:

- sortable values, numbers, like rank, or health points
- enumerated values, like suit, or color
- flags and flip-states, like trump, or visibility
- additional rules

[1]To avoid potential confusion: I do not distinguish between pile-to-pile or hand-to-pile moves, as I understand the hand itself as a pile.

- additional exceptions

Most games have static card attributes and focus on how the player uses the cards. Games that allow dynamic card attributes mostly use counters or flags to show that a card has a lifetime, or has temporarily entered a different state.

The only general recommendations that are possible for card attribute manipulation is that it should be possible to change a value without complicated interfaces, and that every attribute should be clearly visible.

Flag-values, like the trump, however should not be visible, because it is part of the game to remember which suit is the trump suit.

Learning: The virtual nature of digital card games allows changed states to be indistinguishable from the original state without knowing the original state. Evaluate how state changes should be visualized and if it breaks game mechanics.

CHAPTER 4

Model

4.1 Introduction

Motivation: Finding an abstract Card Game Model

To create an universal abstract model for digital card games I proceeded as follows:

- declare the models data structure
- analyze static and dynamic aspects of card games
- analyze both possible and allowed state changes
- declare the models final structure

Are card games describable by an abstract model? What would be the benefits of having a such a model?

1. Games represented by a machine readable model could be executed by an engine. Developing new card games, or digitizing an existing one, is then reduced to extracting its characteristics.
2. Having such a model could help understanding basic card game principles. From those principles, it could be possible to derive game design patterns.
3. We would have a new way of classifying card games by their use of certain model aspects (not every game would use every aspect of the model).
4. Building a model to describe card games is a scientific approach to this field of research that is - to my knowledge - unprecedented.
5. The model is unbiased concerning rules. Rules are a part of the model and can be changed.

To find the model I identified each games characteristics and split them into static and dynamic attributes. To that end, I used the same games that I have used to extract the design patterns (see *List of card games* A). According to 2.4, I looked only at card games that use cards as primary instrument of gameplay. It is crucial for this approach to choose a broad variety of card games, with as many different characteristics as possible, to ensure that the model has as good a coverage of card game aspects as possible. The more often I iterate this empirical process, the better the derived model. To test and refine the model, I additionally picked random games and tried to apply the model to them to see if it is complete.

Possible Problems of an abstract Card Game Model

Some aspects of card games can hardly be modelled because they are elusive. *Incredibrawl* and *Minderheitenquartett* have such elements.

Another problem are game dynamics I was unable to identify, because I chose the wrong games.

Other problems are:

1. The model can only represent a subset of all games.
2. It is possible to intentionally create an instance that can not be represented by the model.
3. The model does not include other game mechanics such as a scoring boards (Bridge) or a betting system (Poker).

4.2 Model

Following, I describe the formal abstract model that I designed after analyzing the games A. The model is based on the basic objects described in 2.3. Then I explain the need for a rule checking algorithm, and how to transform real rules to model rules.

The Move

The model's basic data structure shown in Figure 4.1 is mostly trivial. However, the relations between the objects can be very complex. The move has a very specific role in card games and is therefore modelled separately. The basic actions "remove", "add" and "change" are omitted in the model above, but will be considered later. For example, in classic card games like *Schnapsen* the cards themselves can not be changed. In *Magic: The Gathering*, however, attributes, like health points, or attack points, can change. In the beginning of a game of *1000 Blank White Cards*, no attributes exist at all.

Removing the players from the model changes a lot, because the game state is therefore decoupled from who plays, and when. That means the player is not defined by the model, nor is the interaction between players. As a consequence the model does not take social skills or meta levels of gameplay into account. However, the model allows subsystems 4.2 dealing with these concerns if necessary.

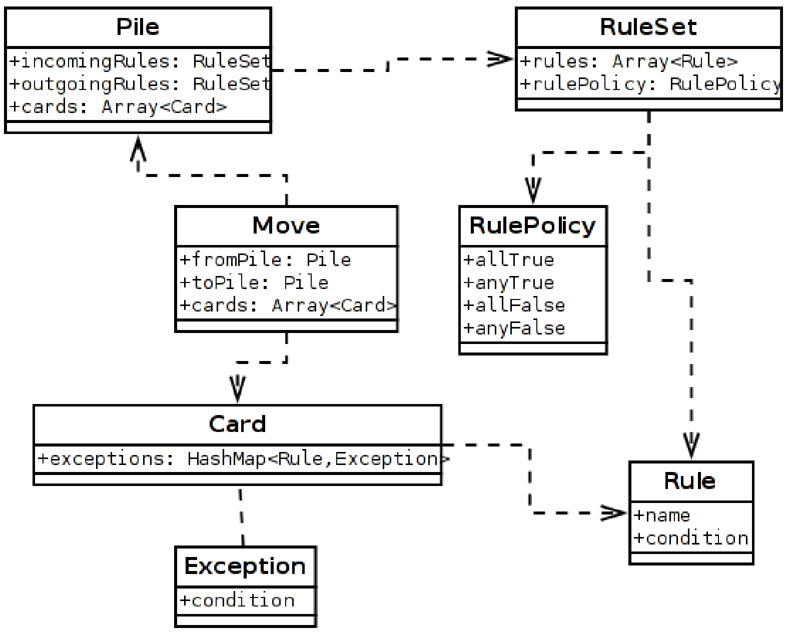


Figure 4.1: The move is the interaction relation between piles and cards. Rules define if a move is valid or not. Some cards can have exceptions to those rules.

Move vs Action - The Difference

Is it necessary to differentiate between an action and a move? Why isn't a move an action? The biggest difference between moves and actions is, that an action is a direct change of the game state, whereas a move only changes to which pile a card currently belongs to. An action can be applied to all game objects, tableaux, piles and cards, whereas a move can only be made with cards. A move is a more specific type of interaction and is used so often in card games that I had to isolate in the model. Actions offer the following interaction patterns like shown in Figure 4.2:

1. add objects
2. remove objects
3. change (create if not exists) attributes

The model allows events that can trigger actions or moves automatically. This enables the designer to create more complex or named interactions like *tap a card* in *Magic: The Gathering* or *deal* in *Schnapsen*.

Static vs Dynamic Gameplay - Identifying the Problems for the Rules Subsystem

In this subsection I identify the problems the model has to solve concerning static and dynamic aspects of card games.

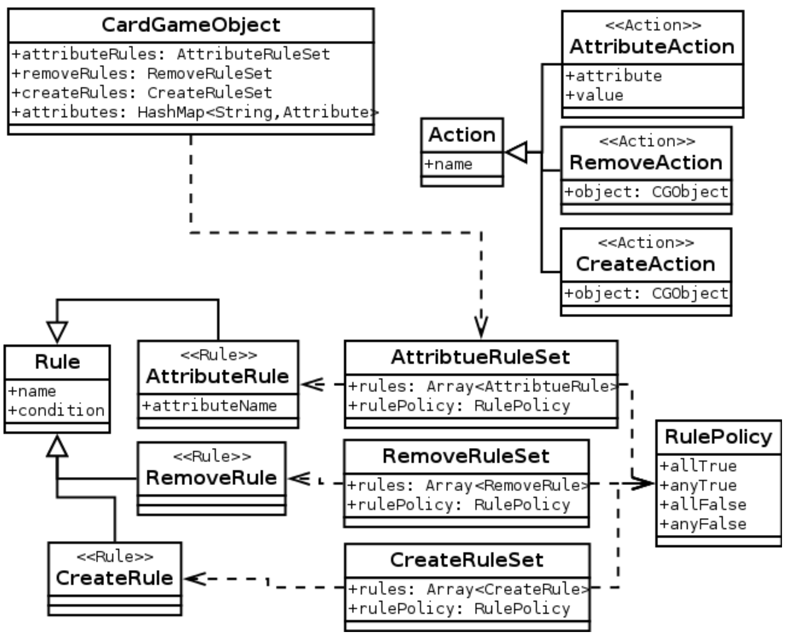


Figure 4.2: Actions are more complex than moves because they influence the game state directly and can be applied to all basic objects.

Card games have rule sets that are written in plain language, and do not change once a game starts: the static aspect of the rules. Most rules are written out of the player's perspective: what a player may do and what is forbidden to do. To be compatible with the proposed model, the rules have to be transformed into a machine readable format.

Examples of games, that allow no changing of rules during gameplay, are *Schnapsen*, *Poker: Texas Hold'em* and many more, a counter-example is *1000 Blank White Cards*.

Games with static rules allow the player to develop strategies based on statistics and mathematics. In *Poker: Texas Hold'em*, sophisticated players know the probabilities of certain combinations by heart. Therefore, they are able to estimate their chance of winning.

In *1000 Blank White Cards*, other strategies, like creativity, or ingenuity have to be explored.

A lot of rules have exceptions allowed only in certain situations, which is a dynamic aspect of rules. *Jassen* implements the patterns *Follow The Suit* 5.4 and *Trump* 5.4. But: If the only trump left in your hand is the under (also known as jack or knave), you are not forced to play it. This is an exception to a simple rule: playing a card, you must match the trump suit unless the only trump you hold is the under. The model has to be able to deal with the *problem of exceptions*.

As mentioned before, there are games that have only a small set of static rules in the beginning of a game and develop their rules as a mechanic of the game. Examples are *1000 Blank White Cards*, in which players make their own rules, and *Dominion*, in which different card effects unfold by following the story. Not only the rules, but also the attribute dimensions of game objects can change.

A more structured approach to rules that are dynamically “created” can be observed in games that allow the optional use of *The Wild Card* 5.6, like *Rummy*, which can also be played without a joker. In such a game the structure of rules has a meta-level, which en- or disables certain rules. Other games, like *Android Netrunner*, even have completely different rule sets depending on which role a player takes: The *Runner* has to steal Agendas from the *Corporation* and both roles have different rules and strategies to achieve their goal.

Also, there are extensions to existing card games called *variants* or *variations*. The rules that make the difference to the original game are mostly unwritten family rules that have been introduced to make the game easier, more amusing, or quicker than it was designed to be. Rules defined in this manner have highly dynamic aspects because most of them add exceptions to existing rules. Sometimes they are even created while playing the game.

The model has to be able to deal with the *problem of dynamism*.

I chose to use *modular subsystems* 4.2 for the model. The *rules model*, as one of the subsystems, will solve the above mentioned problems.

It might seem logical to build the rule data structure with a dependency to the player, concerning most human written rules are formulated in the following way: “When a player X then Y.”, but this model is built upon game objects, actions, and moves. The player’s abilities to interact with other players or the game itself are not compromised by that fact. Players still feel the same freedom as in a real playing situation which is an important aesthetic 2.4.

Subsystems

A subsystem is a library concerned with a specific aspect of the game state independent of other subsystems. Every subsystem has its own architecture, but reads and stores its data on the same shared game state. All subsystems are accessible by the conditions in rules and exceptions, therefore allowing complex game state dependencies, as required by some games. The following subsystems will be explained:

1. **Rules:** which piles have which incoming our outgoing rules, and which other rules exist?
2. **Player order:** how the game is structured: is it turn based, or something different?
3. **Events:** every interaction might trigger events in certain situations
4. **Counters:** most games require the possibility to mark a card as a special card and/or count something for a pile or card
5. **Scoring:** deciding how many points a player gains for which action is often non trivial
6. **Ranking:** the rank of a card often decides who wins the trick

Rules

As pointed out in *Static vs Dynamic Gameplay* 4.2 the model has to solve two basic problems: exceptions, and dynamics. The rules model I came up with is yet another approach (to existing approaches [26], [58], [20]) which uses *rule contexts* and *policies*, rather than defining one function, that operates on the whole game state.

Using *rule contexts* divides the algorithm to check the rules into two parts: the preparation of the context (query, filtering, and mapping of the game state) and the resolution to a boolean value.

I will explain the rules model with an example, the "trump-under-rule" from *Jassen* mentioned in 4.2:

1. the *center pile* (the context) gets a rule named "Follow-The-Suit"-rule.
2. once the trump is defined, the *trump under* gets an exception referencing the "suit-rule"
3. once a move triggers a rule-check on the center pile (by a move), a rule context is created. In this case an "Incoming Rule-Context" is created from the more basic move-context. It always contains an "incomingCard".
4. the next thing happening, is that all incoming rules on "center" are iterated and the basic rule-context is extended by the results of the query for each rule. In this case, the *query* of the rule fetches the pile the card originally came from and names it "originalPile".
5. if the incoming card has an exception to a rule, the rule is ignored.
6. finally the rule (pseudo-code) can be checked: "incomingCard.suit == this.cards[0].suit || this.cards[0].suit NOT IN (originalPile.cards.suit)"
7. if the incoming card is the *trump under*, the rule checking algorithm evaluates to no result, because the pile has only the "Follow-The-Suit" rule, and this card is an exception to it. The policy in this case allows the pile to accept the card.

Player Order

The player order defines which player is in control (play an action or a move), while the other players have to wait. A turn might be split into phases the player has to declare. This is the case in *Magic: The Gathering* or *Android Netrunner*.

Uno implements *Jump-In* 5.2, which allows players to play a card even though they are not in control at that moment. But the game implements *Turn Based* 5.2, and so the turn ends, and the control is passed to the player who intercepted the current players turn.

A typical example of continuously played games is *Speed* sometimes also called *Spit*. In this game the objective is to get rid of your cards as fast as possible, making physical speed and alertness the important winning strategies. Games like this, implementing *Event Based* 5.2, have no need for a player order and will not use this subsystem to organize control.

Events

In most trick taking card games like *Schnapsen*, *Canasta*, or *Jassen*, finishing a trick is an event that triggers the cards of the center to be moved to the winners score-pile. In the real life, players have to do this move manually. The model allows the events-subsystem to detect the end of a trick and automatically trigger the move by firing the event "trickFinished".

Events can also be used in games where a counter on a card counts down until it reaches zero (event “counterZero”) and the card has to be removed. This event can automatically remove the card from the game.

Another example indicating the requirement fore an event subsystem can be found in games implementing *Disclosure* 5.4 (e.g. “marriage” in *Jassen*). The player shows cards from the hand to the others, decides which card to play for a trick, and the rest is moved back to the players hand. A possible implementation would be to move the cards to a temporary pile. From that pile the player decides with card to play or draw back. A “post-move” event on that temporary pile moves the remaining cards in the marriage to the corresponding pile (hand or center).

Counters and Markers

Counters 5.7 and Markers are an important addition for many card games. The principle of a counter is to add a temporary attribute to a card or a pile with a number that is in- or decreasing. A counter that does not change over time acts as a tag or flag. The counter’s or marker’s name is important as it is saying what they are counting, or what is different to the object without the marker. Most counters are de- or incremented manually, other counters may be triggered by events.

Markers can even be structured hierarchically, or by placement. In *Cardcassone*, the players piece is used to show if the player occupies a lawn or a street simply by how the piece is placed: lying on the belly means the piece represents a farmer, standing on its feet means that it owns a city or street.

Scoring

Many card games do not have a simple winning condition but use a sophisticated scoring system. Calculating a score for a played game and building the sum over multiple games is a popular pattern.

In *Skat*, you have to reach a minimum amount of points in order to play on. If you have less than 30 points you are “Schneider”, and that means the points are doubled or tripled.

Contract Bridge uses a complex score counting algorithm using a scoring board: A player can gain points in different categories (contract, over trick) and can gain boni (Slam, Double).

Similarly to *Skat*, the score you gain depends on how you estimate the outcome. The scoring subsystem has to keep track of bets and events over time to calculate the score correctly.

In *Rummy*, a player can use multipliers on tricks to improve their score. The scoring subsystem has to keep track which multipliers have already been used.

Ranking

To decide if a trick is won or how many points a player gains for, it a card comparing algorithm is required. Normally a trick is won when one card outranks the others in the trick. *Rummy* implements *Natural Order* 5.9, and higher order cards outrank lower order cards (eg. a King outranks a Queen).

But in this abstract context the idea of “winning a trick” can be interpreted differently. In *Magic the Gathering*, for example, a trick is the battle of two creatures. The ranking algorithm then decides which is the stronger one. In a hypothetical implementation of the game it might trigger an attribute change events that decrease the health points.

The “jack-under-rule” example mentioned in the *rules subsystem* 4.2 could alternatively be implemented by introducing a different ranking-algorithm for the trump suit, which itself changes every game.

CHAPTER 5

Card Game Design Patterns

5.1 Introduction

The patterns found through analyzing the games listed in A are described using the method covered in section 1.3 of the Introduction 1.3. The patterns are documented using five attributes: name, description, consequences, usage, and relations.

The patterns have been extracted with consideration for relevance to card games. However, their descriptions partly refer to more common game design patterns or are even general patterns. This list is the result of the research done in the context of this thesis. The categorization of patterns is specific to card games and organized from a card game designer's point of view, and thus, not consistent with a general pattern classification.

The attributes used to describe the design patterns are taken from [10] where they are explained in detail.

1. **Alternative Names:** the name is an easy to remember shorthand for the pattern. Some patterns are known by alternative names listed here.
2. **Description:** context in which the pattern was found and how it works in general.
3. **Consequences:** some patterns have (dis)advantages, some promote other consequences. This section lists possible consequences of applying this pattern.
4. **Using the pattern:** this section brings the pattern into a bigger context and describes choices or situations a designer is faced with using this pattern.
5. **Relations:** most patterns can not be used alone or are implicitly related to others. This section reveals those relations. Relations are (a)symmetric and there are three types of them: **instantiates**, **modulates**, **conflicts**. This produces the five relations a certain pattern can have: *instantiating*, *modulating*, their counterparts *instantiated by*, *modulated by* and *potentially conflicting*.

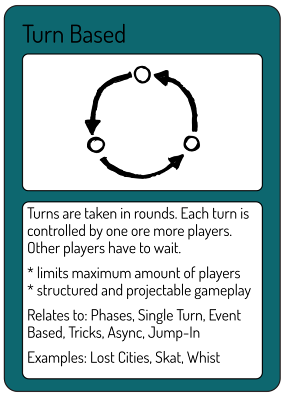
5.2 Continuity Patterns

There are two types of patterns in this category: discrete and continuous patterns. Most games use a discrete form of scheduling the user interaction, so the *Turn Based* approach is most commonly used.

Turn Based

Alternative Names:

Turn Based (Round Based)


Turn Based

Turns are taken in rounds. Each turn is controlled by one ore more players. Other players have to wait.

* limits maximum amount of players
* structured and projectable gameplay

Relates to: Phases, Single Turn, Event Based, Tricks, Async, Jump-In

Examples: Lost Cities, Skat, Whist

Figure 5.1: Pattern *Turn Based*

Description:

The game's time structure is organized into slots, and each slot is controlled by one or more players. Every player gets the chance to control the game until a specific action or move is finished, or a given amount of time has passed and the turn ends. Taking control, a player can make moves or take actions 4.2. After the player finished their interaction, the next player takes over and so on. The one(s) in charge change following a pattern defined by the rules. The most common pattern being the seating arrangement of the players around the table: clockwise or counter clockwise.

Every trick taking game like *Schnapsen*, *Whist*, or *Skat* implements this pattern. Games with asynchronous gameplay, like *Lost Cities*, require this pattern. Turns can also be taken in teams.

Consequences:

While one player is in control the others are forced to wait. Some games allow the waiting players to intercept (5.2) the current player when certain events occur. On the other hand, turn based games allow the waiting players to control fair play (they can check if the others follow the rules). In digital card games this should be done by the computer. In synchronous digital card games waiting can get very annoying, especially when there are a lot of players. Therefore, a player's actions must be observable and comprehensible.

In my tests, players experienced turn based games as structured and project-able, whereas simultaneous gameplay (like in *Speed*) was perceived as being faster and allowing for less thinking time.

Using the pattern:

When designing a card game, this might be one of the first and most influential design decisions. The core of a card game is defining who has control over the cards, when, and for how long.

When using this pattern, the other players have to wait for the player in control. There are several ways to solve the waiting problem. The naive approach of reducing the maximum number of allowed players is not a good idea. Limiting the amount of players should be used to balance the odds rather than solving the waiting problem.

Patterns, like Jump-In 5.2, or limiting the absolute amount of actions, are only two possibilities to overcome the aforementioned waiting-problem. Interestingly, most digital card games I analyzed did not put much effort into designing the aesthetics of the waiting process. While it is very important to optimize the user experience for the player currently in charge, the others are idling and their game experience must not be forgotten.

That is why most digital card games either implement an (audio-) chat when played synchronously, or inform the players that they do not have to wait actively, because they will receive a reminder (e.g. a push-notification) when the other players have finished their turn. Sending a reminder makes the game asynchronous 5.10.

When the game uses Tricks 5.4, the designer can decide if players have time to think about their reaction, or if every player has to decide upfront which card they want to use in the trick.

Relations:

Instantiates Phases 5.2 even if there is only one phase.

Potentially conflicts with Event Based 5.2.

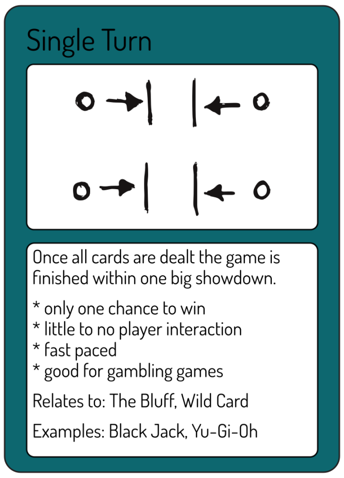
Conflicts with Single Turn 5.2

Modulates Tricks 5.4, Async 5.10 and Jump-In 5.2.

Single Turn

Alternative Names:

Single Turn, Single Round, Showdown



Single Turn
Once all cards are dealt the game is finished within one big showdown.
* only one chance to win
* little to no player interaction
* fast paced
* good for gambling games
Relates to: The Bluff, Wild Card
Examples: Black Jack, Yu-Gi-Oh

Figure 5.2: Pattern *Single Turn*

Description:

Once all cards are dealt, the game is finished within one big showdown. Every player gets the chance to present the choice and arrangement of cards once only. It does not matter how many actions happen before the showdown begins (e.g. cards are dealt, etc), the important thing is that every player has at most one opportunity (sometimes none) to play their hand.

Playing with a spinning top is like a single turn in a card game: the game starts with only one interaction and is over when the spinning top stops.

Depending on the rules, the game may already be over once the spinner is rotating. Therefore, most games implementing this pattern are gambling games like the medieval game *One and Thirty* (first described in [79], a very interesting book originally written, but not finished, in the seventeenth century about medieval sports and games), an old ancestor of *Black Jack*. In such games, each player is dealt cards until they decide to stop and finally, each players hand is compared. Depending on the winning conditions, one hand besieges all others, choosing a winner.

There are some turn based card games that can be decided within the first turn, without ever giving the other players a chance, which is a special implementation of the single turn pattern. *Yu-Gi-Oh* has a name for this game dynamic, calling it “First-Turn-Kill” [16], an established strategy.

Consequences:

Games offering only one chance to win can get imbalanced, because players cannot react to each other. Some games, like *Black Jack*, counter this consequence by allowing only one card to be drawn until every player refuses to draw further cards, thus weakening the pattern.

Players do not really play against each other because there is no interaction between them.

Using the pattern:

This pattern limits which actions a player can choose from, so the designer should try to focus on meta game patterns when implementing this pattern. *Poker: Texas Hold’em* is a great example for that by introducing *The Bluff* 5.3. This makes the game more exciting and allows the players to develop strategies (based on chance and probability) without completely changing the main game mechanic.

Also, *The Deck* 5.8 of cards has to be chosen according to the specific rules, to avoid imbalance in chances of drawing.

Relations:

Potentially conflicts with Wild Card 5.6 outbalancing card value/influence.

Promotes The Bluff 5.3

Conflicts with Turn Based 5.2

Jump-In

Alternative Names:

Jump-In, Interception (Zwischenwerfen)

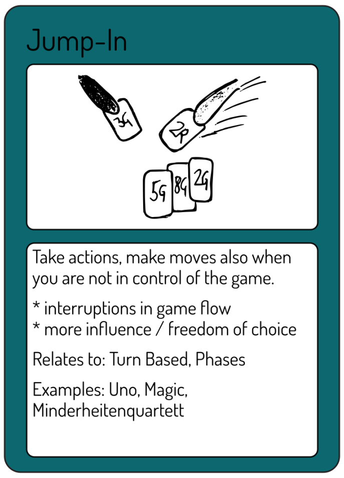

Jump-In
Take actions, make moves also when you are not in control of the game.
* interruptions in game flow
* more influence / freedom of choice
Relates to: Turn Based, Phases
Examples: Uno, Magic, Minderheitenquartett

Figure 5.3: Pattern *Jump-In*

Description:

A player can theoretically take actions at any time. Most card games forbid such behavior and use patterns like *Turn Based* to clearly separate any action by other players to the one currently in control. Implementations of Jump-In, as an exception to this strict agreement of non-interference, are very different. A variant of *Uno* allows a player to discard a card at any time, if it matches the one just played. In *Magic: The Gathering*, every time a spell is played, a player is allowed to play an instant/interrupt. While two players fight over two cards in *Minderheitenquartett*, a third player can take sides and play a booster to support or handicap one of them.

Consequences:

Allowing an interception can be problematic if it takes too long. In the above mentioned variation of *Uno*, it is possible for multiple players to have the same card. As long as one of them still holds one, the interruption may go on, and sometimes it is difficult to say who has played the last card, to determine who is next.

The games mentioned above use interceptions as a means of balancing player or card power influence. Jump-In increases a player's level of influence 2.4 and therefore allows more freedom of choice. When a card exists that reads "You win the game", other players can only fight it using interrupts.

Allowing interrupts will force the current player to anticipate them and carefully choose the actions accordingly.

Using the pattern:

The duration of an interception should be kept as short as possible to minimize the downtime for the other players. *Magic: The Gathering* has an interrupt-card saying "Counter target spell.". Plain, simple, and straightforward, meaning it does not take long to read the card. The interaction following this interruption normally takes seconds, not disturbing normal gameplay.

Although interrupts are a possible balancing method for overpowered cards, randomness through dealing and drawing can dampen the effect.

Also see the paragraph about balancing in **Consequences**.

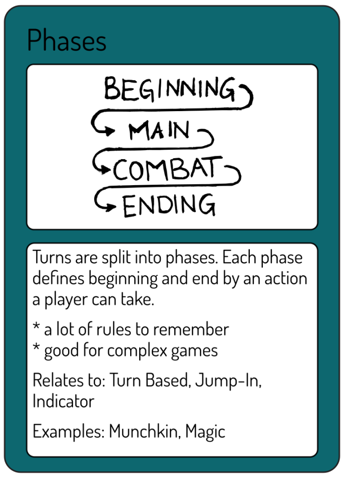
Relations:

Can be instantiated by Turn Based 5.2.
Can be modulated by Phases 5.2.

Phases

Alternative Names:

Phases, Sequences


BEGINNING
MAIN
COMBAT
ENDING

Phases

Turns are split into phases. Each phase defines beginning and end by an action a player can take.

* a lot of rules to remember
* good for complex games

Relates to: Turn Based, Jump-In, Indicator

Examples: Munchkin, Magic

Figure 5.4: Pattern *Phases*

Description:

Some games have a wide variety of possible actions a player can take. Because of this complexity, many card games that are hard to learn like *Munchkin*, *Magic: The Gathering*, *Yu-Gi-Oh*, and others alike, implement turn phases. Normally, turn phases restrict the actions allowed within them. Some actions, or moves, require some precondition, or force some follow-up action. For example, in Magic, you can't pay for a creature if your resources are tapped 5.7. Therefore, the action *untapping* is a precondition for the action *bring a creature into the game*.

Consequences:

Clearly structured phases require the player to remember every step. In real life games, a player forgetting a certain step is normally punished in some way. Most digital card games however make implicit use of phases as the player cannot circumvent or change the defined sequence.

Phases also have the side effect that players tend to develop tactics for every phase and compare them to the actions other players take in the same phase.

Using the pattern:

It is important to clearly define which action starts or ends a phase. One can choose how to define the transitions between the phases. Either make it clear when a player can't go back, or what a player is allowed to do in a certain phase.

Implementing this pattern in a digital card game requires the designer to make use of events so that the basic card movements and actions determine the current phase unambiguously.

Indicators (5.5) are a good way to visually display which phase the player currently passes.

Relations:

Can be instantiated by Turn Based 5.2.

Modulates Jump-In 5.2.

Potentially conflicts with Event Based 5.2.

Event Based

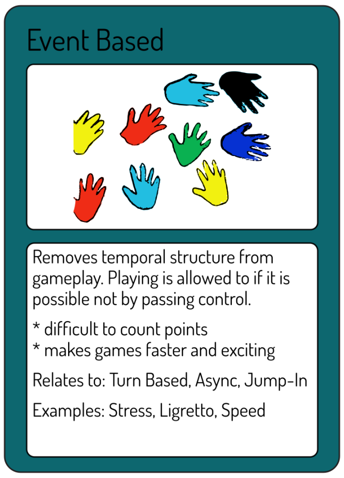
Alternative Names:

Event Based, Play-When-Possible

Description:

Removing temporal structure (e.g. turns) from a game makes it event based. Taking an action, or playing a move, is only restricted by fact-based rules. All players play simultaneously, without waiting for each other, but waiting for something to happen. Normally the game is over once the first player reaches the objective, like losing your hand. Games like *Stress*, *Spit*, or its variation, *Speed*, have no time-related, but event-related rules. The question, if an action or a move is allowed, is answered by the cards every player can see. There is no player who leads the hand.

A good example of parallel gameplay is the game *Ligretto*, using four colors and cards ranging from one to ten. In a game of three players, every player tries to get rid of 13 cards. Only four of those cards are laid out in front of the players. Those cards can be used to build up piles from one to ten in the center of the table. Every player tries to build those piles simultaneously. The player who first finishes removing all cards wins the game.



Event Based

Removes temporal structure from gameplay. Playing is allowed to if it is possible not by passing control.

* difficult to count points
* makes games faster and exciting

Relates to: Turn Based, Async, Jump-In

Examples: Stress, Ligretto, Speed

Figure 5.5: Pattern *Event Based*

Consequences:

The dynamics such games produce are heavily dependent on the players ability, experience, and skill. Normally, a player with high dexterity wins over a player with high skill in the long run and vice versa in the short run.

It is interesting to notice that players who are related to each other tend to add their own rules to event based games. This normally happens to balance experience versus skill. In the context of this pattern, the custom rules I discovered primarily are restrictions. They start like this: “The eldest player may not...”, “If there were three cards of type X, a fourth is not allowed”, and so on.

People who have already played such games have much higher chances to win than newcomers, who first have to adapt to the new event based pattern.

As counting points is very hard in fast paced games, normally the player wins who’s cards have been played first.

Using the pattern:

The basic non-breakable rule in event based games is that a player has to wait for a certain cue to make their move or action. It is up to the designer to define which actions/moves allow which.

Creating an action-move-dependency-tree helps both understanding and explaining the game's mechanics and getting a general idea of which dynamics might emerge.

While most event based games use a very small set of rules, these try to exploit the human cycle of perception and reaction. Examples are: similarity to exploit confusion, placement to exploit hand-eye coordination, or parallelism to exploit attention.

Relations:

Conflicts with Turn Based 5.2 and Async 5.10.

5.3 Player Interaction Patterns

As a player can choose between an action or a move 4.2, player interactions are limited to the same constraints. In the context of card games, an interaction between players means passing a card or passing control. In most cases, only two players interact, however, some games implement multi-player interaction.

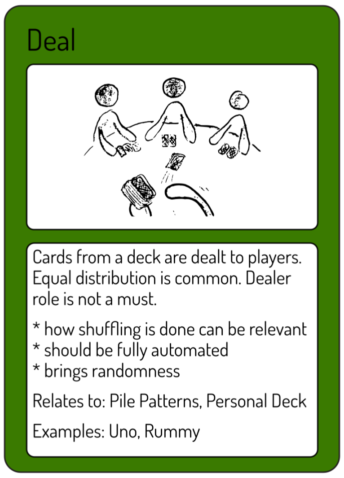
Deal

Alternative Names:

Deal, Distribute, Assign

Description:

The beginning of every card game has the more or less important part in which the cards are unpacked and distributed to each player. Most games distribute the same amount of cards to each player and even remove cards from the deck (pre-shuffle and pre-deal) so that the deck can be divided evenly. Some games trivialize *The Deal*, others hold it sacred. Cutting 3.7 and shuffling 3.6 are interactions involved with dealing cards. One could literally write a book about shuffling alone. David Bayer and Persi Diaconis [6] came to the conclusion that it takes at least seven *riffle shuffles* to thoroughly mix a deck of 52 cards. A riffle shuffle is a shuffle where cards are split into two packs and those packs are mixed back together.



Deal

Cards from a deck are dealt to players. Equal distribution is common. Dealer role is not a must.

* how shuffling is done can be relevant
* should be fully automated
* brings randomness

Relates to: Pile Patterns, Personal Deck

Examples: Uno, Rummy

Figure 5.6: Pattern *Deal*

Most classic card games have rules about who deals the cards and who follows as the next dealer. Others have no rules about dealing. Collectible card games promote every player having their own cards, so no dealing in the classical meaning is required. In *Bartok*, all cards are placed face down in the center of the table, and the players have to draw their own starting hand.

Examples for games without a dealer are games like *Solitaire*, or games in which the cards have to be ordered on piles, like *Spit*, or *Skip-Bo*, where the deck is split rather than dealt.

Consequences:

In games that hold more cards than the amount of cards being dealt before the game starts, like *Uno*, or *Rummy*, special piles are being used to store them. E.g. *The Base* 5.5, bidding, drawing, or scoring piles 5.5.

Dealing is a way of bringing randomness into the gameplay and therefore the cards have to be balanced accordingly.

Using the pattern:

In digital card games, dealing has to be implemented in a way the player can easily follow the procedure by using animations. The time used for dealing in the games researched is more or less constant, whereas the amount of cards being dealt can vary.

It is important that the player does not get the impression of biased dealing. Therefore, dealing is normally implemented as an automated phase in gameplay. An exception is *Skip-Bo*, where the player can choose how many cards should be dealt. In most other researched digital games, dealing happens automatically, as there is nothing to be decided. Avoid interaction whilst the deal animation is running.

Most digital card games use more or less advanced animations to improve interaction aesthetics. In the digital card games I looked at, no implementations of this pattern included any interaction.

Relations:

Instantiates one or more pile patterns 5.5

Potentially conflicts with Personal Deck 5.8

Discussion Points

Alternative Names:

Discussion Points, Voting, Amend or Repeal, Argue, Verbal Proof

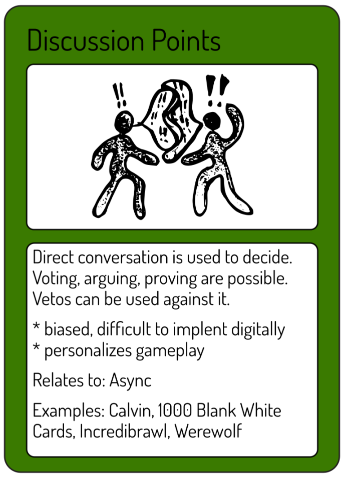
Description:

Discussion, as a game mechanic, can be used either to pursue a game goal or to change the game itself. The games *Calvinball*, *Nomic*, or *1000 Blank White Cards* implement Discussion Points to allow players to define and modify their own rules in the course of the game. Games like *Resistance* or *Superhero Audition* use vote-cards to decide which character or player stays in the game.

In *Eleusis Express*, each player may guess the secret rule.

In one variant of *Incredibrawl*, Discussion Points are used to let a player argue as to why a card belongs to a certain category. If the other players agree, the player who has won the argument gets a new location card as a reward.

Another example of Discussion Points can be found in the game *Bartok*. At the end of each round, the winner creates a new rule which remains in play for every following round. The creation of a new rule can be rejected by veto and then has to be discussed.



Discussion Points

Direct conversation is used to decide. Voting, arguing, proving are possible. Vetos can be used against it.

* biased, difficult to implent digitally
* personalizes gameplay

Relates to: Async

Examples: Calvin, 1000 Blank White Cards, Incredibrawl, Werewolf

Figure 5.7: Pattern *Discussion Points*

Nomic has been played on forums and email-groups for years and has transformed itself from a card game to a new digital interpretation.

Some games implement an in-game chat, that allows players to communicate. This mechanism normally has no direct consequence on gameplay.

Consequences:

Discussions have the potential to be lengthy and unfocused, as it is the nature of all things that can not be formalized. In the games I researched, groups that have been chosen randomly started struggling and heavily arguing. In some cases, the game even had to be stopped because no agreement was possible. Groups that play regularly together and know each other tend to have less problems with discussions.

Discussions require a means of communication and cannot be structured or modelled in a way that they could be moderated by a computer. Therefore, some games transform verbal discussion to a structured vote.

Implementing a direct way of communication changes the game to be more of a simulation than a asynchronous card game.

Using the pattern:

Most games using this pattern include every player in the discussion, but sometimes only two opponents can argue over a fact. Some games help speeding up and mediating the discussion by declaring roles, like the discussants, judge or observers.

One interesting implementation can be found in variants of *Werewolf*. Players are connected using their smart phone, and the werewolves vote which inhabitant should be devoured next.

Relations:

Potentially conflicts with Asynchronous Gameplay 5.10

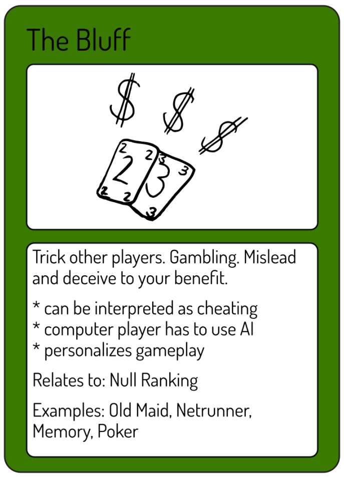
The Bluff

Alternative Names:

The Bluff

Description:

This pattern arises from game dynamics and can be found in many card games like *Old Maid*, *Android Netrunner*, *Poker: Texas Hold'em*, and even competitive variants of *Memory*. The Bluff itself requires no interaction with cards, because players interact on other layers. The Bluff is all about tricking your opponent into believing that you have, or do not have, something (points, cards, ...). The goal is to make the other players anticipate a certain action or behaviour, but to act differently and get closer to winning the game. Most of the time, bluffing is connoted negatively, as in misleading, deliberately deceiving, being rough, or having an unfair advantage. But as soon as a game contains The Bluff, every player is allowed to, and as such it is the skilled bluffer who wins.



The Bluff

Trick other players. Gambling. Mislead and deceive to your benefit.

* can be interpreted as cheating
* computer player has to use AI
* personalizes gameplay

Relates to: Null Ranking

Examples: Old Maid, Netrunner, Memory, Poker

Figure 5.8: Pattern *The Bluff*

Consequences:

Games showing this pattern rely on randomness and hidden information. Both ludemes are implemented by almost every card game and, as such, the Bluff should be possible in every competitive card game.

When there are no rational means left to optimize your winning strategy, players try to rely on other patterns, like guessing the opponent's intents or strategy. It normally takes some time before a player masters The Bluff, because it can not be learned from a rule book, or from counting cards. It can only be learned by playing other players.

Using the pattern:

A designer cannot build The Bluff into a game, nor can it be avoided to be included to a certain extent. It is a dynamic that is used by players to their own advantage. However, the designer can facilitate it by creating situations in which players are confronted in such a way, that the next decision's outcome is dependent on cards controlled, but hidden, by the opponent. It can be further facilitated by reducing the amount of card game patterns that involve real interaction and by increasing the ranking complexity.

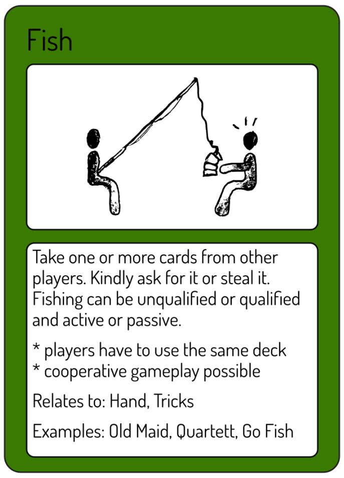
Relations:

Modulates Null Ranking 5.9

Fish

Alternative Names:

Fish, Steal


Fish

Take one or more cards from other players. Kindly ask for it or steal it. Fishing can be unqualified or qualified and active or passive.

* players have to use the same deck
* cooperative gameplay possible

Relates to: Hand, Tricks

Examples: Old Maid, Quartett, Go Fish

Figure 5.9: Pattern *Fish*

Description:

Choose and take one or more cards under opponent control. In my research, I found fishing in different implementations: (un)qualified, and/or active, or passive.

In *Old Maid*, fishing is a rule, and the player must draw a card. Thus, it is unqualified and passive as it is a self-contained game action, and the outcome is purely based on chance. This means that the player has no possibility of influencing which card, or type of card will be drawn. Other examples can be found in *Quartett*, or *Go Fish*, where fishing is the main mechanic, and players explicitly ask for a certain card, suit, or rank.

Active fishing can be found in games that have cards which allow a player to fish another card, like *Magic: The Gathering*. In this example, a player can steal cards from others any time, making it an active call.

Consequences:

Letting a player look at an opponent's hand before choosing a card changes the odds and may unbalance a game.

In every implementation, this pattern allows a player to guess what other cards the opponent might hold and derive strategies or tactics from that information.

After stealing one or more cards without replacing or exchanging them, players might hold different amounts of cards.

Using the pattern:

The designer has to decide upon the level of fishing qualification. Some games let the player call for a suit or a rank, which is not the same as letting a player look at the opponent's hand. Most games, however, only allow stealing one card by chance.

If the game uses tricks, and the amount of tricks is used for scoring, it is important to replace stolen cards to keep the amount of cards controlled by each player in equilibrium.

Relations:

Instantiates The Hand 5.5
Modulates Tricks 5.4

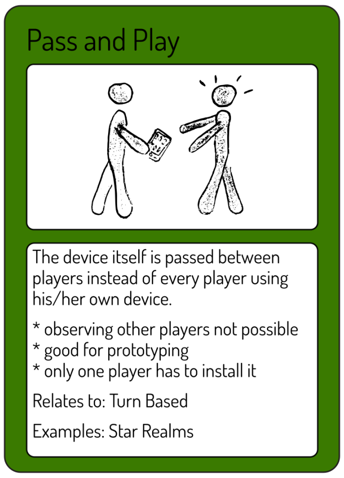
Pass and Play

Alternative Names:

Pass and Play

Description:

Being a purely digital design pattern, *Star Realms* and other digital card games implement it as an alternative to online gaming or playing versus an AI. As many mobile devices are small, designers face the problem of object placement and visibility. Splitting the screen so that every players state is visible at all times takes a lot of space. Without splitting the screen, the device has to be handed from player to player.



Pass and Play

The device itself is passed between players instead of every player using his/her own device.

* observing other players not possible
* good for prototyping
* only one player has to install it

Relates to: Turn Based

Examples: Star Realms

Figure 5.10: Pattern *Pass and Play*

Consequences:

In games like *Chess* or *Memory*, all players have to know and perceive the game in the same way, at all times. In games with a concealed hand, players hide parts of the game state from other players and therefore this has to be possible using Pass and Play too. Keeping the hidden parts secret can be difficult using Pass and Play.

When the device was passed, all other players have no information about the game state and can not do anything but wait. This possibly gets dull.

Using the pattern:

Pass And Play can be used instead of implementing distributed game state synchronization. It also can be used if the game does not have to be played online.

The passing phase of this interaction should be designed to be as easy, but secure, as possible. A player might have to lock the screen before passing the device so that the hand stays concealed. Once the opponent unlocks the screen, only the opponent's hand is visible.

Relations:

Instantiates Turn Based 5.2

Modulates The Hand 5.5

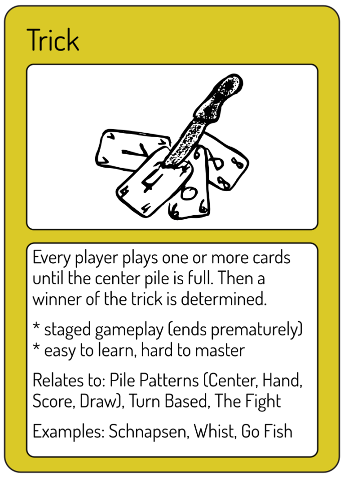
5.4 Trick Patterns

Tricks are the systematic, rule-based approach to decision taking in card games. The players use the trick-taking-rules to decide who has won a round. Tricks are strongly connected to scoring systems of card games, but some games allow tricks to have different effects.

The Trick

Alternative Names:

Trick



Trick

Every player plays one or more cards until the center pile is full. Then a winner of the trick is determined.

* staged gameplay (ends prematurely)
* easy to learn, hard to master

Relates to: Pile Patterns (Center, Hand, Score, Draw), Turn Based, The Fight

Examples: Schnapsen, Whist, Go Fish

Figure 5.11: Pattern *Trick*

Description:

The Trick is one of the most common card game patterns and is implemented in various ways. A typical Trick can be found in *Whist*: first, player A moves a card from *The Hand* 5.5 to *The Center* 5.5. All other players (there are four players in Whist) play a card until the Center is full. After the last player in this turn is finished, the cards are compared and a winner or “taker” is named. In the case of *Whist*, the player who played the highest ranking card wins.

There are two basic ways to determine the winner of the game: *plain*, which means the amount of tricks taken after a series of turns determine the winner, or *point counting*, where the sum of the value of the cards won in the tricks decide.

Consequences:

Trick based games are strongly structured, because players prepare it, and it has strong repercussions to the winning strategy. This is also one of the reasons why trick taking games “are by far the most varied and widespread form of card-play in the West.” [57].

The *Schnapsen* might be decided before the last trick is taken, because Point-Counting-Strategy is used to determine the winner.

Using the pattern:

The designer has to balance complexity between trick- and point-counting-logic. Most games I researched put more complexity into point-counting, as to how a trick is won. None of the games visually showed the current value of the trick or the value of the trick after it was taken. It seems as if most designers want the player to keep track of the points.

Besides the visualization of the trick, the designer has to decide how much space is reserved for the Center pile. An online variant of *Go Fish* I found uses a lot of space for showing the hidden hands of all players, and only small space is left for the center pile where all the action happens. I did not find a game which implemented The Trick in a way emphasizing the trick as the central game mechanic and aesthetically beautiful at the same time.

Relations:

Instantiates Turn Based 5.2, The Hand 5.5 in concealed form, The Center 5.5, The Score Pile 5.5

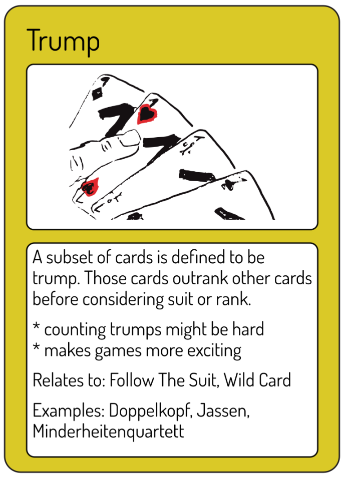
Modulates The Draw Pile 5.5

Potentially conflicts with The Fight 5.4

The Trump

Alternative Names:

Trump, Atout



Trump

A subset of cards is defined to be trump. Those cards outrank other cards before considering suit or rank.

* counting trumps might be hard
* makes games more exciting

Relates to: Follow The Suit, Wild Card

Examples: Doppelkopf, Jassen, Minderheitenquartett

Figure 5.12: Pattern *Trump*

Description:

Trumps are a subset of the deck's cards, ranking higher than any card in the game. Normal trick taking games like *Whist* or *Skat* use the trump pattern in so called "suit games". A suit game is a game in which a trick is decided upon suits firstly, and ranks only secondly. A game using Trumps (eg. *Doppelkopf*) defines a special subset of cards as trump which are considered before suits and ranks.

More generally, trumps are cards that outrank all or a subset of the other cards.

Minderheitenquartett has a more complex implementation: certain groups of cards are trumps only against another group of cards, narrowing the Trump definition.

Consequences:

Some games define which cards are trump, others let a player choose a suit or something else to identify the trump cards.

In games where a player can choose the trump suit, the game has to be balanced by repetition and switching the player who can choose. To set a limit to the amount of games that have to be played, most games define a more or less complicated scoring mechanism.

Memorizing which trumps have already been played is a key tactic to winning the game. However, none of the digital games I analyzed helped the player with this task.

Using the pattern:

Games that already have a trick taking mechanism can be spiced up by using this pattern. Games without trumps are easier to play and understand.

Also, implementing trumps without *Follow the Suit* 5.4 may lead to unbalanced gameplay, as the trump's power can be lost without it.

Additional special rules that change the ranking within the subset of Trump cards might be added when implementing the pattern (like in *Jassen* where the Trump Jack wins all tricks).

Relations:

Instantiates The Trick 5.4, The Hand 5.5, Center Pile 5.5

Modulates Suits and Ranks 5.6, Follow the Suit 5.4

Potentially conflicts with Wild Card 5.6

The Fight

Alternative Names:

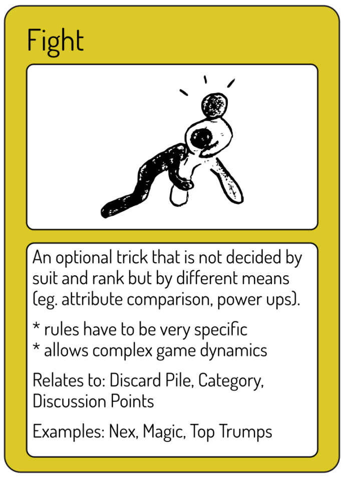
The Fight, The Battle, The Combat

Description:

Some card games use characters, or creatures, an example being *Nex*. Those cards normally have a picture showing the creature and list all of its abilities (the card's attributes). The Battle can be seen as a mini game embedded in a bigger game. Comparing those attributes, decides which card sent into battle survived. The card that loses a battle is discarded, but there are plenty of variants of "card death". A lightweight variant of The Fight can be found in *Top Trumps*, where players compare one attribute and, according to the ranking system, the higher or lower card wins. All sorts of top trump games exist, some with educational value, others purely for fun.

More complex battle mechanics, and thus more complex interaction sequences, can be found in *Magic: The Gathering* or similar collectible card games.

The Fight and *The Trick* seem similar. In both cases, attributes or suits/ranks are compared, and a winner is defined. The patterns are different because *The Fight* is an optional two player interaction, whereas *The Trick* is played every round and by every player. *The Battle* can have multiple phases, and while *The Battle* is fought, additional interactions are allowed. *The Trick* is an encapsulated event without any further interaction other than playing the cards, defining the winner (or winning team), counting the scores, or executing other actions and moving them to the *Score Pile*.



Fight

An optional trick that is not decided by suit and rank but by different means (eg. attribute comparison, power ups).

* rules have to be very specific
* allows complex game dynamics

Relates to: Discard Pile, Category, Discussion Points

Examples: Nex, Magic, Top Trumps

Figure 5.13: Pattern *The Fight*

Consequences:

Winning a fight is often harder to achieve than winning a trick because tactics can be combined in multiple ways. Compared to *The Trick*, it is more competitive and direct, because it is executed face to face and involves experience and tactical skill.

Games with more than two players force the players not involved in the fight to idle.

Using the pattern:

The designer has to choose how deterministic the battle mechanism works. It is possible to include interactive elements into the fight, like *Discussion Points* 5.3.

When using The Fight as an offensive game mechanic it might be a good idea to balance the game by allowing some defensive tactics.

As The Fight is a mini game on its own, it would be possible to completely change the interaction methodology and visualization.

Relations:

Instantiates Discard Pile 5.5, The Base 5.5 and Category (Type) 5.6.

Modulates Discussion Points 5.3 and Score Pile 5.5.

Potentially conflicts with The Trick 5.4

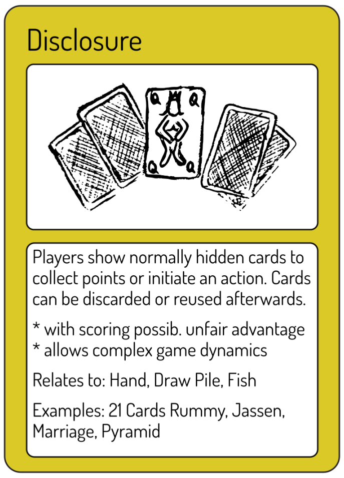
The Disclosure

Alternative Names:

The Marriage, Peek, Forced Disclosure

Description:

Whenever a player gets an advantage by disclosing or revealing a previously concealed or hidden card, The Disclosure can be found. *Jassen* or *Marriage* have a list of defined combinations of cards that reward the player with score points once the cards have been shown to the other players. *Jassen* has additional rules when the disclosure is allowed. *Magic: The Gathering* contains cards allowing a player to take a peek at any opponent's hand cards giving the privilege of private foresight, because other players may not see the card and it is not permanently revealed. Other games (mostly gambling card games) like *Pyramid* use *The Disclosure* as a test for a bluff or a call.



Disclosure

Players show normally hidden cards to collect points or initiate an action. Cards can be discarded or reused afterwards.

* with scoring possib. unfair advantage
* allows complex game dynamics

Relates to: Hand, Draw Pile, Fish

Examples: 21 Cards Rummy, Jassen, Marriage, Pyramid

Figure 5.14: Pattern *The Disclosure*

Consequences:

Once a card has been revealed the player who showed the card either has an advantage or a disadvantage. If the other players now know that there is a Draw Pile the probability of drawing the revealed card(s) decreases.

Using the pattern:

The kind of (dis)advantage that comes with The Disclosure has to be balanced with other game mechanics. If the game uses a Draw Pile the amount of copies per card is relevant. If not some other card exchange like Fish might become necessary.

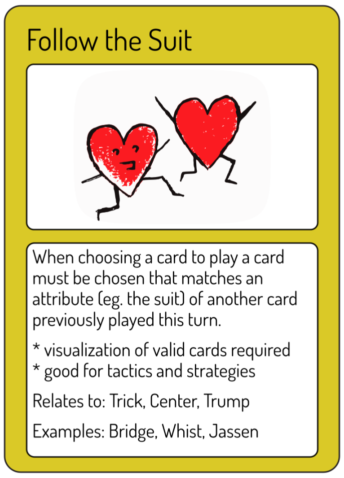
Relations:

Instantiates The Hand 5.5.
Modulates Fish 5.3.
Modulates Draw Pile 5.5.

Follow the Suit

Alternative Names:

Follow the Suit, Following Suit


Follow the Suit

When choosing a card to play a card must be chosen that matches an attribute (eg. the suit) of another card previously played this turn.

* visualization of valid cards required
* good for tactics and strategies

Relates to: Trick, Center, Trump

Examples: Bridge, Whist, Jassen

Figure 5.15: Pattern *Follow the Suit*

Description:

Most trick-taking games force the player to choose from a given suit when playing a trick. The suit is defined by the card played by the leading player. Popular examples are *Whist*, *Bridge*, *Skat*, *Tarock* and many more. The two main differences in games implementing this pattern are:

1. the exceptions: in situations, in which the player can decide to not follow the suit, although being obliged to
2. the consequences: the alternatives the player is allowed to chose from if not being able to follow the suit (e.g. because having no cards of the forced suit)

Games like *President* allow the player to skip the round, whereas games like *Whist* force the player to play any card if following suit is impossible. Interestingly, most games rely on trust in enforcing this pattern. *Jassen* allows the player who defined the Trump 5.4 to withhold the highest card (Bauer/Jack) as an exception to the Follow the Suit pattern.

Consequences:

Games implementing this pattern must have:

1. a list of rules that specify how to follow the suit
2. how the suit to follow is defined
3. what exceptions exist to the rules and when they are legal

Using the pattern:

Relying on trust in the players the designer has to accept that once a player breaks this rule the game enters an undefined (inconsistent) state. Thus, it is very important to decide which methods are used to control the correct execution of the pattern. Possible methods are:

1. interrupt The Move 4.2 and allow the player to choose only allowed cards reducing the dimension of Freedom of Interaction 2.4

2. signal the other players that a player played an illegal card
3. no interference at all

Relations:

Instantiates The Trick 5.4, Center Pile 5.5.
Modulates The Trump 5.4.

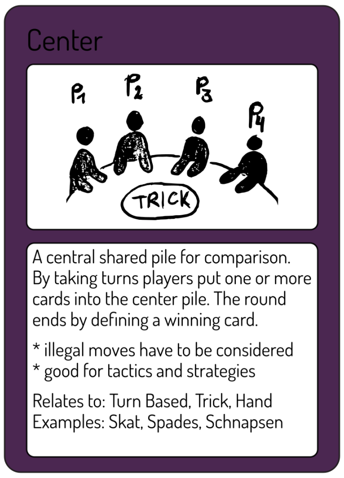
5.5 Pile Patterns

A pile can exist temporarily or permanently, with a fixed or dynamic limit of amount of cards, be concealed or visible, ordered or not and can be private or shared. Piles have so many characteristics and thus different implementations that I was only able to list a small collection of popular patterns.

The Center

Alternative Names:

The Center



A central shared pile for comparison. By taking turns players put one or more cards into the center pile. The round ends by defining a winning card.

* illegal moves have to be considered
* good for tactics and strategies

Relates to: Turn Based, Trick, Hand
Examples: Skat, Spades, Schnapsen

Figure 5.16: Pattern *The Center*

Description:

This is the pile where players place the cards they want to compare before they or the rules decide what happens next. The Center is a shared pile. The combination of the patterns *Turn Based* 5.2, *The Trick* 5.4 and *The Center* are very popular and can be found in *Skat*, *Schnopsn*, *Spades*, *Jassen* and many more. Every player interacts once with The Center. It is forbidden to remove or touch cards from the center until the end of the round. Theoretically, there are many possibilities to change the selection of cards that end up in the center pile, but in most games I researched, players put one or more cards into the pile face up by taking turns until everybody had one try.

In digital card games the move from *The Hand* 5.5 to *The Center* is often just a click.

The Center is easily mixed up with *The Draw Pile* as in *Go Fish*. The difference is, that *Go Fish* does not use The Center as a place for comparison.

Consequences:

Having a center pile simplifies developing winning strategies as everyone interacts with it and one can learn from the others just by watching and memorizing what cards the opponents played.

All cards in the center pile should be visible to all involved players. This might be a problem when there are a lot of players and only a small device screen.

Using the pattern:

The center pile is a temporary pile only. It is the place where a Trick is taken or a Fight is played out.

Every player follows a strategy but using this pattern the tactics are played out in the open - on the center pile.

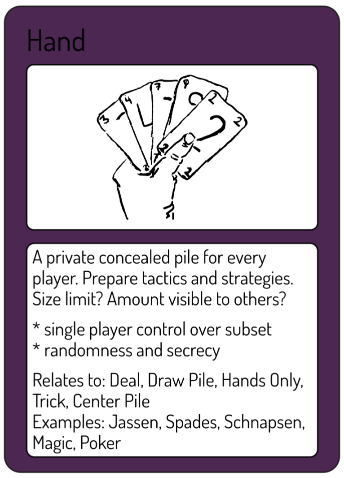
Relations:

Instantiates The Trick 5.4, Turn Based 5.2
Modulates The Hand 5.5, The Score Pile 5.5

The Hand

Alternative Names:

The Hand


Hand

A private concealed pile for every player. Prepare tactics and strategies. Size limit? Amount visible to others?

* single player control over subset
* randomness and secrecy

Relates to: Deal, Draw Pile, Hands Only, Trick, Center Pile
Examples: Jassen, Spades, Schnapsen, Magic, Poker

Figure 5.17: Pattern *The Hand*

Description:

The Hand represents all the cards dealt to a player before the game starts. In closed hand games all those cards are kept concealed and private to the player. The Hand holds the cards that can be used in future Actions and Moves 4.2. It can be considered to be a temporary pile a player uses to plan his tactics and strategy.

In my research I was not able to find a game that uses an open hand. Normally games are played with open hand to teach the rules and consequences of decisions.

Card games without concealed cards and a so called "Hand" do exist (*Open-Face Chinese Poker*) but the Hand in these games lack the feature of being a temporary place for cards that are later used in moves and actions.

Every trick-taking game uses the concealed hand pattern. Card games without The Hand pattern are difficult to identify as a card game.

Consequences:

The concealed hand pattern brings randomness and secrecy to gameplay. A player can secretly prepare a sequence of Actions and Moves and find better strategies or tactics.

Randomness comes in two fashions. Firstly a player does not know which card will be the next to be added to The Hand. Secondly a player does not know which card another player will play next.

Using the pattern:

When players have a secret pile of cards the designer has to decide:

1. how the cards come into the players hand - in the beginning by dealing or drawing, during the game by drawing and other moves
2. how the cards leave the players hand - e.g. if a rule or pattern limits what cards may be used like Follow The Suite 5.4
3. how many cards a player is allowed to hold
4. what happens if a player has no cards left
5. how the hand is different to other concealed piles the game might use

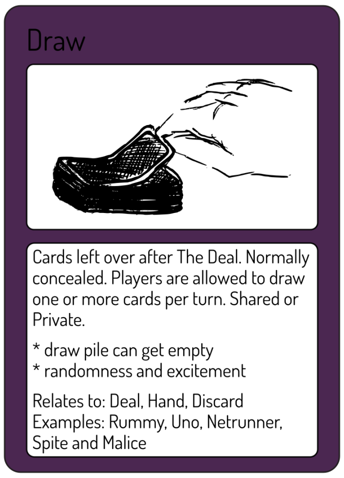
Relations:

Instantiates Deal 5.3.
Modulates Draw Pile 5.5, Hands Only 5.10, The Trick 5.4, The Center 5.5.

The Draw Pile

Alternative Names:

The Draw Pile



Draw

Cards left over after The Deal. Normally concealed. Players are allowed to draw one or more cards per turn. Shared or Private.

* draw pile can get empty
* randomness and excitement

Relates to: Deal, Hand, Discard
Examples: Rummy, Uno, Netrunner, Spite and Malice

Figure 5.18: Pattern *The Draw Pile*

Description:

Card games that distribute the cards to every player have three possibilities concerning card distribution:

1. deal all cards and cap the number of players (static amount of cards)

2. deal a certain amount of cards to each player and remove the rest from the game

3. deal a certain amount of cards to each player and put the rest on a draw pile

The third category of games require a Draw Pile.

Games like *Rummy* have a constant drawing amount: each player draws and discards one card. *Uno* has a variable drawing phase: if no special event was triggered in the turn before the current player, no cards must be drawn. But the player can be forced to draw more than one card by using a special +2-card. In games with a limited maximum amount of cards per hand, like Android Netrunner, the players can be allowed to draw cards until the hand is "full".

The Draw Pile can be a shared pile, like in *Rummy* or *Uno*, or every player might have his own pile, usually in games using *The Deck* 5.8 like *Magic: The Gathering*.

To bring randomness to the gameplay the Draw Pile is normally concealed.

Consequences:

Having a Draw Pile removes a lot of cards from the active gameplay. Of course they are added to the game by drawing new cards, but that takes time in form of rounds which changes the strategy and tactics players have to learn.

Using the pattern:

The designer has to decide what happens when the Draw pile is empty and no card can be drawn anymore. *Uno* shuffles the Discard Pile and reuses it as new Draw Pile. In *Android Netrunner*, the player that cannot draw new cards from the Draw Pile loses the game instantly.

If drawing from the Draw Pile is a forced move in every player's turn (*Rummy*), it can be automated so that players do not forget it.

If the amount of cards a player holds must keep constant (*Spite and Malice* forces the player to hold 5 cards) the amount of cards drawn will vary.

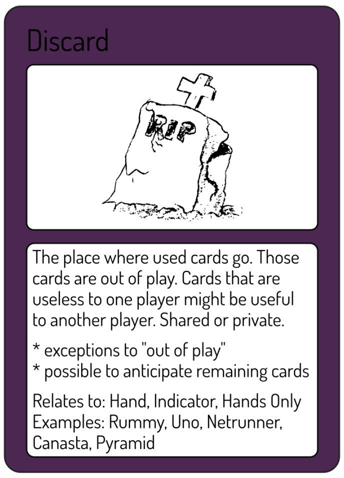
Relations:

Instantiates The Hand 5.5, Deal 5.3
Modulates The Discard Pile 5.5.

The Discard Pile

Alternative Names:

The Discard Pile, The Graveyard



Discard

The place where used cards go. Those cards are out of play. Cards that are useless to one player might be useful to another player. Shared or private.

* exceptions to "out of play"
* possible to anticipate remaining cards

Relates to: Hand, Indicator, Hands Only
Examples: Rummy, Uno, Netrunner, Canasta, Pyramid

Figure 5.19: Pattern *The Discard Pile*

Description:

"Used" cards are moved to the discard pile. *Rummy* and *Uno* use the same pile for all players. In *Magic: The Gathering*, *Dominion* and *Android Netrunner* every player has an own discard pile.

Cards on the discard pile are normally considered "out of play". Those cards cannot be used again. Games differ significantly as to when a card counts as "used" but most games I looked at count a card as used when a defined sequence of actions and moves involving this card is finished.

Rummy, *Canasta* or *Magic: The Gathering* allow retrieving cards from the discard pile.

Pyramid allows to store the top card from the discard pile for later use.

Consequences:

The Discard Pile allows the players to count how many and which cards remain in play and to guess their owners if the game has a static amount of cards in it.

Using the pattern:

When using this pattern the following aspects might be relevant:

1. when cards may be drawn from the discard pile
2. if the discard pile is shared among all players
3. if the order of cards in the discard pile is relevant
4. and when it is allowed to look at all cards in the discard pile

It is possible to re-use the discard pile by shuffling it as the new draw pile (*Uno*).

Relations:

Modulates *Indicator* 5.5, *Hands Only* 5.10 and other Pile Patterns 5.5.

Instantiated by The Draw Pile 5.5

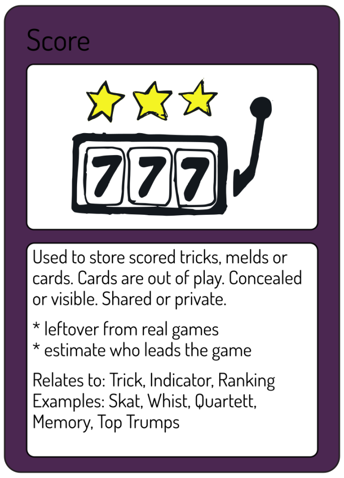
The Score Pile

Alternative Names:

The Score Pile

Description:

Whenever a player wins a trick or scores a card and it has to be moved to a special pile for later summation of all points, this is called the score pile. Almost every trick-taking game uses score piles in order to mark the used cards as “out of play”. After a trick, the cards are moved to the Score Pile face down, so that it is not possible to check what cards have been played. When the game has finished the sum of all card values held in each score pile defines the winner. Besides *Skat*, *Whist* or *Jassen* games like *Quartett*, *Memory* or *Top Trumps* use score piles to just count the amount of scored cards.



Score

Used to store scored tricks, melds or cards. Cards are out of play. Concealed or visible. Shared or private.

* leftover from real games
* estimate who leads the game

Relates to: Trick, Indicator, Ranking
Examples: Skat, Whist, Quartett, Memory, Top Trumps

Figure 5.20: Pattern *The Score Pile*

Consequences:

Score piles are used to keep record and act as a proof. Thus it is more important than other piles. Depending on the winning strategy a score pile can be an advantage or disadvantage for a player or an opponent.

Using the pattern:

It can be allowed or disallowed to look at the cards in a score pile. It is possible to force players to keep the amount of points scored by their opponents in memory instead of allowing them to peek at the cards.

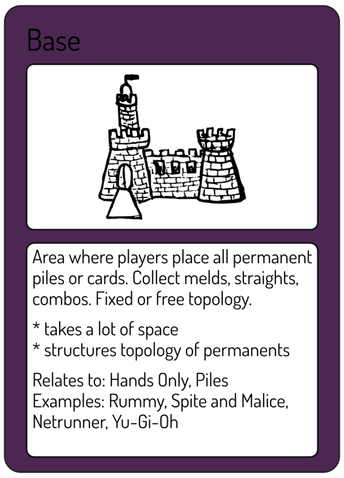
Relations:

Modulates The Trick 5.4, Ranking patterns 5.9

The Base

Alternative Names:

The Main Area, The Base, Headquarters


Base

Area where players place all permanent piles or cards. Collect melds, straights, combos. Fixed or free topology.

* takes a lot of space
* structures topology of permanents

Relates to: Hands Only, Piles
Examples: Rummy, Spite and Malice, Netrunner, Yu-Gi-Oh

Figure 5.21: Pattern *The Base Tableau*

Description:

This is the area or tableau where the player places all permanent piles or cards. They can be concealed or visible to the opponents.

In *Rummy*, a player has to collect melds (combinations of cards) in The Base to score points. *Spite and Malice* limits the amount of collection piles a player can use.

Strategic card games like *Android Netrunner*, *Yu-Gi-Oh* and *Magic: The Gathering* use fine grained pile topologies (servers, creatures or other category cards are collected there). There exist playmats outlining the zones, areas and different piles the player is allowed to use.

Consequences:

When the players have to physically divide their preparation space it is necessary to have a declared area where piles and other permanents go. In *Magic: The Gathering* The Base is used to store cards for later use - to prepare tactics and winning strategies. Less complex games like *Rummy* use it for counting.

The place The Base takes can become a problem when the player is allowed to create as many piles as desired or required.

Using the pattern:

In my research only Magic and Netrunner had a dynamic amount of piles.

Ligretto has no dynamic amount of piles, because the amount of “1”-cards is limited (every player has four).

Each pile requires additional interactions and additional space. The game complexity can be reduced if the allowed piles in The Base are predetermined.

Relations:

Conflicts with Hands Only 5.10

Indicator

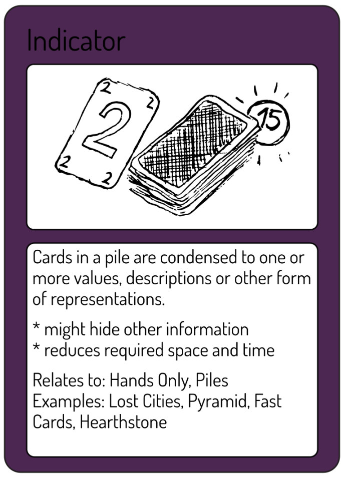
Alternative Names:

Indicator

Description:

This pattern is hard to find in the real world, but is commonly used in digital card games. The cards in a pile are condensed to one or more values, descriptions or other form of representations. Counting cards in a pile or looking something up is a recurring action in card games. Digital card games have the opportunity to do automatic counting or aggregation.

Lost Cities uses a pile indicator in the top of an open pile to sum the points per column. *Pyramid* and *Fast Cards* uses a pile indicator to show how many cards remain in the drawing pile. The 3D-visualization in *Hearthstone* allows the game to show the amount of cards remaining with the thickness of the pile. This gives the user experience aesthetics a very realistic touch. It feels like the thickness is something our brain is processing fast in order to learn how much cards are remaining. The classic approach is to show the exact number.



Indicator

Cards in a pile are condensed to one or more values, descriptions or other form of representations.

* might hide other information
* reduces required space and time

Relates to: Hands Only, Piles
Examples: Lost Cities, Pyramid, Fast Cards, Hearthstone

Figure 5.22: Pattern *Indicator*

Consequences:

Indicators are used as a supportive means of gameplay. When counting does not take time the players can act faster and focus on tactics and strategy. Specially games with a lot of piles use this pattern.

Using the pattern:

The Indicator can use different visualizations. If the indicating value is a number it is necessary that the unit gets obvious. In the games I researched for this work I found that the unit is not explained, it is expected that the player understands that its the amount of cards or the amount of points by guessing it. People I talked to when researching different card games told me that part of playing card games is to estimate and trust your gut feeling, something that might get lost when overusing the indicator.

Relations:

Instantiates Pile Patterns 5.5

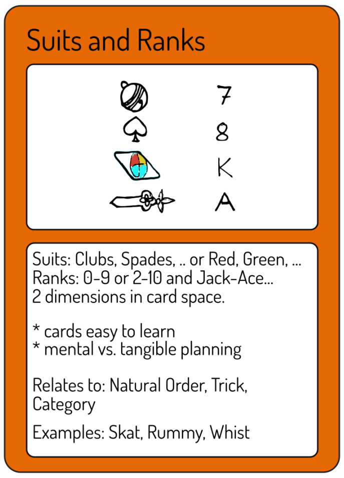
5.6 Card Patterns

Cards as game objects have attributes in multiple dimensions. Most low complexity games have the same attribute dimensions for all cards, more complex games have more diversity. Cards can also have effects that manipulate the game state directly instead by a move. These patterns describe the structure of attribute dimensions or how cards are used in gameplay.

Suits and Ranks

Alternative Names:

Suits and Ranks



Suits and Ranks

7
8
K
A

Suits: Clubs, Spades, .. or Red, Green, ...
Ranks: 0-9 or 2-10 and Jack-Ace...
2 dimensions in card space.

* cards easy to learn
* mental vs. tangible planning

Relates to: Natural Order, Trick, Category

Examples: Skat, Rummy, Whist

Figure 5.23: Pattern *Suits and Ranks*

Description:

The Suits and Ranks pattern can be understood as the predecessor to the Category pattern. It is a “standard” Category pattern including the “standard” ranking pattern *Natural Order* 5.9. A suit is one specific category or type, the ranks define the order within the set of cards of one suit. Classic card games use the so called “Standard 52-card deck”. Different cultural regions use different faces: French, German, Swiss-German, Italian and Spanish.

Games like *Rummy*, *Whist* or *Bridge* use the “French pattern”, have the suits Spades, Hearts, Diamonds and Clubs and rank from 2 to 10 and from Jack to Ace.

Skat uses the “German pattern” having Hearts, Bells, Acorns and Leaves and rank from 7 to 10 and from Jack to Ace.

Classic card games have the commonality that they use 4 suits but different ranking systems and sometimes special cards like a Wild Card 5.6.

Spite and Malice uses a standard card deck, but its rules make suits irrelevant, only the rank is relevant.

Uno also implements the Suit and Ranks pattern. It has 4 suits: Yellow, Green, Blue and Red and ranks from 0 to 9 and some cards that are not ranked like “Stop”, “Reverse” or “+2”. Of course Uno would not be Uno without a “Special Suit” that has no color and lets the players switch between colors.

The big difference between the Category pattern and the Suits pattern is that categories and types have a meaning and need a textual description and icons while suits are directly identified by color or symbol.

Consequences:

It is a commonly found pattern, almost all traditionally known card games implement it. It is easier to learn games based on this pattern, because other patterns require new players to learn the cards first. The vast amount of different games based on the standard deck shows that a lot of card games hide complexity in abstract systems (ranking, scoring, etc). Those games are hard to master, because tactics and strategies have to be planned mentally. Games that do not implement this pattern (like *Magic: The Gathering*) have a very dynamic gameplay compared to games with this pattern (*Rummy*, *Whist*, ...). Games that mix this pattern with other patterns (like *Uno*) make the game more tangible and only little mental planning is required.

Using the pattern:

The Suits and Ranks pattern comes with many strings attached. As this pattern has consequences for player interaction (requiring The Trick) and ranking (requiring Natural Order) it it is hard to mix it with other patterns without destroying basic concepts of this pattern.

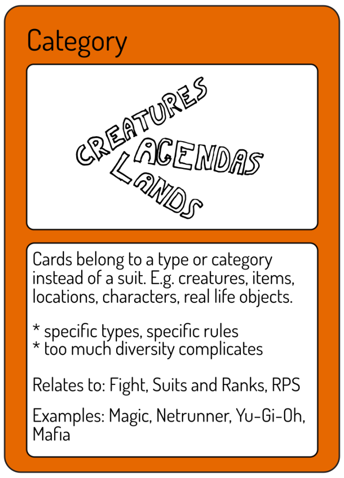
Relations:

Instantiates The Trick 5.4, Natural Order 5.9
Conflicts with Category (Type) 5.6

Category (Type)

Alternative Names:

Card Categories, Card Types


CREATURES
AGENDAS
LANDS

Category

Cards belong to a type or category instead of a suit. E.g. creatures, items, locations, characters, real life objects.

* specific types, specific rules
* too much diversity complicates

Relates to: Fight, Suits and Ranks, RPS

Examples: Magic, Netrunner, Yu-Gi-Oh, Mafia

Figure 5.24: Pattern *Category*

Description:

Most collectible card games like *Magic: The Gathering* or *Yu-Gi-Oh* group their cards into types. They use creatures, characters, locations, items, whatever real life subject or object a card can represent. The observed categories/types suggest tactics and strategies. A creature can be used in a fight, a server can be advanced, an interrupt can be used although its not your turn.

Categories are not just a distinction by attribute groups or single attribute like suits or colors. If a player draws an "Agenda" in *Android Netrunner* the general possibilities (e.g. to win the game) are known instantly without knowing the specifics. *Mafia* (aka *Werewolf*) uses character cards that are held concealed and use no other cards having suits or ranks.

Consequences:

Games implementing this pattern tend to be collectible card games, because they allow to create new cards. Having this complexity of a huge amount of available cards to choose from, an inexperienced player has blind spots when developing a strategy.

Using the pattern:

Adding categories to a deck makes sense when the game has a lot of cards. Collectible Card Games use this pattern.

The designer can create imbalances within the cards of one category. E.g. there can be creatures supporting a more offensive or defensive gameplay. Thus the pattern is especially strong when combined with Rock-Paper-Scissors (a general game design pattern).

Relations:

Possibly conflicts with Suits and Ranks 5.6

Modulates Rock-Paper-Scissors 5.9, The Fight 5.4

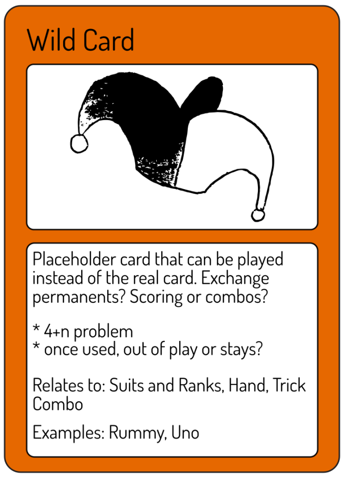
Wild Card (Joker)

Alternative Names:

Wild Card, The Joker

Description:

Some games allow the player to use a "placeholder" card instead of the card required to play a certain move. In *Rummy*, the player has to create combinations of the same cards or "straights" (cards having the same suit, ordered by their rank, no gaps) or "collections" (cards of the same rank, an implementation of *The Combo* 5.6). If a card is missing, e.g. if the player wants to combine three Queens but has only two, the wild card can be used instead.


Wild Card

Placeholder card that can be played instead of the real card. Exchange permanents? Scoring or combos?

* 4+n problem
* once used, out of play or stays?

Relates to: Suits and Ranks, Hand, Trick Combo

Examples: Rummy, Uno

Figure 5.25: Pattern *Wild Card*

Consequences:

If the game uses Suits 5.6 Wild Card is without suit and can be a card on its own or represented by any card like the King or Queen.

The pattern brings additional randomness to the game. It makes gameplay less predictable and its harder for players to develop sustaining tactics and strategies.

If allowed in *Poker: Texas Hold'em* the *Wild Card Poker Paradox* [18] shifts the positions of hands: If a wild card is allowed there are more possibilities to form three of a kind instead of two pair.

Using the pattern:

Some games have an explicit Joker-Card, others declare an existing one as the wild card. Once it has been used it makes a difference if the wild card goes out of play or stays in the game. An example can be found in *Rummy* which allows other players to re-use the wild card and thus making all wild cards permanently usable.

Relations:

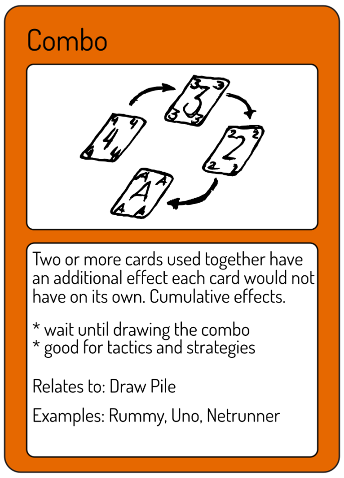
Modulates Combo 5.6

Modulates Suits and Ranks 5.6

The Combo

Alternative Names:

The Combo, Combination, Stacking


Combo

Two or more cards used together have an additional effect each card would not have on its own. Cumulative effects.

* wait until drawing the combo
* good for tactics and strategies

Relates to: Draw Pile

Examples: Rummy, Uno, Netrunner

Figure 5.26: Pattern *The Combo*

Description:

Combos are cards that have a cumulative effect. *Jassen* does not have a deck to draw from but players can show card combinations to score points. *Uno* allows players to stack up +2 cards. *Magic: The Gathering* uses a whole category of cards called "Enchantments" which change the attributes of the enchanted card. In *Android Netrunner* the corporation can boost servers by using "Upgrades". To score points in *Rummy* you have to build "Melds" like "Streets".

Combos are a strong pattern and can be found in many games.

In *Go Fish* the cards belonging to each other have to be found by comparing their rank.

Almost every vying game like *Poker* uses combos for scoring.

Consequences:

Implementing The Combo can have different goals. One is to build a combination by collecting cards from different sources like The Draw Pile or other player's hands. Another is to wait until the combo is dealt to you.

Using the pattern:

A player can use different actions or moves to collect or find the cards required to build a combo. Some games use balancing mechanics against strong combos like the "Stop"-card (Skip a Turn) in *Uno*.

Combos can be used to have different effects. Some combos only have an effect for the scoring system like melds in *Jassen* or a "hand" in *Poker*.

Relations:

Modulates The Draw Pile 5.5, Wild Card 5.6

5.7 Marking Patterns

Digital card games have the problem of device interface size and object placement. That is why many games choose to use visual and/or structural means of decision making support and information aggregation. Marking patterns are patterns that describe how game objects can be emphasized or upgraded.

Select

Alternative Names:

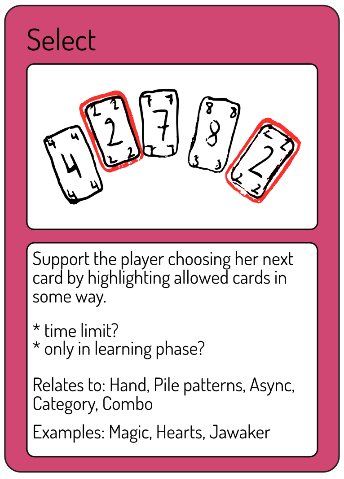
Select, Activate, Tap, Rotate

Description:

Selecting cards for a Move 3.12 or an action is the base mechanic of card games. The selection can be final (once the card is selected the move or action follows directly) or a preparation: the card is only shown in detail or highlighted but taking the actual move or action requires an additional interaction. Often this interaction is automated and the game is automatically selecting a card for the player. Games that allow you to use a card without highlighting that card in any way (with color, marked by a Counter 5.7 are not implementing the Select pattern. The Select pattern is required when multiple alternatives are presented to the player.

Games in *Jawker* and *Spades* highlight the cards you can move to the center pile. Swiping the card towards the center moves it there. In those games selection and move happens in one interaction.

In *Hearts+*, *Order and Chaos Duels* and *Magic Duels* a card can be selected before a move or an action is forced. This is done by bringing the card into focus, rotating it relatively to the rest (called “tap”) or highlighting it.


Select

Support the player choosing her next card by highlighting allowed cards in some way.

* time limit?
* only in learning phase?

Relates to: Hand, Pile patterns, Async, Category, Combo

Examples: Magic, Hearts, Jawaker

Figure 5.27: Pattern *Select*

Consequences:

Games with a lot of possible tactics use this pattern so that a player can calculate the risks or balance the strategy before taking an action. Selection is one way to help solving decision problems in games. As such, it is important to note that Selection takes time and might be problematic in asynchronous gameplay.

Using the pattern:

If the game implements the Category pattern, cards normally have a lot of attributes. If the game is a collectible card game, it is impossible to know all cards. Using the selection pattern in such situations may be crucial to make the game playable in the first place.

A selection always happens in the context of a pile. If the card is not visually separated from the pile during the selection, highlighting can be a visual problem. Designers use different approaches using brightness, borders or transformations (of size, relative position, rotation...).

Relations:

Instantiates Hand 5.5 and pile patterns 5.5

Modulates Category (Type) 5.6 and Async 5.10

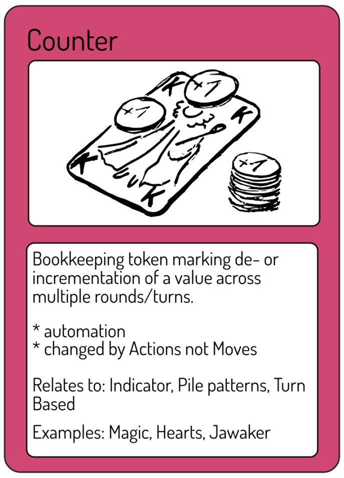
Counter

Alternative Names:

Counter, Token, Marker

Description:

Counters are bookkeeping tokens for incrementing/decrementing a value across multiple turns. Like the Indicator 5.5 it is used to ease focusing on the gameplay rather than memorizing the game state. Typically chips, small stones or tiles are used to count a certain game state value over time. *Magic: The Gathering* uses them to count how much a creatures strength has been enhanced. In *Spades* a counter is used to track one's bid and must not be changed until a round has finished. *Android Netrunner* uses counters to keep track of one's credits as the game has its own currency and economy.



Counter

Bookkeeping token marking de- or incrementation of a value across multiple rounds/turns.

* automation
* changed by Actions not Moves

Relates to: Indicator, Pile patterns, Turn Based

Examples: Magic, Hearts, Jawaker

Figure 5.28: Pattern *Counter*

Consequences:

Counters have the advantage that the player does not have to memorize all the game states values. Concealing this information by disallowing physical tokens on the other hand benefits the player with the best memory.

Using the pattern:

Counters are a great way to bring diversity into an otherwise static game. Games that use a currency and an economy must implement *Counter*.

In my experiments with players I had to realize that the handling of counters is problematic because they can get lost or be the reason for dispute.

The digital representation of counters I found were always plain numbers, the player never had to count the tokens themselves.

Relations:

Instantiates Turn Based 5.2

Instantiated by Indicator 5.5

Modulates pile patterns 5.5

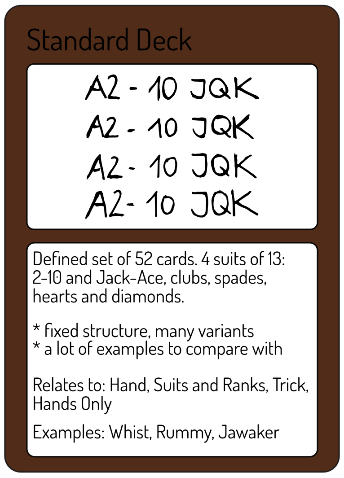
5.8 Deck Patterns

The cards a single game is played with are a subset of all the cards the game includes. There are different ways to define which cards a player may use in a game. The following patterns represent the most common ways to do so.

The Standard Deck

Alternative Names:

Standard Deck, 52-Card-Deck



Standard Deck

A2 - 10 JQK
A2 - 10 JQK
A2 - 10 JQK
A2 - 10 JQK

Defined set of 52 cards. 4 suits of 13: 2-10 and Jack-Ace, clubs, spades, hearts and diamonds.

* fixed structure, many variants
* a lot of examples to compare with

Relates to: Hand, Suits and Ranks, Trick, Hands Only

Examples: Whist, Rummy, Jawaker

Figure 5.29: Pattern *The Standard Deck*

Description:

Every card game having a defined set of cards - meaning that players cannot create their own cards - use the Deck pattern and a lot of them use the Standard Deck. The differences between collectible card games like *Hearthstone* and a classic card game like *Whist* is that Hearthstone's cards are unique for the game whereas Whist cards are the same as for *Rummy*. Classic card games like *Rummy*, *Whist* or *Skat* use the Standard Deck with various faces. Although the design are different in different regions (Spanish, German, French, ...) the rules are the same.

Consequences:

Games that use standard decks allow many game variants. The complexity of games built upon the Standard Deck is limited and still it is where the most popular games come from.

It nowadays normally comes without rules as the rules for games with it can be looked up in books or on the internet.

Digital games using the Standard Deck do not use a standardized visualization. Almost every game I found has slightly different visualizations of it. Be it in color or pictures, differences apply, possibly because of copyright issues.

Using the pattern:

Every card game uses some sort of deck. What differentiates the games then?

1. the structure of the deck (see Card Patterns 5.6)
2. the way the deck is distributed (see Pile Patterns 5.5)
3. and how the deck is used in the game (see Interaction 3.1)

Following some statements that explain how the standard deck limits the above differentiation:

1. has a fixed structure
2. is evenly distributed to each player
3. each player has a hand
4. remaining cards are held in a draw pile or all cards are dealt
5. the game is decided by taking tricks (loosing all cards) or building piles (collecting cards)
6. a scoring mechanism makes each game comparable to another

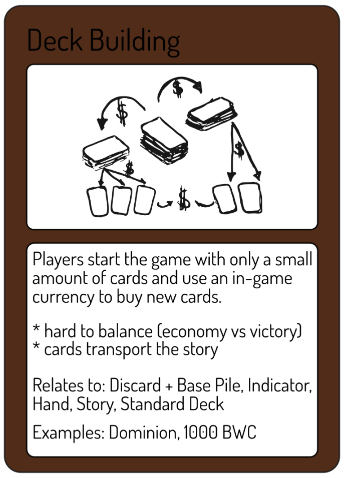
Relations:

Instantiates Natural Order 5.9, Hand 5.5, Suits and Ranks 5.6, Trick 5.4
Modulates Personal Deck 5.8, Deck Building 5.8
Conflicts with 5.10

Deck Building

Alternative Names:

Bulid your own Deck, Extend while you play, Deck-building



Deck Building

Players start the game with only a small amount of cards and use an in-game currency to buy new cards.

* hard to balance (economy vs victory)
* cards transport the story

Relates to: Discard + Base Pile, Indicator, Hand, Story, Standard Deck

Examples: Dominion, 1000 BWC

Figure 5.30: Pattern *Deck Building*

Description:

In most card games players share a finite set of cards they can draw, choose and play from. Games like *Dominion* (the first deck building game), introduce a private deck and a shared stack. Once a player's private deck runs out of cards, the discard pile is shuffled and used as private deck once more until the shared stack is empty or a winning condition is reached. Choosing wisely which cards a player buys, the odds to win are influenced. Popular successors of *Dominion* are *Legendary: Alien Encounters*, *Mage Knight* or *A Few Acres of Snow*.

The game *1000 Blank White Cards* takes a different approach to building your own deck. Every player has to create a new card each turn. What the card does is totally undefined and only restricted by a small set of (behavioural) rules.

Consequences:

The mechanic of a shared stack of cards where players buy cards from requires an in-game currency and thus has the same consequences any economy based game has. If the game does not end when all cards in the stack are sold, but because a winning condition is reached the main winning strategy is to be fast and the designer has to be careful about designing cards that speed up the gameplay. Balancing the card effects is one of the hardest tasks in Deck Building games and can only be predicted by very complex models or by playing the game a lot.

Using the pattern:

One has to realize that balancing a Deck Building game is a time consuming task. Every new type of card attribute introduces a new complexity level. In order to be perceived as a fun game it is easier to develop the game in iterations.

Relations:

Instantiates The Discard Pile 5.5

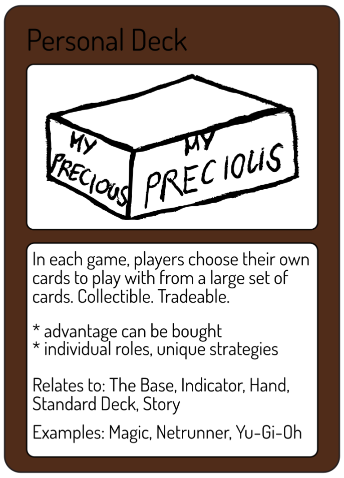
Personal Deck

Alternative Names:

Personal Deck

Description:

Collectible card games strongly promote this pattern. In *Hearthstone*, *Magic: The Gathering*, *Yu-Gi-Oh* and others, the players get stronger the better cards exist in their collections. Over time, a player collects a stack of cards from which the cards to play with are chosen. It is not allowed or possible to play with all cards a player owns. *Magic: The Gathering* has over 15.000 (in 2015) cards to collect. The cards a player chooses to battle others is called the "Personal Deck". *Android Netrunner* has a free online implementation where personal decks of all existing cards can be built [55].



Personal Deck

In each game, players choose their own cards to play with from a large set of cards. Collectible. Tradeable.

* advantage can be bought
* individual roles, unique strategies

Relates to: The Base, Indicator, Hand, Standard Deck, Story

Examples: Magic, Netrunner, Yu-Gi-Oh

Figure 5.31: Pattern *Personal Deck*

Consequences:

Collectible card games are not that popular as most of them cost a lot of money. It is even difficult to find players that play at the same level and with the same intent. Having a lot of money optimizes a player's odds to win (more, better cards to choose from).

Most players I interviewed are however fascinated by these games and the worlds they are situated in. The complexity allows players to feel special or give the impression they found a unique combination of tactics and strategy.

Using the pattern:

The digital domain is the optimal environment for collectible card games as it possibly removes the money-factor. A lot of collectible card games have been developed but only some of them reached a critical amount of players that play the full spectrum of the game.

Designing a world around a game that uses the Personal Deck pattern helps

Relations:

Instantiates The Hand 5.5, The Base 5.5

Modulates 5.5

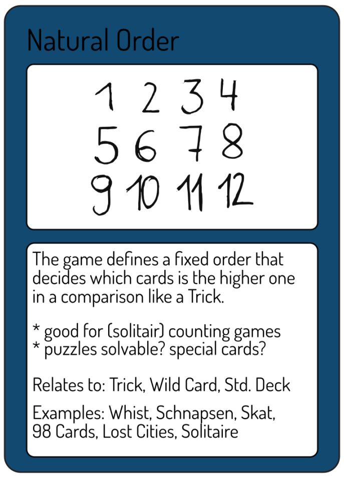
5.9 Ranking Patterns

All games implementing trick patterns 5.4 require some order in which the card attributes are compared.

Natural Order

Alternative Names:

Natural Order, Numerical Order



Natural Order

1 2 3 4
5 6 7 8
9 10 11 12

The game defines a fixed order that decides which cards is the higher one in a comparison like a Trick.

* good for (solitair) counting games
* puzzles solvable? special cards?

Relates to: Trick, Wild Card, Std. Deck

Examples: Whist, Schnapsen, Skat, 98 Cards, Lost Cities, Solitaire

Figure 5.32: Pattern *Natural Order*

Description:

In the games researched there were two situations where natural order was relevant: in tricks and when ordering (stacking) cards.

Games like *Whist*, *Schnapsen*, *Skat* or *Rummy* define which card wins a trick based on their natural order. The standard 52-cards deck uses numerical order from 2 to 10 and then defines Jack, Queen, King and Ace. In *Spite and Malice* the Standard Deck is also used, but the game is played “ace low”, meaning that the King is the highest card and the Ace is the lowest (lower than 2).

Ligretto, *98 Cards* and *Lost Cities* implement the *Natural Order* pattern, as all three games have number cards and the cards can only be stacked if the natural order is preserved.

Using numbers on the cards does not mean *Natural Order* is used. *Uno* does not implement *Trick* and no ranking is required.

Consequences:

If the game has no other mechanics than tricks based on natural order, the cards used are interchangeable as it is only the mathematical relationship that matters.

Mathematical order does not have to be thought only in +1/-1 terms. *98 Cards* allows a lower card to be put on a rising pile if the card is exactly 10 points less. In *Mod 13* the next valid card is identified by an arithmetic operation.

Some sorting puzzles in *Solitaire* are not solvable if the cards are purely randomized and no algorithm checks if the puzzle is solvable at all.

Using the pattern:

Games like *Speed* and *98 Cards* use additional mechanics.

As explained in the consequences a designer has to think about dead-lock situations where a puzzle cannot be solved.

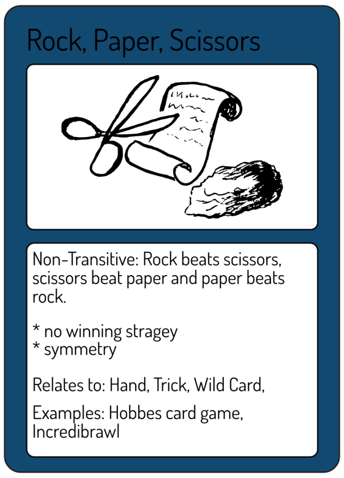
Relations:

Instantiated by Standard Deck 5.8
    Modulated by Trick Patterns 5.4
    Modulates Wild Card 5.6
    Conflicts with Null Ranking 5.9

Rock, Paper, Scissors

Alternative Names:

Rock-Paper-Scissors, Zero-Sum Ranking



Rock, Paper, Scissors

Non-Transitive: Rock beats scissors, scissors beat paper and paper beats rock.

* no winning stragey
* symmetry

Relates to: Hand, Trick, Wild Card,

Examples: Hobbes card game, Incredibrawl

Figure 5.33: Pattern *Rock Paper Scissors*

Description:

Found in the children's game with the same name the concept of non-transitivity has one important effect: In its basic form there is no winning strategy. The player has to somehow figure out what choice is the best at each moment [10]. The game *Incredibrawl* introduces "power types" for its characters and in a "scrap" (taking a trick) the power types defy each other using this pattern.

The pattern is also used in educational card games like the *Hobbes card game* [4].

Consequences:

If a game needs some decision that involves two players the pattern can be used. Rock-Paper-Scissors is purely random and introduces symmetry between actions or tactics. In its basic form it is impossible for players to develop tactics or strategies.

Using the pattern:

When used in its basic form, a meta game has to be found that allows the players to gain knowledge over their opponents strategies. Certain effects could be bound to the players choice removing the randomness factor and giving the others a means of prediction.

Relations:

Modulates The Hand (in its concealed form) 5.5, Trick 5.4, The Draw Pile 5.5

Potentially conflicts with Wild Card 5.6 and Natural Order 5.9

Null Ranking

Alternative Names:

Null Ranking

Description:

In games where the value of a card does not matter, they can be equal in their rank because only collecting, combinations or patterns matter for winning a game. This can be found in games like *Memory*, *Old Maid* and in a weaker form in *Quartett*.

Consequences:

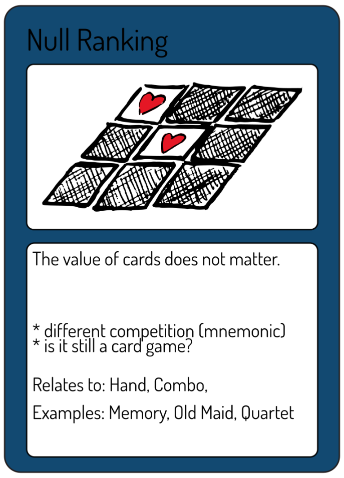
Games implementing this pattern have a very different winning strategy. *Memory* is not about finding the best strategy to crash your opponent, it is more a test of one's mnemonic abilities. Such games are on the edge of being a card game at all because it could be interpreted that *Domino* printed on cards would be a game implementing this pattern.

Using the pattern:

When a deck of cards lacks a method of comparing the value of two cards other means of winning a game have to be found.

Relations:

Potentially conflicts with The Hand 5.5
Instantiates The Combo 5.6
Modulates Deck Building 5.8



Null Ranking

The value of cards does not matter.

* different competition (mnemonic)
* is it still a card game?

Relates to: Hand, Combo,
Examples: Memory, Old Maid, Quartet

Figure 5.34: Pattern *Null Ranking*

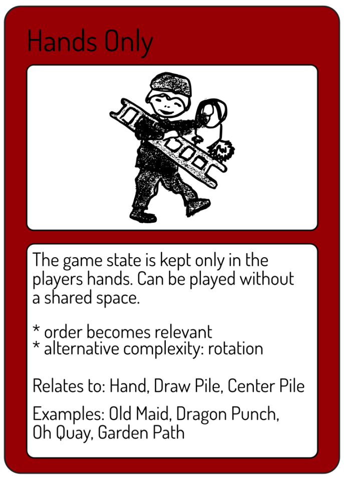
5.10 Meta Patterns

These patterns are more general and not strictly bound to card games but can be observed as a side effect of playing digital card games.

Hands Only

Alternative Names:

Table free, Desk less, Hands only



Hands Only

The game state is kept only in the players hands. Can be played without a shared space.

* order becomes relevant
* alternative complexity: rotation

Relates to: Hand, Draw Pile, Center Pile

Examples: Old Maid, Dragon Punch, Oh Quay, Garden Path

Figure 5.35: Pattern *Hands Only*

Description:

There exist card games like *Old Maid*, *Dragon Punch* or the hardly popular indie games *Oh Quay* and *Garden Path* (more concept than indie) which are played without a shared space. There is no center, draw or score pile or anything that would be important to keep track of. The game state is kept only in the players hands. Players interact by exchanging, discarding or sorting their cards.

It is open for discussion if a game server counts as a shared space or not. To be consistent to the way these patterns have been found and described, a game server which does not introduce a new pile or some other game element, does not count. It does only data synchronization and therefore digital card games can implement the *Hands Only* pattern.

Consequences:

Games which are played without a shared space require the game state to be saved within the players hands. At first it may seem that such games must be most simple. But complexity can be generated in many ways: *Oh Quay* does not allow the players to rearrange their hand like in *Bohnanza*, which brings more complexity to the game because suddenly sorting matters. *Garden Path* introduces the rotation of the cards to be of importance.

Using the pattern:

In printed card games the amount of cards dealt to each player is limited by the players physical ability. Games implementing this pattern normally have a small deck size. Digital games remove this limitation but a designer should be aware that breaking this rule might generate the implicit use of other patterns (e.g. pile patterns) without thinking about the consequences.

Most games that are hands only, are swift and have only a couple of rounds. Hands only really emphasizes ubiquitous card gaming.

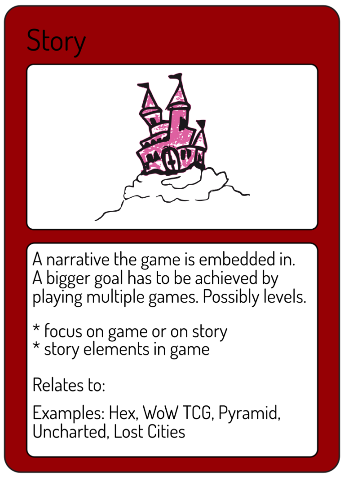
Relations:

Instantiates The Hand 5.5
    Potentially conflicts with pile patterns 5.5

Story

Alternative Names:

Story, Embedding, Intro, Integration



Story

A narrative the game is embedded in. A bigger goal has to be achieved by playing multiple games. Possibly levels.

* focus on game or on story
* story elements in game

Relates to:

Examples: Hex, WoW TCG, Pyramid, Uncharted, Lost Cities

Figure 5.36: Pattern *Story*

Description:

Some digital card games do not directly jump into gameplay but have a welcome screen. This interface is sometimes merged with a narrative. Other games also allow the configuration of options from the welcome screen.

Card games per se do not require or have levels, but digital card games have to put a single game into a bigger meta context. They give an introduction to a story or setting, generally embedding the card game in a virtual narrative surrounding. *WoW TCG* or *Hex* even embed the card game in a MMORPG world.

In *Lost Cities* it is “a journey to undiscovered countries”.

In *Pyramid Solitaire Saga* Helena and Kingsley travel through time.

The environment the actual game is embedded in can influence the odds if a state is tracked over multiple games. In *Uncharted: Fight for Fortune* where the story is in fact a whole world, collecting items in the original game adds special cards in the card game.

Consequences:

Digital games seldom include rules printed on paper, so the player has to learn the game on the same interface later played on. The more complex a game gets the more complicated it is to explain it using only such a small device as a mobile phone for example.

Digital devices offer much more options in story telling than paper only.

Using the pattern:

The story, the configuration and the explanation of the environment of the game should be done in a consistent way.

The story has to be recognizable in the game.

Pyramid Solitaire Saga has a nice story but the story has no influence on the game which could have been done better.

Relations:

No direct relations.

Asynchronous Gameplay

Alternative Names:

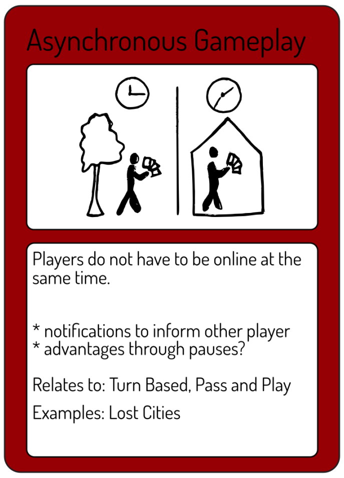
Asynchronous Gameplay, Playing with pauses

Description:

Asynchronous gameplay is a feature of games that do not require the players to interact with it simultaneously. Example: Alice's turn is finished and her actions are submitted to a game server. Alice now has to wait for Bob's response and the game switched into a waiting state. Alice can not change anything until Bob sent his response. Whenever Bob starts the game again he receives Alice's actions and takes his turn. Once finished his actions are sent, and now he has to wait for Alice's response and so on.

Digital card games seem to be a perfect fit for asynchronous gameplay because most card games are *Turn Based* 5.2 like *Lost Cities*.

Albeit using turns some card games require synchronous gameplay because players would loose advantage. *Event Based* 5.2 Games like *Speed* require all players to be alert simultaneously. They have to wait for an opportunity to lose cards.



Players do not have to be online at the same time.

* notifications to inform other player
* advantages through pauses?

Relates to: Turn Based, Pass and Play

Examples: Lost Cities

Figure 5.37: Pattern *Async*

But turn based games like *Chess* or *Whist* (because players switch turns) and asynchronous gameplay are two different things. The pattern described here means that players neither have to be in the same location nor have to play at the same time.

Consequences:

If the games winning strategy depends on simultaneous player interaction effects like in *Ligretto* or *Poker* the game cannot be played asynchronously.

Implementing this pattern multiplayer games require a game server to save the game state until the game is continued. The game state then has to be synchronized between players.

Also a notification system is required so that players are informed when they can make the next move.

Using the pattern:

The designer has to decide what will be asynchronous. Asynchronous gameplay forces players to wait. It has to be checked if a player can gain an advantage from that.

Sometimes it might be better to implement *Pass and Play* 5.3 like in *Uncharted: Fight for Fortune*.

Relations:

Instantiates Turn Based 5.2
Conflicts with Event Based 5.2, Pass and Play 5.3

5.11 Examples

In this section I try to design a game in two ways by defining some mutual parameters and then using different patterns. This shows that the resulting games are very different. These examples should also show how a game can be designed in an iterative manner using design patterns. There is no implementation of these examples, as that should only help with understanding how to use design patterns.

Another example of how to create a game based on a pattern-like approach has been done by *David Sirlin* [70] for the collectible online card game *Kongai*.

Parameters

1. it should be a collectible card game. A player starts with a set, but gets better possessing more cards.
2. players challenge each others
3. there should be heroes
4. heroes should be equipped with items
5. heroes fight each other until one party wins

Social Media Online Game

For this environment I chose the following patterns:

1. **Category 5.6:** four fractions
2. **Async 5.10:** this game does not require synchronous gameplay, however players have to wait until the opponent chose with a party to answer the challenge with
3. **Single Turn 5.2:** this game does not require synchronous gameplay
4. **Indicator 5.5:** some indicators need to be defined for the opponent's decks, so players can more easily determine who they are willing to battle. The indicators show which fractions they use and which level-value the accumulated items have.
5. **Personal Deck 5.8:** each player designs a deck from the cards collected.

The Cards

This implementation offers 80 cards to collect. As the game is played online, in the environment of a social media platform, cards could be distributed using "achievements" or other means the platform offers. When a user starts to play this game, the player is given 3 random heroes and 12 random items of the associated fractions.

1. There are 12 hero-cards

2. Each hero has a name, a picture and a list of attributes (strength, ...)
3. Each fraction has 20 cards, leaving 17 items
4. Each item has a name, a picture, a description which attributes are changed or which ability a hero gains once equipped with it

The Rules

1. The heroes have the following Dungeon & Dragon-style attributes:
   a) **strength:** increases attack power, increases defense
   b) **agility:** increases chance to score a critical hit
   c) **stamina:** increases health and reduces damage taken
2. Heroes have no abilities on their own.
3. Items can give heroes special abilities.
4. A hero can hold up to three items.
5. Abilities influences decisions made by the combat-engine.
6. I realized that it makes sense implementing the *Rock Paper Scissors* 5.9-pattern so that the heroes of each fraction have strengths and weaknesses over other fractions.

Once a player has designed a deck by selecting 3 heroes and choosing items for each hero, it is time to fight other players in combat. Looking into the list of players an opponent for a battle is chosen by an algorithm or manually, based on three facts that are published on this list:

1. the heroes used in the players deck
2. the amount of cards used in this deck
3. some other aggregated values of those cards

It is feasible to fight an opponent whose deck has the same or less strength than yours, but in this game you don't achieve much[1] from fighting a weak opponent. The timeout for a waiting game is four hours. The player can wait for up to three opponents. The player looks at those statistics (*Indicator* 5.5) and chooses an opponent likely to lose this confrontation.

Six heroes fight against each other until one party is dead. The players cannot influence the fight directly because of the *Single Turn* 5.2 pattern.

The first decision the combat-engine has to take is which hero is the first to attack. If no items influence this decision, this is chosen by chance.

Then the normal battle starts. The fight is decided by a Dungeon & Dragon-style engine resulting in a combat report similar to this:

[1]The exact score gained by winning a game is not pointed out here because we only focus on the design patterns and general balancing questions. In this example we use "BattlePoints". The exact values can be determined by testing.

```
Player 1: Heroes A, B, C
Player 2: Heroes D, E, F

Player 1 attacks Player 2 (+10% attack chance Player 1)

Round 1:
A attacks D, D loses 5 HP
B attacks D, D loses 4 HP and dies
E attacks A, A loses nothing (parry)
F attacks A, A loses 15 HP and dies
C attacks E, E loses 4 HP

Round 2:
B attacks E, E loses 5 HP and dies
F attacks B, B loses 4 HP
C attacks F, F loses 4 HP

Round 3:
F attacks C, C loses 5 HP
C attacks F, F loses 5 HP and dies

Player 1 wins with Hero C alive with 4 HP.
Player 1 would score 150 BattlePoints:
- 50 BattlePoints per lost Hero -> 50 remaining
- 20 BattlePoints for injured Hero
Player 1 scores 30 Battlepoints

Player 2 has lost and receives a penalty of -10% attack
chance for 4 hours.
```

The losing player has to heal the wounded heroes and therefore, has an attack chance penalty of four hours. There are many different possibilities to reward or penalize win and loss. But I would not define a rule that says “The loser has to give away three items that are chosen by chance”. That would be a devastating experience for the player, because the combat is not interactive, and each lost combat would then cost valuable items. The game would be imbalanced and would not be fun.

Consequences

To implement this game, I realized that besides the card-design and balancing questions it would have been necessary to implement a combat-engine, additionally to the abstract card-game engine. Also, the question arose if the incentive to win BattlePoints is encouraging enough to play the game. Winning is rewarding, but maybe this is not enough in the long term.

The complexity of this implementation lies within the combat engine. The benefits of using the engine have to be compared to the restrictions it imposes.

Basic patterns like the continuity patterns strongly influence how the game is played, where the complexity of the game is hidden and how much enjoyment the player perceives.

Defining the rules and items, I automatically obeyed the chosen design patterns. Ex.: I would not create an item that says "When you equip a hero with this item, other players cannot see your hero's attributes.". While possible, it is undesirable for a number of reasons: it conceals information in crucial parts of the game, it contradicts the Indicator pattern and it imbalances the game, making it less/no fun.

From this example I learned that the actual objects a game is constructed from have to obey the design patterns to a certain extent, in order to maintain balance and perception of fun and fairness.

Extending the game might become complicated, because items might be overpowered, useless or only applicable in some special situations that might not be predictable.

Simple Role Playing Game

For this setting I chose the following patterns:

1. **Category 5.6:** four fractions
2. **Turn Based 5.2:** in comparison to the first game I want to emphasize the *Freedom of Interaction* 2.4
3. **Personal Deck 5.8:** in collectible card games players have their own decks
4. **The Discard Pile 5.5:** "Trash". Each round players have to abandon cards
5. **Counter 5.7:** health points of heroes can be altered.

The Cards

This time I will use different attributes for the heroes. Each hero has:

1. health points for head, arms, legs and torso - each minimum 5 and maximum 10
2. strength or attack points - minimum 1 maximum 5

1. **Heroes:** 1 hero per deck (representing the player)
2. **Helmets:** 10 per deck. Increases defense on head
3. **Weapons:** 20 per deck. One- or two-handed, increases attack, maximum +4
4. **Shields:** 10 per deck. Increases defense on arms
5. **Trousers and Boots:** 10 per deck. Increases defense on legs
6. **Armors:** 10 per deck. Increases defense on torso
7. **Healing Potions:**: 20 per deck. Remove maximum 5 harm-counters on any body part

The Rules

1. before the game begins every player shuffles each item category and puts them face down in front of them. I realized that I need the *Base* 5.5 here. There are no dynamic piles in this game so this should not be a problem.
2. in the first round every player plays the only hero card to show the others what hero has been chosen
3. each turn a player may choose two cards from any pile and equip the hero with it. If the player chooses not to equip the hero with the drawn card, it has to be put into the trash. If the hero is already equipped with an item at the defined, the currently equipped item has to be put into the trash.
4. after choosing items the player can choose to attack. It is not allowed to attack in the first turn.
5. once the head has as many harm-counters as defense, the game is over and the remaining player has won
6. if the arms have as many harm-counters as defense, the hero cannot consume any more potions
7. if the torso has as many harm-counters as defense, the hero cannot be equipped with new items
8. if the legs have as many harm-counters as defense, the hero cannot attack any more

The battle is decided like this:

```
Player 1 Hero:
  - attack: 2
  - head defense: 10
  - legs defense: 8
  - arms defense: 8
  - torso defense: 5
  - item: a two handed sword-item with +4 attack

Player 2 Hero:
  - attack: 5
  - head defense: 9
  - legs defense: 9
  - arms defense: 7
  - torso defense: 6
  - item: a shield with +5 arm defense

Player 1 rolls a dice to decide which region will be
```

```
targeted by the attack.

1,2 = torso
3 = arms
4 = legs
5,6 = head

If player 2 wants, one dice is rolled to decide if
arms or legs are used to defend the attack.

1, 3, 5 = arms
2, 4, 6 = legs

Depending on the outcome, harm-counters are put on the
affected body part.
```

Consequences

Using *Counter* 5.7 the gameplay focuses on the heroes defense statistics instead of card interaction.

Allowing the player to draw new items each round requires *The Base* 5.5.

This game could also be played with real cards and counters.

Comparison

The question, if the game is better suited for real life implementation rather then digital implementation, depends on the environment the game is played in, the design patterns used, the complications found playing the game, etc.

On the other hand, some digitally implemented card games may even be too complicated to play in real life. An example are card games that use complicated card arrangements like *Pyramid*. However, in most card games every player takes responsibility for the own card's arrangement and all players manage the shared cards together. Sometimes this function is fulfilled by a special player, normally called the dealer. Digitally based card games make this position obsolete, letting every player concentrate only on their own cards. This does not implicate that players automatically trust in digital solutions. An example are gambling card games which are played remotely. Although poker is played over the internet world wide, [34] points out that digital implementations hosted in countries with lax gambling regulations cannot necessarily be trusted.

There are parameters that directly indicate which patterns have to be applied. In this example the parameter "collectible card game" implies that the *Personal Deck* 5.8 pattern has to be used, because game balance is influenced by having more cards.

CHAPTER 6

Implementation

6.1 Introduction

The implementation of the model was done in three prototypes, each having a different goal. The shared goal of all prototypes was to create a browser based game experience that allows to test the model and evaluate different interaction paradigms.

The web browser as a plattform for application development takes advantage of the WORA concept (“write once run anywhere”) [38]. Normally web based applications have a server and a client side. To understand and evaluate card game development it is sufficient to focus on the client side and frontend development methodologies.

The following subsections shortly describe the final prototype, compare it to its predecessors, and explain the technology (libraries, frameworks) used to create it.

6.2 User Interface

Both prototypes implement the game *Uno*. However, it is possible to use the same code base to build other games, as the game logic and assets are decoupled from the main application.

The goal of the first prototype was to enable the players to build the game before and while playing it. The plan was to allow players to freely create, remove, or change Tableaux, Cards, Piles and Rules. It turned out, that it is not required to build a user interface for this task, as the basic structure of a game does not change that often.

The second prototype was more restrictive. It automatically started the game and did not allow the player to do the shuffling and dealing. Also, it did not allow illegal moves by returning the card to the players hand. Inspired by the original idea to let the player control everything, the second prototype allowed the user to resize, rotate, and move tableaux.

The final prototype is a mix of the original idea and the second approach. It is allowed to do illegal moves, however the interface shows that the player did an illegal move. It is not possible to transform the tableaux and game objects, because positioning and sizing is defined by the game (two player *Uno*).

Both prototypes use *drag and drop* as the core interaction mechanic to move cards from one pile to another.

While the card positioning in the second prototype was implemented using a simple rotation, the final prototype uses a circle segment improving the look and feel.

6.3 Technology and Frameworks

The technology and methodologies used to develop desktop like applications or games on the basis of the browser engine are rapidly changing. Comparing an article from 2008 [71] to modern approaches like React Native from 2017 [50] shows how quickly the methodologies evolve. Also, the different approaches are strongly influenced by the companies behind them.

Today, frameworks, libraries and component based approaches are competing, and they all promise to solve classic problems when programming for the web browser, such as asynchronous execution in a single threaded environment while still delivering a high frames per second ratio.

The final prototype was built using *React* [21] which implements the reactive programming paradigm [44]. The build chain is based upon *npm* and the *react-app* [60] template. This includes a lot of features supporting frontend browser development:

- live reloading
- source mapping
- hot code reloading
- automatic handling of static assets
- extensive debugging tools
- a build chain based on *webpack*
- offline capabilities (progressive web app)

Second Prototype

For the second prototype I built a custom build chain based on *grunt* [32], *bower* [11], and *grunt-file-blocks* [33]. It turned out, that the custom build chain is hard to maintain and complicated to extend. Using additional technologies such as *HammerJS* [35] for touch based drag and drop and *Compass* [27] for css authoring further complicated the setup.

Third Prototype

The setup I used for the final prototype was originally developed by Microsoft [73] and is configuration less and built on *npm* and *babel* [5]. Because there is no configuration, the developer can not exchange frameworks or libraries the setup uses, which eases build chain maintenance and dependency management. The HTML templates are written in *JSX* [42]. The logic is implemented in *EcmaScript 6* [1] and strongly typed using *Typescript* [72].

The shuffle algorithm used in the final prototype is based upon *fisher-yates in-place shuffling* [68].

6.4 SVG in the Browser

Using SVG, scalable vector graphics, in the browser is one possible approach to solve the problem of different device-pixel-resolutions. Bitmap graphics are upscaled on devices with higher DPI (Dots Per Inch), which can cause blurry images. SVG's have been supported by browsers for a couple of years now (2017) and performance is comparable in most major browsers except Opera [75], however, there are differences in implementation.

While there exist frameworks for working with SVG's in the browser like *RaphaelJS* [64], I did not use any framework for embedding the SVG's.

In the second prototype I embedded the SVG's using an `iframe`. Embedding them using an `img`-tag would not have been possible because Safari does not support *SVG Fragment Identifiers* ([39] [14]), which I used to bundle all cards in one file.

The final prototype was optimized for request-performance, which suffers using the iframe-fragment-technique, and does not use SVG's.

6.5 Drag and Drop with Touch in the Browser

Touch based interaction in the browser is still problematic [62]. The relevant W3C standard (*Touch Events* [23]) is still listed as draft (in 2017) and is implemented differently by browser vendors.

Libraries like the aforementioned *HammerJS* normalize development across major browsers and help detecting gestures. Also, different devices have different multi touch sensor behaviour.

The final prototype does not require multi touch, as it does not use gestures like "pan" or "zoom". To enable touch based interaction for the final prototype I used *react-dnd-multi-backend* [61].

6.6 Comparison to Existing Frameworks

There exist a lot of 2d game engines. All engines require the development of additional game logic, such as score counting, continuity management, player interaction mechanics, or rule management. For card games the following two engines have been considered:

1. GCCG - http://gccg.sourceforge.net

2. Vassal - http://vassalengine.org

While *GCCG* was specifically developed to support collectible trading card games *Vassal* has a more general approach and is oriented towards board games. Both frameworks would not have allowed to experiment with different interaction techniques.

6.7 Assets and Prototypes

Assets

1. 52 Standard Card Deck - Chris Aguilar - https://code.google.com/p/vectorized-playing-cards [3]
2. 52+2 wild Standard Card Deck - David Bellot - http://svg-cards.sourceforge.net [7]
3. Uno Card Deck - Дмитрий Фомин (Dmitry Fomin) (Own work) [CC0], via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:UNO_cards_deck.svg [25]

Source Code and Online Version

The final prototype can be found on https://abendstille.at/uni/master/proto3. The code for the final prototype is hosted on GitHub: https://github.com/gruzilla/react-uno.

The second prototype can be found on https://abendstille.at/uni/master/proto2. The code for the second prototype is also hosted on GitHub: https://github.com/gruzilla/cardgames.

CHAPTER 7

Conclusion

When I started working on this thesis the first idea was to write about cards in a more general meaning. Discovering that cards are used in many different contexts the focus shifted towards card games. Research showed, that there is very little literature on card games. So I focused on the design patterns and on an abstract model.

I came to the conclusion that card games are not only about interacting with cards. They are about an abstract meta game where lots of different aspects are taken into consideration.

To create the catalogue of *Digital Card Game Patterns* 5 and to give a comprehensive overview 34 (digital) *Card Games* A.1 have been analyzed by using a method described by Björk and Holopainen [10]. With those patterns other games can be analyzed and compared and new games can be designed. Much more *Games* A.2 have been considered and I continue to discover new games and patterns.

Real life card games are more about aesthetics, social distinction, and physical interaction that are hard to digitize as shown in *Interaction* 3 using Löwgren's interaction aesthetics [47]. This is why most digital card games are not card games, they are *video games*.

Digital card games can feel like real card games, but these examples are virtual simulations, not digital representations (see examples in *State of the Art* 2.1). The degree of *Freedom of Interaction* 2.4 reaches from click-only, drag and drop, and touch to much more. During my research I learned that some people refuse digital representations of real-life card games because they "use different rules" or "don't make fun". Future work could try to identify and measure those elusive aspects of card games that get lost.

Almost all digital card games I researched are competitive and players know only a part of the game state (imperfect information). Implementing two *Prototypes* 6.1 for the web browser on touch based devices, I was able to test different game design and interaction patterns, to work with *The Model* 4, to show differences in the front end build chain and state management, and to learn about the problems of scalable vector graphics and touch based drag and drop in the browser.

The final prototype can be accessed here: https://abendstille.at/uni/master/proto3

There are two interesting questions concerning the model for future work:

Future work could be to test the limitations of the model by implementing other games with the same engine.

Also, based on the model-information and game state changes, machine learning algorithms could be trained using reinforcement learning.

APPENDIX A

List of Card Games

A.1 List of Card Games

1000 Blank White Cards

A public domain card game which allows the players to make the cards.

https://www.boardgamegeek.com/boardgame/4550/1000-blank-white-cards

98 Cards

Singleplayer Stacking and Counting Card Game

https://itunes.apple.com/us/app/98-cards/id1234430446

https://play.google.com/store/apps/details?id=com.vdh.ninetyeight.android

Aces Spades

Implementation of the classic Spades game

https://play.google.com/store/apps/details?id=com.concretesoftware.spades_demobuynow

Bridge

Bidding and trick taking game. Complicated scoring rules.

https://www.playok.com/en/bridge

Black Jack

Also known as twenty-one. Counting and betting gambling game.

Calvinball

Game in which you make the rules up as you go along. Rules cannot be used twice. No game is like another.

http://calvinandhobbes.wikia.com/wiki/Calvinball

Canasta

Pattern matching and trick taking card game with Joker.

https://www.playok.com/en/canasta

Card Crawl

Single or multiplayer survival card game

https://itunes.apple.com/us/app/card-crawl/id950955524
https://play.google.com/store/apps/details?id=com.tinytouchtales.cardcrawl

Doppelkopf

Bidding and trick taking card game. Double pack of cards. Disclosure.

https://www.online-doppelkopf.com

Dragon Punch

Hands only card game based on video games like the Street Fighter series. Its all about trying outwit your opponent with the timing and positioning of your attacks and defences.

https://www.kickstarter.com/projects/556682087/dragon-punch-a-tiny-fighting-game-you-can-play-any

Eleusis Express

A card game in which one player secretly decides on a rule which determines which cards may be played on top of each other. The other players then use deductive logic to work out the secret rule.

http://www.logicmazes.com/games/eleusis/express.html

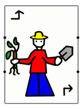
Fast Cards

Singleplayer (on iOs also multiplayer) implementation of the classic games Spit/Speed

https://itunes.apple.com/us/app/fast-cards-card-game/id669627256

https://play.google.com/store/apps/details?id=donnaipe.rapido

Garden Path



Hands only card game that uses positioning and rotation of cards.

https://boardgamegeek.com/filepage/37612/garden-path-full-game-pdf

Hearts+

Implementation of the classic Hearts game

https://itunes.apple.com/us/app/hearts/id398890666

Hex: Shards of Fate

Multiplayer Collectible Card game

http://store.steampowered.com/app/410380/HEX_Shards_of_Fate

Hobbes Card Game

Educational card game. Explores the state of nature.

https://sites.google.com/site/howtodosimulationgames/examples-of-simulations/political-studies/state-of-nature

Incredibrawl

Rock-Paper-Scissors based card game. Allows open table discussion.

http://www.vision3games.com/incredibrawl

Jassen

Multiplayer trick taking strategy card game.

http://jassen.mohrenbrauerei.at

Karnöffel

Oldest identifiable European card game.

http://www.parlettgames.uk/histocs/karnoeffel.html

Lost Cities

Singleplayer or Asynchronous Multiplayer Stacking and Counting Card Game

https://itunes.apple.com/us/app/lost-cities/id465062454

Mao

A shedding-type card game in some ways similar to Uno where (in the advanced version) the winner of a round adds a new rule of her choice to all subsequent rounds.

http://gameofmao.com

Minderheitenquartett

Satirical matching card game where discussions are allowed to resolve decisions.

https://www.minderheiten-quartett.de

Munchkin

Simplified Dungeons and Dragons type card game with phases and battles and categories.

http://www.worldofmunchkin.com/game

Oh Quay

Hands only card game where the order of your hand makes a difference.

http://decktet.wikidot.com/game:oh-quay

Quartett

A pattern matching card game also known as Happy Families. Similar to Top Trumps.

Rummy

Pattern matching card game.

https://cardgames.io/rummy

Skat

Old german bidding and trick taking card game.

https://www.playok.com/en/skat

Spite and Malice

Stacking and counting card game. Also known as Skip-Bo.

https://play.google.com/store/apps/details?id=com.twopersonstudio.games.spiteandmalicefree

Star Realms

Multiplayer Collectible Card game

https://itunes.apple.com/us/app/star-realms/id893447125
https://play.google.com/store/apps/details?id=com.starrealms.starrealmsapp
http://store.steampowered.com/app/438140/Star_Realms

Throwdown

Multiplayer Collectible Card game based on cartoons currently aired primarily in USA

https://itunes.apple.com/de/app/animation-throwdown/id1080816579
https://play.google.com/store/apps/details?id=com.kongregate.mobile.throwdown.google

Uncharted: Fight for Fortune

Single or multiplayer small-device-based card game based on the Uncharted Saga

https://www.playstation.com/en-ie/games/uncharted-fight-for-fortune-psvita

Werewolf

Single Turn card game. The card you get defines the role you play in the village.

https://www.playwerewolf.co/rules

Whist

Plain trick taking card game.

https://cardgames.io/whist

Yu-Gi-Oh! Duel Links

Collectible card game.

https://itunes.apple.com/de/app/yu-gi-oh!-duel-links/id1068378177
https://play.google.com/store/apps/details?id=jp.konami.duellinks
http://store.steampowered.com/app/601510

A.2 Also Considered Games

- 1000 Blank White Cards
- 98 Cards
- A Few Acres of Snow
- Android Netrunner
- Bartok
- Black Jack
- Bohnanza
- Bridge
- Calvinball
- Canasta
- Cardcassone
- Chess
- Dominion
- Domino
- Doppelkopf
- Dragon Punch
- Eleusis Express
- Fast Cards
- Garden Path
- Go Fish
- Hanafuda Koi Koi
- Hearthstone
- Hearts+
- Hex
- Hobbes card game
- Incredibrawl
- Jassen
- Jawker
- Karnöffel
- Kongai
- Legendary: Alien Encounters

- Ligretto
- Lost Cities
- Mafia
- Mage Knight
- Magic Duels
- Magic: The Gathering
- Mana Clash
- Marriage
- Memory
- Minderheitenquartett
- Mod 13
- Munchkin
- Nex
- Nomic
- Oh Quay
- Old Maid
- One and Thirty
- Open-Face Chinese Poker
- Order and Chaos Duels
- Poker
- Poker: Texas Hold'em
- President
- Pyramid
- Pyramid Solitaire Saga
- Quartett
- Resistance
- Rummy
- Schnapsen
- Schnopsn
- Set
- Skat
- Skip-Bo
- Solitaire
- Spades
- Speed
- Spit
- Spite and Malice
- Star Realms
- Stress
- Superhero Audition
- Tarock
- Top Trumps
- Uncharted: Fight for Fortune
- Uno
- Werewolf
- Whist
- WoW TCG
- Yu-Gi-Oh

Bibliography

[1] ECMAScript 6. http://www.ecma-international.org/ecma-262/6.0. Accessed: 2017-12-06.

[2] Kyle Russell Aaron Baker, Jared Hatfield. https://code.google.com/p/card-surface/. Accessed: 2015-04-04.

[3] Chris Aguilar. 52 standard card deck. Accessed: 2017-12-05.

[4] Victor Asal. Playing games with international relations. *International Studies Perspectives*, 6(3):359–373, 2005.

[5] Babel. https://babeljs.io. Accessed: 2017-12-06.

[6] Dave Bayer and Persi Diaconis. Trailing the dovetail shuffle to its lair. *The Annals of Applied Probability*, pages 294–313, 1992.

[7] David Bellot. 52+2 wild standard card deck. Accessed: 2017-12-05.

[8] Larry Bethurum. http://pdp-10.trailing-edge.com/decuslib10-01/01/43,50110/bingo.gam.html, 1966. Accessed: 2017-11-01.

[9] Staffan Bjork and Jussi Holopainen. Patterns in game design. 2004.

[10] Staffan Bjork and Jussi Holopainen. Games and design patterns, 2006.

[11] Bower. https://bower.io. Accessed: 2017-12-06.

[12] Jacob Buur and Astrid Soendergaard. Video card game: an augmented environment for user centred design discussions. In *Proceedings of DARE 2000 on Designing augmented reality environments*, pages 63–69. ACM, 2000.

[13] caniuse.com. https://caniuse.com/#feat=webgl. Accessed: 2017-11-18.

[14] SVG caniuse.com. https://caniuse.com/#feat=svg-fragment. Accessed: 2017-12-06.

[15] Canvas2D specification. http://www.w3.org/tr/2dcontext2/. Accessed: 2013-10-02.

[16] Yu-Gi-Oh Wikia Community. http://yugioh.wikia.com/wiki/first_turn_kill. Accessed: 2015-07-19.

[17] Chris Crawford. *Chris Crawford on game design*. New Riders, 2003.

[18] curiouser.co.uk. http://www.curiouser.co.uk/paradoxes/wildcard.htm. Accessed: 2017-11-20.

[19] Raimund Dachselt and Robert Buchholz. Natural throw and tilt interaction between mobile phones and distant displays. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 3253–3258. ACM, 2009.

[20] G. Dharamshi. Generic java rule engine framework, May 17 2005. US Patent 6,895,575.

[21] React Docs. https://reactjs.org/docs. Accessed: 2017-12-06.

[22] Encyclopaedia Britannica. http://www.britannica.com/ebchecked/topic/641944/whist. Accessed: 2015-04-03.

[23] W3C Touch Events. https://dvcs.w3.org/hg/webevents/raw-file/tip/touchevents.html. Accessed: 2017-12-06.

[24] Jose M. Font, Tobias Mahlmann, Daniel Manrique, and Julian Togelius. *A Card Game Description Language*, pages 254–263. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[25] Дмитрий Фомин (Dmitry Fomin). Uno card deck (own work) [cc0], via wikimedia commons. Accessed: 2017-12-05.

[26] Martin Fowler. https://martinfowler.com/bliki/rulesengine.html, 2009. Accessed: 2017-11-20.

[27] Compass CSS Authoring Framework. http://compass-style.org. Accessed: 2017-12-06.

[28] Pablo Funes. Complexity measures for complex systems and complex objects. *Last modified on 28th October*, 1996.

[29] Game Engines. https://github.com/bebraw/jswiki/wiki/game-engines. Accessed: 2013-04-18.

[30] Go Fish Javascript/CSS3 Card-Game. http://www.gofish-cardgame.com/. Accessed: 2013-04-19.

[31] Khronos Group. https://get.webgl.org/get-a-webgl-implementation/. Accessed: 2017-11-18.

[32] Grunt. https://gruntjs.com. Accessed: 2017-12-06.

[33] grunt-file blocks. https://www.npmjs.com/package/grunt-file-blocks. Accessed: 2017-12-06.

[34] Chris Hall and Bruce Schneier. Remote electronic gambling. In *Computer Security Applications Conference, 1997. Proceedings., 13th Annual*, pages 232–238. IEEE, 1997.

[35] HammerJs. https://hammerjs.github.io. Accessed: 2017-12-06.

[36] HTML5 Game Engines. http://html5gameengines.com/game-engine-overview/. Accessed: 2013-04-18.

[37] Robin Hunicke, Marc Leblanc, and Robert Zubek. Mda: A formal approach to game design and game research. In *In Proceedings of the Challenges in Games AI Workshop, Nineteenth National Conference of Artificial Intelligence*, pages 1–5. Press, 2004.

[38] Minh Q. Huynh and Prashant Ghimire. Browser app approach: Can it be an answer to the challenges in cross-platform app development?

[39] W3C SVG Fragment Identifiers. https://www.w3.org/tr/svg/linking.html#svgfragmentidentifiers. Accessed: 2017-12-06.

[40] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 234–241. ACM, 1997.

[41] Henry Jenkins. Game design as narrative architecture. *Computer*, 44:s3, 2004.

[42] object-oriented programming language JSX, a statically-typed. https://jsx.github.io. Accessed: 2017-12-06.

[43] Jürgen Ludwig. http://landsknechtsrotte.de/portal/k-einleitung.html. Accessed: 2013-10-16.

[44] Kennedy Kambona, Elisa Gonzalez Boix, and Wolfgang De Meuter. An evaluation of reactive programming and promises for structuring collaborative web applications. In *Proceedings of the 7th Workshop on Dynamic Languages and Applications*, DYLA '13, pages 3:1–3:9, New York, NY, USA, 2013. ACM.

[45] Albert HT Lam, Kevin CH Chow, Edward HH Yau, and Michael R Lyu. Art: augmented reality table for interactive trading card game. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 357–360. ACM, 2006.

[46] Seth Lloyd. Measures of complexity: a nonexhaustive list. *IEEE Control Systems Magazine*, 21(4):7–8, 2001.

[47] Jonas Löwgren. Toward an articulation of interaction esthetics. *New Review of Hypermedia and Multimedia*, 15(2):129–146, 2009.

[48] Carsten Magerkurth, Adrian David Cheok, Regan L Mandryk, and Trond Nilsen. Pervasive games: bringing computer entertainment back to the real world. *Computers in Entertainment (CIE)*, 3(3):4–4, 2005.

[49] Chris Marrin. Webgl specification. *Khronos WebGL Working Group*, 2011.

[50] [author.] Masiello, Eric and [author.] Friedmann, Jacob. *Mastering react native :*. Packt,, Birmingham, [England] ; Mumbai, [India] :.

[51] Mike Develin. http://bantha.org/ develin/cardgames.html, 2002. Accessed: 2015-04-03.

[52] Bill Moggridge and Bill Atkinson. *Designing interactions*, volume 17. MIT press Cambridge, 2007.

[53] Janet H Murray. The last word on ludology v narratology in game studies. In *DiGRA 2005 Conference: Changing views of worlds in play*, 2005.

[54] Ather Nawaz. A comparison of card-sorting analysis methods, 2012. Paper presented at APCHI 2012, Matsue, Japan 28-31 August, and published in Proceedings of the 10th Asia Pacific Conference on Computer-Human Interaction, pp. 583-592.

[55] Filip Gokstorp Neal Terrell, Joel Koepp, Dan Hutchins, and John Warwick. https://www.jinteki.net. Accessed: 2017-11-01.

[56] D Parlett. What is a ludeme? *Incompleat Gamester*, 2007. http://www.davpar.eu/gamester/ludemes.html.

[57] David Parlett. *A history of card games*. Oxford University Press, USA, 1991.

[58] Burkhard Peuschel and Wilhelm Schäfer. Concepts and implementation of a rule-based process engine. In *Proceedings of the 14th international conference on Software engineering*, pages 262–279. ACM, 1992.

[59] Photon Storm Ltd. http://phaser.io/. Accessed: 2015-02-28.

[60] react app. https://github.com/facebookincubator/create-react-app. Accessed: 2017-12-06.

[61] react-dnd-multi backend. https://github.com/louisbrunner/react-dnd-multi-backend. Accessed: 2017-12-06.

[62] HTML5 Rocks. https://www.html5rocks.com/en/mobile/touch. Accessed: 2017-12-06.

[63] Kay Römer and Svetlana Domnitcheva. Smart playing cards: A ubiquitous computing game. *Personal Ubiquitous Comput.*, 6(5-6):371–377, January 2002.

[64] A. Krishna. Sagar. *Instant RaphaelJS starter*.

[65] Mizuki Sakamoto and Tatsuo Nakajima. Augmenting yu-gi-oh! trading card game as persuasive transmedia storytelling, 2013.

[66] Jesse Schell. *The Art of Game Design: A book of lenses*. Taylor & Francis US, 2008.

[67] Alireza Sahami Shirazi, Tanja Döring, Pouyan Parvahan, Bernd Ahrens, and Albrecht Schmidt. Poker surface: combining a multi-touch table and mobile phones in interactive card games. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, page 73. ACM, 2009.

[68] Fisher-Yates Shuffle. https://bost.ocks.org/mike/shuffle. Accessed: 2017-12-06.

[69] Kevin Silver. What puts the design in interaction design, 2007.

[70] David Sirlin. http://www.sirlin.net/articles/designing-kongai, 2014. Accessed: 2017-11-20.

[71] Frank Thiesing and Sebastian Kortemeyer. Entwicklung moderner web-anwendungen mit open-source-bausteinen. *Informatik-Spektrum*, 31(2):115–132, Apr 2008.

[72] TypeScript. https://www.typescriptlang.org. Accessed: 2017-12-06.

[73] Microsoft TypeScript-React-Starter. https://github.com/microsoft/typescript-react-starter. Accessed: 2017-12-06.

[74] Richard Van Eck. Digital game-based learning: It's not just the digital natives who are restless. *EDUCAUSE review*, 41(2):16, 2006.

[75] Gojko Vladić, Neda Milić, Željko Zeljković, and Darko Avramović. Evaluating web browser graphics rendering system performance by using dynamically generated svg.

[76] W3C SVG Specification. http://www.w3.org/tr/svg11/. Accessed: 2013-04-19.

[77] Daniel Wigdor and Dennis Wixon. Mechanics, dynamics, and aesthetics-chapter 16:the application of mda. In *Brave NUI World*, pages 107–114, 2011.

[78] James L Williams. *Learning html5 game programming: A hands-on guide to building online games using Canvas, SVG, and WebGL.* Addison-Wesley Professional, 2012.

[79] Francis Willughby, David Cram, Jeffrey L. Forgeng, and Dorothy Johnston. *Francis Willughby's Book of Games: A Seventeenth-Century Treatise on Sports, Games, and Pastimes*. Ashgate Publishing, Ltd., 2003.