

# Cherries

## Game Design Document

Berend Wouda, NGamed

April 4, 2009

During late 2006 and early 2007, there was demand for an IRC game that could be played with a small amount of players on the `irc.gnug.org` server. Cherries was consequently conceived.

Cherries is a simple and easy to understand, newly designed card game for 2 or more players. It revolves around a deck of cards, of which a player takes one card every turn. The player may then *keep* it or *give* it to another player. The receiving player must handle the card. The card may change a player's cherry stockpile or score, or be a residing modifier. The goal of the game for is to score points by managing a cherry stockpile where scoring may vary as multiple game types are supported.

Of this game a digital implementation is to be made. This implementation will make use of the SIF framework, to enable the game to be played using the IRC protocol as a front end. This allows people to play the game in IRC channels with the use of CherryBot, the IRC bot that will implement the Cherries game.

This project is approached with no set methodology and has only two deliverables: this game design document and the software implementation.

## Preface

This document is a game design document for the Cherries game, and its software implementation. It contains the full design of the game and a comprehensive design overview of the software. It also contains metadata such as the original idea, the attached roles, and the approach to the project.

### Version

#### **6 January 2008**

Document structure created.

#### **29 January 2008**

Game design chapter finished.

#### **4 April 2009**

Balance issues aside, game design (and subsequently its chapter) completely finished.

- “Cards” section reorganised and embedded into the “Gamestate” section.
- Game flow, turn flow, and modifying change activity diagrams added.
- Basic card visual designs added.
- Standard Set card visual designs added.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Idea . . . . .	7
1.2	NGamed . . . . .	7
1.3	References . . . . .	7
1.4	Contact . . . . .	8
<b>2</b>	<b>Approach</b>	<b>9</b>
2.1	Project . . . . .	9
2.1.1	Goal . . . . .	9
2.1.2	Conditions . . . . .	9
2.1.3	Methodology . . . . .	9
2.1.4	Documentation . . . . .	10
2.1.5	Deliverables . . . . .	10
2.2	Facilities . . . . .	10
<b>3</b>	<b>Game Design</b>	<b>11</b>
3.1	Overview . . . . .	11
3.2	Requirements . . . . .	11
3.3	Gamestate . . . . .	11
3.3.1	Gametype . . . . .	12
3.3.1.1	End-of-turn scoring condition . . . . .	12
3.3.1.2	Winning condition . . . . .	12
3.3.1.3	Reference graphs . . . . .	12
3.3.1.4	Example gametypes . . . . .	12
3.3.1.5	Official gametype . . . . .	13
3.3.2	Card sets . . . . .	14
3.3.2.1	Cards . . . . .	14
3.3.2.1.1	Card types . . . . .	15
3.3.2.1.1.1	Cherry . . . . .	15
3.3.2.1.1.2	Score . . . . .	15
3.3.2.1.1.3	Modifier . . . . .	16
3.3.2.1.2	Visual design . . . . .	16
3.3.2.1.2.1	Size . . . . .	16
3.3.2.1.2.2	Front . . . . .	16
3.3.2.1.2.3	Back . . . . .	17
3.3.2.2	Symbol . . . . .	17

3.3.3	Deck . . . . .	17
3.3.4	Players . . . . .	18
3.3.4.1	Player . . . . .	18
3.3.4.1.1	Received card stack . . . . .	19
3.3.4.1.2	Modifier card stack . . . . .	19
3.3.4.1.3	Cherry stockpile . . . . .	19
3.3.4.1.4	Score . . . . .	19
3.3.4.2	Turn order . . . . .	19
3.3.4.3	First player . . . . .	19
3.3.5	Game flow step . . . . .	19
3.4	Gameplay . . . . .	19
3.4.1	Game setup . . . . .	20
3.4.2	Game flow . . . . .	20
3.4.2.1	Change in players . . . . .	21
3.4.2.2	Deck renewal . . . . .	21
3.4.2.3	Turn flow . . . . .	21
3.4.2.3.1	Drawing cards . . . . .	22
3.4.2.3.2	Giving cards . . . . .	22
3.4.2.3.3	Handling cards . . . . .	23
3.4.2.3.3.1	Handling cherry cards . . . . .	23
3.4.2.3.3.2	Handling score cards . . . . .	23
3.4.2.3.3.3	Handling modifier cards . . . . .	23
3.4.2.3.4	Changing score . . . . .	24
3.4.2.3.4.1	Modifying change . . . . .	24
3.4.2.3.5	Aging modifiers . . . . .	24
3.4.3	Game end . . . . .	25
3.5	Conflict resolution . . . . .	25
3.6	House rules . . . . .	25
<b>4</b>	<b>Software Design</b>	<b>27</b>
4.1	Overview . . . . .	27
4.2	Requirements analysis . . . . .	27
4.2.1	Functional requirements . . . . .	27
4.2.2	Non-functional requirements . . . . .	27
4.2.3	Pseudo-requirements . . . . .	27
4.3	System design . . . . .	27
4.3.1	Design goals . . . . .	27
4.3.2	Architecture . . . . .	27
4.3.3	Design decisions . . . . .	27
4.4	Object design . . . . .	27
4.4.1	Class model . . . . .	27
4.4.2	Object model . . . . .	27
4.4.3	Dynamic model . . . . .	27
4.5	Implementation . . . . .	27

<b>Appendix A: Standard Set card list</b>	<b>28</b>
Cherry cards . . . . .	28
Score cards . . . . .	29
Modifier cards . . . . .	30

# 1 Introduction

## 1.1 Idea

During late 2006 and early 2007, there was an increase in the popularity of an IRC game on the GnugIRC server ([irc.gnug.org](http://irc.gnug.org)) called *Acro*, which revolved around finding sentences or word groups to fit a randomly generated acronym according to a randomly selected theme. Although the game was fun, especially in large groups, it required players to vote for each other's acronyms, so the game could not be played with just two players. Some players expressed disappointment in that games could not always be played due to a lack of players.

During the same period I wanted to write an IRC bot for no particular reason other than interest, and I'm always interested in making games, so the idea arose to create an IRC bot that could support a (simple) game that could be played with two or more players.

After this the popularity of IRC gaming on the GnugIRC server waned, possibly due to a lack of players, and my time was taken up by other things. Over time though, I've been designing this game off and on, and am now determined to create the game and IRC bot as a personal project.

Related to this is the development of SIF, the Simple IRC Framework, which is the framework the IRC bot will run on. This separation came due to the wish to perhaps in the future make more IRC related applications, and my drive to generalize, abstract, and encapsulate.

## 1.2 NGamed

NGamed is an informal developer/publisher name used by me for releasing games. The name KirkWarez is not used as per usual with things I release to the internet, because the NGamed name may at some point go commercial. At the time of writing, NGamed is not a company.

## 1.3 References

**RFC 1459** The RFC for the original IRC protocol, by J. OIKARINEN and D. REED in May 1993. Details can be found at <http://tools.ietf.org/html/1459>. This protocol is abstracted by SIF, but the types of messages are still relevant.

**NGamed** The developer of Cherries and CherryBot. Details can be found at <http://www.ngamed.com>.

**SIF (Simple IRC Framework)** The basis of the Cherries IRC bot. Referenceable documentation is not yet available.

## 1.4 Contact

**Berend “Kirk” Wouda** *NGamed, Designer & Developer*

AIM: YunaRaider

ICQ: 71235033

MSN: bjwouda@hotmail.com

Skype: yunaraider

E-mail: kirk@kirkwarez.com, kirk@ngamed.com

Website: <http://www.kirkwarez.com>, <http://www.ngamed.com>

## 2 Approach

### 2.1 Project

#### 2.1.1 Goal

The development of a game for two or more players, and a software adaptation of that game that can conduct the game in an IRC channel.

#### 2.1.2 Conditions

The project can be deemed successful if the following conditions have been met:

1. The game can be played with a minimum of two players.
2. The software can conduct the game inside an IRC channel.
3. The game design documentation is clear, concise, and complete enough to allow other implementations.

#### 2.1.3 Methodology

Unlike normal software projects this project will not be subject to a specific methodology, due to personal preference. The project is already underway at the time of writing, and no set order has been followed. The development is also mixed with the development of SIF. The project will come about in its own way, as it is a personal project and the available time and energy may be limited.

However, the project roughly suffices the following methodology:

1. Document the approach.
2. Design the game and document it.
3. Design the software and document it.
4. Implement the software.
5. Deployment and/or release of the software.

These simple phases are mixed as wished. No advanced software development lifecycle phases are used.

### 2.1.4 Documentation

Documentation for the project will be constructed using LyX, which is a front end for L<sup>A</sup>T<sub>E</sub>X. Publicly released documentation will be made available in Adobe PDF format.

Code documentation will be constructed as comments inside the source code. Publicly released code documentation will be made available in JavaDoc's HTML format.

### 2.1.5 Deliverables

The project will have the following deliverables:

1. The Game Design Document, which contains the game design and the software design.
2. The software implementation.

## 2.2 Facilities

During the project the following facilities will be used:

- **Anatricia** as workstation for development and testing. Anatricia is a PC with an AMD 64 X2 4000+ CPU and 4GB PC5300 DDR2 RAM.
- **Windows 2000** as operating system for development and testing. Other operating systems *may* be used for testing as well.
- **Java 1.6.0** as programming language and virtual machine.
- **Eclipse 3.4.2** as integrated development environment and “versioning” system (through local repositories).
- **L<sup>A</sup>T<sub>E</sub>X** and **LyX 1.6** as documentation tools.
- **JavaDoc 1.6.0** for generating interface documentation.
- **GnugIRC** ([irc.gnug.org](http://irc.gnug.org)) as IRC server for testing.
- **mIRC 6.21** as IRC client for testing.

# 3 Game Design

## 3.1 Overview

The following is a quick introduction to the Cherries game to make the next sections more understandable:

- Cherries is a turn-based card game, that makes use of special Cherries game cards.
- It can be played by two or more players, and any player can join in or leave at any moment.
- The players take turns drawing cards from a central deck and either keep a drawn card or give it to another player. The receiving player has to handle the card.
- A player keeps track of all received cards, all active modifiers, a cherry stockpile (which is an amount of cherries), and a score.
- There are multiple possible gametypes, but the playing rules are always the same. The only things that differ per gametype is the scoring and winning condition parameters.

## 3.2 Requirements

In order to play the Cherries game, the following is required:

- One or more Cherries game card sets (see section 3.3.2).

The following is not required but may be useful during a Cherries game:

- A relatively flat surface.
- Markers to keep track of a player's cherry count, score, and/or modifier lifetimes, such as pen and paper or dice.

## 3.3 Gamestate

The state of a Cherries game consists of the following:

- A gametype, which is described in section 3.3.1.
- A number of card sets, which is described in section 3.3.2.

- A deck, which is described in section 3.3.3.
- A number of players, which is described in section 3.3.4.
- A step in the game flow, which is described in section 3.3.5.

### 3.3.1 Gametype

A Cherries game keeps track of the played gametype. A gametype is a variable aspect of the rules of the Cherries game, and provides parameters for the following conditions:

- The end-of-turn scoring condition, which is described in section 3.3.1.1.
- The winning condition, which is described in section 3.3.1.2.

By varying these two parameters, the length and difficulty of a game can be changed.

Gametypes can be visualized in a graph for easy reference, as described in section 3.3.1.3. Some example gametypes are given in section 3.3.1.4. The official Cherries gametype is given in section 3.3.1.5.

#### 3.3.1.1 End-of-turn scoring condition

A player's score (see section 3.3.4.1.4) is changed near the end of his or her turn according to that player's cherry count and modifier card stack (see section 3.4.2.3.4), and a gametype defines the original score change for the amount of cherries. In its most general form, this parameter is a discrete integer score change *distribution* over nonnegative integer cherry amounts.

#### 3.3.1.2 Winning condition

A game is won when a player has a score (see section 3.3.4.1.4) equal to or higher than a certain score, and a gametype defines this score. This parameter is a single number in  $\mathbb{N}$  (a positive integer).

#### 3.3.1.3 Reference graphs

The reference graph of a gametype has nonnegative integer cherry amount brackets along the horizontal axis, and integer score amounts along the vertical axis. This setup can be used to display the score change distribution parameter using histograms (like any other distribution), and to display the winning score parameter with a single horizontal line along the entire width of the graph.

#### 3.3.1.4 Example gametypes

The distribution parameter can be any discrete distribution of integer score change amounts over nonnegative integer cherry amounts. So, a Cherries game could be played with the following gametype for example:

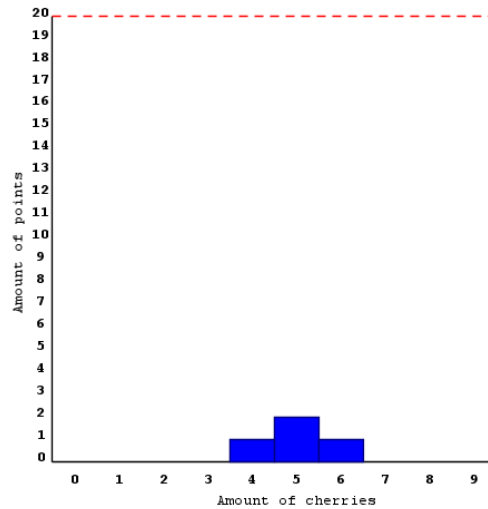


Figure 3.1: Reference graph of an example Cherries gametype.

- If a player has **4** cherries at the end of his or her turn, his or her score change is **1**. If a player has **5** cherries at the end of his or her turn, his or her score change is **2**. If a player has **6** cherries at the end of his or her turn, his or her score change is **1**. Otherwise, his or her score change is **0**.
- If a player has a score of **20** or higher, he or she wins the game.

Figure 3.1 shows this gametype as a reference graph.

A Cherries game could even be played with something like the following gametype:

- If a player has **1** cherry at the end of his or her turn, his or her score change is **3**. If a player has **2** cherries at the end of his or her turn, his or her score change is **2**. If a player has **3** cherries at the end of his or her turn, his or her score change is **1**. If a player has **5** cherries at the end of his or her turn, his or her score change is **-1**. If a player has **6** cherries at the end of his or her turn, his or her score change is **-2**. If a player has **7** cherries at the end of his or her turn, his or her score change is **-3**. If a player has **20** cherries at the end of his or her turn, his or her score change is **20**. Otherwise, his or her score change is **0**.
- If a player has a score of **30** or higher, he or she wins the game.

Figure 3.2 shows this gametype as a reference graph.

### 3.3.1.5 Official gametype

The official gametype for Cherries is the following gametype:

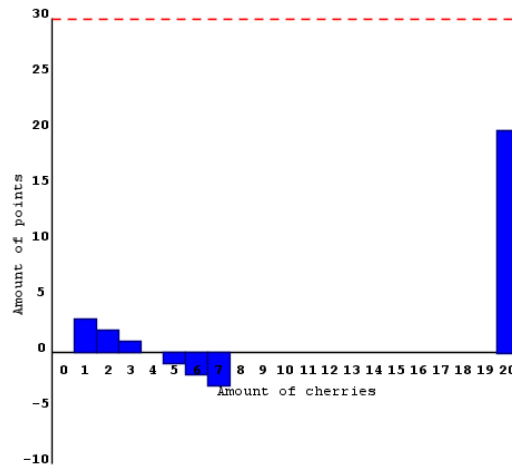


Figure 3.2: Reference graph of an example Cherries gametype.

- If a player has **5** cherries at the end of his or her turn, his or her score change is **1**. Otherwise, his or her score change is **0**.
- If a player has a score of **10** or higher, he or she wins the game.

Figure 3.3 shows this gametype as a reference graph.

### 3.3.2 Card sets

A Cherries game keeps track of the used card sets. A card set consists of:

- A number of Cherries game cards, which is described in section 3.3.2.1.
- An identifier symbol, which is described in section 3.3.2.2.

The following official sets exist:

- The Standard Set, which is listed in Appendix A. It is the first and currently only official Cherries card set.

Any combination of sets can be used to play a Cherries game. Therefore, each set should contain an adequate amount of cherry cards (see 3.3.2.1.1.1). This way, each set can be used on its own to play a Cherries game, and combinations of sets do not decrease the percentage of cherry cards.

#### 3.3.2.1 Cards

A card set consists of a number of Cherries game cards specific to that set. A Cherries game card consists of the following:

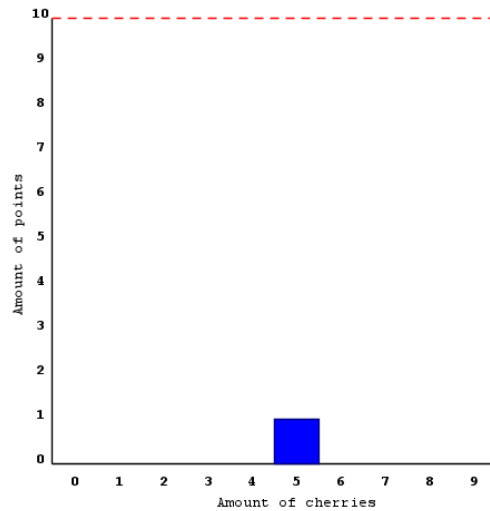


Figure 3.3: Reference graph of the official Cherries gametype.

- A number of attributes according to its type, which is described in section 3.3.2.1.1.
- A visual design, which is described in section 3.3.2.1.2.

**3.3.2.1.1 Card types** Cherries game cards have a type, which is one of the following:

- Cherry, which is described in section 3.3.2.1.1.1.
- Score, which is described in section 3.3.2.1.1.2.
- Modifier, which is described in section 3.3.2.1.1.3.

**3.3.2.1.1.1 Cherry** Cherry cards are cards that change a player's cherry stockpile.

Cherry cards have the following attributes:

- A name.
- A type, which is the 'Cherry' type.
- A cherry change, which is a number in  $\mathbb{Z}$  (a negative integer, zero, or a positive integer). This number is used to change a cherry stockpile (see section 3.4.2.3.3.1).

**3.3.2.1.1.2 Score** Score cards are cards that change a player's score.

Score cards have the following attributes:

- A name.
- A type, which is the 'Score' type.

- A score change, which is a number in  $\mathbb{Z}$  (a negative integer, zero, or a positive integer). This number is used to change a score (see section 3.4.2.3.3.2).

**3.3.2.1.1.3 Modifier** Modifier cards are cards that process and modify a player's cherry or score change.

Modifier cards have the following attributes:

- A name.
- A type, which is the 'Modifier' type.
- A lifetime, which is a number in  $\mathbb{N}$  (a positive integer). This is the amount of turns a modifier is active (see section 3.4.2.3.5).
- A cherry change modification function, which is a  $\mathbb{Z} \rightarrow \mathbb{Z}$  function. This is the function that is used to calculate the modifier's result when running a cherry change through the modifier card stack (see section 3.4.2.3.4.1). This may also be an identity function to have the modifier not affect a cherry change.
- A score change modification function, which is a  $\mathbb{Z} \rightarrow \mathbb{Z}$  function. This is the function that is used to calculate the modifier's result when running a score change through the modifier card stack (see section 3.4.2.3.4.1). This may also be an identity function to have the modifier not affect a score change.

**3.3.2.1.2 Visual design** Cherries game cards have a visual design, which consists of the following:

- The size of a card, which is described in section 3.3.2.1.2.1.
- The front design of a card, which is described in section 3.3.2.1.2.2.
- The back design of a card, which is described in section 3.3.2.1.2.3.

**3.3.2.1.2.1 Size** A Cherries game card measures 5cm by 5cm.

**3.3.2.1.2.2 Front** The design of the front of a card depends on the card, and consists of:

- The set of the card.
- The type of the card.
- The name of the card.
- The specifics of the card.
- The picture of the card.

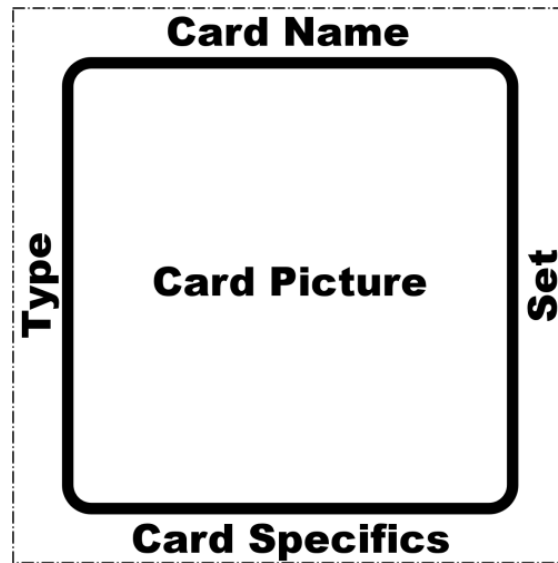


Figure 3.4: Design of the front of a card.

Figure 3.4 shows the *basic* design of the front of a card. The position of the set, type, name, and specifics are fixed. A card may differ in the rest of the layout, including positioning of other elements and colors of all elements and background. See appendix A for the visual designs of the cards from the Standard Set.

**3.3.2.1.2.3 Back** The design of the back of a card is the same for every card. Figure 3.5 shows the design of the back of a card.

### 3.3.2.2 Symbol

A card set has a unique identifier symbol specific to that set. This symbol is marked on all the cards in the set (see section 3.3.2.1.2.2).

### 3.3.3 Deck

A Cherries game keeps track of a single central deck of cards<sup>1</sup>. Its state consists of a nonempty list of cards from a predetermined combination of card sets (see sections 3.4.1 and 3.3.2).

A deck is drawn from (see section 3.4.2.3.1) and as such the state of the deck changes as the game progresses.

A deck's state cannot be the empty list, so when the deck runs out of cards, it is renewed (see section 3.4.2.2) and as such its state then changes.

<sup>1</sup>Note that according to section 3.4.1 multiple decks may be used, however they count as a single deck entity.

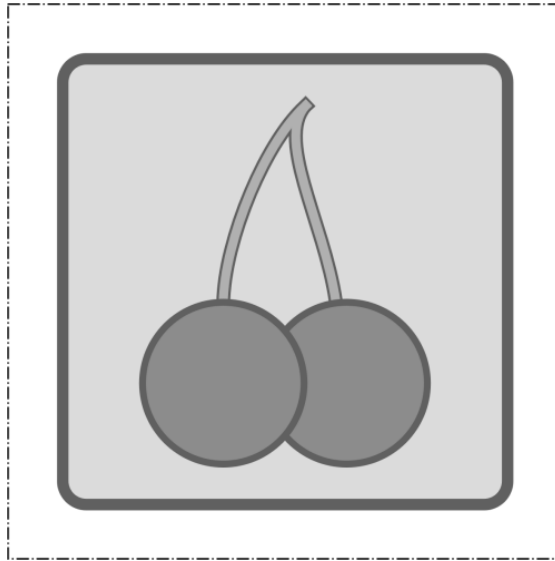


Figure 3.5: Design of the back of a card.

### 3.3.4 Players

A Cherries game keeps track of the following:

- One or more playing players, which are described in section 3.3.4.1.
- The turn order of those players, which is described in 3.3.4.2.
- The first player, which is described in 3.3.4.3.

During the game, the amount of players (and therefore the turn order and who is first in that turn order) can change (see section 3.4.2.1). This then changes the state of the game (possibly suspending or ending it according to section 3.4.2.1).

#### 3.3.4.1 Player

A player's state consists of the following:

- A received card stack, which is described in section 3.3.4.1.1.
- A modifier card stack, which is described in section 3.3.4.1.2.
- A cherry stockpile, which is described in section 3.3.4.1.3.
- A score, which is described in section 3.3.4.1.4.

**3.3.4.1.1 Received card stack** This is a stack of the cherry (see section 3.3.2.1.1.1) and score (see section 3.3.2.1.1.2) cards that have been received, and the received modifier cards that have had their lifetime reduced to 0 (see section 3.3.2.1.1.3). The order of this stack is dictated by the game, and cannot be freely changed. The state of this stack may change as the game progresses (see sections 3.4.2.2, 3.4.2.3.3.1, 3.4.2.3.3.2, and 3.4.2.3.5).

**3.3.4.1.2 Modifier card stack** This is a stack of modifier cards that have a lifetime higher than 0 (see section 3.3.2.1.1.3). The order of this stack is dictated by the game, and cannot be freely changed. The state of this stack may change as the game progresses (see sections 3.4.2.3.3.3 and 3.4.2.3.5).

**3.3.4.1.3 Cherry stockpile** This is a number in  $\mathbb{Z}^*$  (a nonnegative integer) representing an amount of cherries. The state of this stockpile may change as the game progresses (see section 3.4.2.3.3.1).

**3.3.4.1.4 Score** This is a number in  $\mathbb{Z}^*$  (a nonnegative integer) representing an amount of points. The state of this score may change as the game progresses (see sections 3.4.2.3.3.2 and 3.4.2.3.4).

### 3.3.4.2 Turn order

The turn order is a circular sequence of players in the Cherries game. The order of this sequence is determined by who started the game (see section 3.3.4.3) and the subsequent change in players (see section 3.4.2.1).

### 3.3.4.3 First player

The player who started the game (see section 3.4.1) is considered the first player. If the player leaves, the next player in the turn order becomes the first player (see sections 3.3.4.2 and 3.4.2.1).

## 3.3.5 Game flow step

A Cherries game keeps track of the current step in game flow. The state of the game flow changes as the game progresses (see section 3.4.2).

## 3.4 Gameplay

A Cherries game consists of the following phases:

1. The game is set up, which is described in section 3.4.1.
2. The game flow starts, which is described in section 3.4.2.
3. The game ends, which is described in section 3.4.3.

### 3.4.1 Game setup

The following has to be done before starting the Cherries game flow:

1. Decide on the gametype that will be played (see section 3.3.1).
2. Decide on the card sets that will be used (see section 3.3.2).
3. Arrange a turn order between all playing players (see section 3.3.4.2). For example, players could sit in a circular sequence and use a clock-wise turn order.
4. Decide and remember who starts the game (see section 3.3.4.3). For example, players could roll a die or use rock-paper-scissors.
5. Shuffle the available Cherries game cards (see section 3.3.2.1).
6. Arrange the available cards face-down in a deck (see section 3.3.3), and place it in a central place accessible to all players. If a flat surface is used, place it in the center<sup>2</sup>.
7. Initialize any used cherry count and score markers to 0 (see section 3.2).

### 3.4.2 Game flow

The Cherries game flow consists of the following steps:

1. The following steps are repeated until a step ends the game according to section 3.4.3:
  - a) Players may leave or join according to section 3.4.2.1. If this suspends the game, the next step is not executed until the game is unsuspending.
  - b) If the deck is empty, it is renewed according to section 3.4.2.2.
  - c) The first player (see section 3.3.4.3) executes his or her turn according to section 3.4.2.3.
  - d) The following steps are repeated until all players have executed their turns:
    - i. Players may leave or join according to section 3.4.2.1. If this suspends the game, the next step is not executed until the game is unsuspending.
    - ii. If the deck is empty, it is renewed according to section 3.4.2.2.
    - iii. The player that is next according to the turn order (see section 3.3.4.2) executes his or her turn according to section 3.4.2.3.

Figure 3.6 shows the activity diagram of the Cherries game flow.

---

<sup>2</sup>If there is no central place accessible to all players, multiple decks can be used. However, these decks still count as a single entity, and a subdeck running out means that the players using that deck just have to get their cards elsewhere. One could redivide the still available cards when this happens, for example.

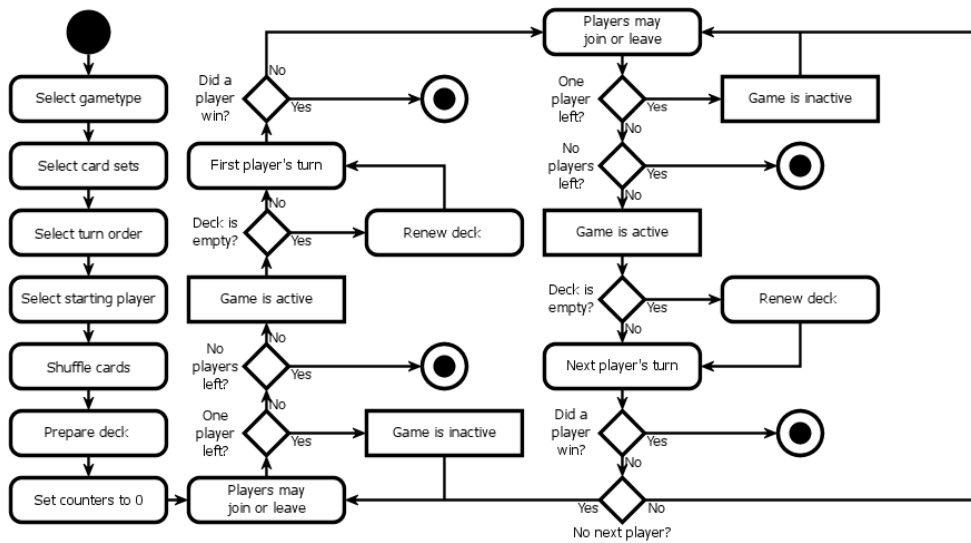


Figure 3.6: Activity diagram of the Cherries game flow.

### 3.4.2.1 Change in players

During a Cherries game players can join or leave. Players can only join or leave before a game flow starts or inbetween player's turns (see section 3.4.2). Players that join are inserted into the turn order between the player that is last in the turn order and the player that is first in the turn order (see section 3.3.4.2 and 3.3.4.3). If the first player leaves the game, the player after the first player in the turn order is considered the first player from then on (see sections 3.3.4.2 and 3.3.4.3).

A game is suspended if there is only one player left. The only things that can happen during a suspended game are the following:

- A player joins, which unsusends the game.
- A player leaves, which ends the game.

### 3.4.2.2 Deck renewal

If all the cards in the deck (see section 3.3.3) have been received, the cards in every player's received card stack (see section 3.3.4.1.1) are shuffled together inbetween the turn of the player that drew the last card and the turn of the player that is next in the turn order (see section 3.3.4), and are used as a new deck.

### 3.4.2.3 Turn flow

A player's turn in the Cherries game flow (see section 3.4.2) consists of the following steps:

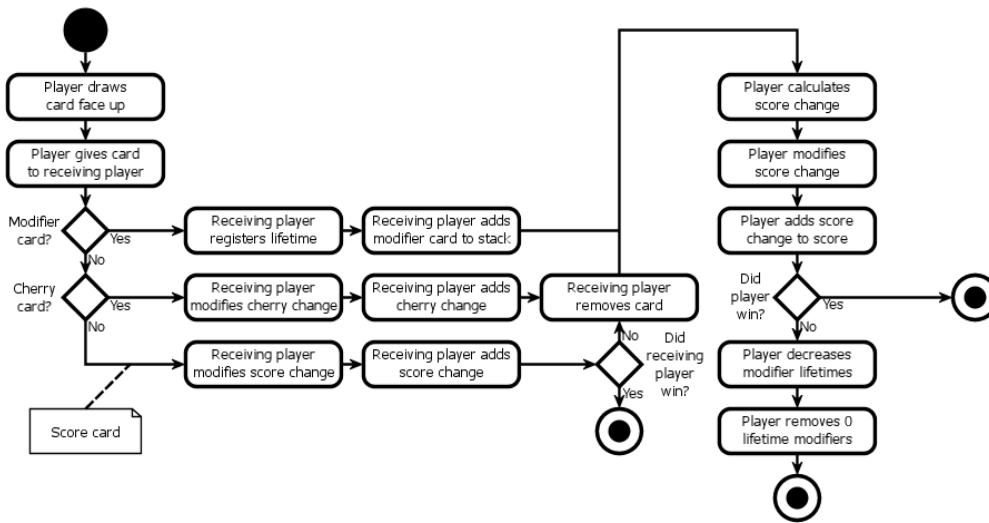


Figure 3.7: Activity diagram of a turn in the Cherries game flow.

1. The player<sup>3</sup> draws a card according to section 3.4.2.3.1.
2. The player keeps that card or gives that card to another player according to section 3.4.2.3.2.
3. The receiving player handles the card according to section 3.4.2.3.3.
4. The player's score is changed according to the gametype according to section 3.4.2.3.4.
5. The player's modifiers are aged according to section 3.4.2.3.5.

Figure 3.7 shows the activity diagram of a turn in the Cherries game flow.

**3.4.2.3.1 Drawing cards** When drawing a card, a card is taken from the top of the deck (see section 3.3.3) by the drawing player, and is held face up so that it is visible to all players.

**3.4.2.3.2 Giving cards** When giving a card to a player, it may in *all* cases be given to *any* player, including the giving player itself. Keeping a card is the same as giving it to oneself.

Other players may advise as to which player should receive the card. However, the player who drew the card has the final decision.

<sup>3</sup>Note that “player” refers to the the player whose turn it is, whereas “receiving player” refers to the player who receives the card.

**3.4.2.3.3 Handling cards** When handling a card, it is handled according to its type (see section 3.3.2.1.1):

- If the card is a cherry card (see section 3.3.2.1.1.1), it is handled as described in section 3.4.2.3.3.1.
- If the card is a score card (see section 3.3.2.1.1.2), it is handled as described in section 3.4.2.3.3.2.
- If the card is a modifier card (see section 3.3.2.1.1.3), it is handled as described in section 3.4.2.3.3.3.

**3.4.2.3.3.1 Handling cherry cards** When handling a Cherry card (see section 3.3.2.1.1.1), the following happens:

1. The cherry change number (see section 3.3.2.1.1.1) is run through the handling player's modifier card stack (see section 3.4.2.3.4.1).
2. The resulting number is added to the handling player's cherry count (see section 3.3.4.1.3). If the cherry count is less than 0 after adding the result, the cherry count is set to 0.
3. The cherry card is added to the top of the handling player's received card stack (see section 3.3.4.1.1).

**3.4.2.3.3.2 Handling score cards** When handling a score card (see section 3.3.2.1.1.2), the following happens:

1. The score change number (see section 3.3.2.1.1.2) is run through the handling player's modifier card stack (see section 3.4.2.3.4.1).
2. The resulting number is added to the handling player's score (see section 3.3.4.1.4). If the score is less than 0 after adding the result, the score is set to 0.
3. The score card is added to the top of the handling player's received card stack (see section 3.3.4.1.1).

**3.4.2.3.3.3 Handling modifier cards** When handling a modifier card (see section 3.3.2.1.1.3), the following happens:

1. Any measures to register the lifetime of the modifier are taken (see section 3.2). If the modifier was *kept* by a player (see section 3.4.2.3.2), its lifetime is increased by one to account for the turn it was played<sup>4</sup>.
2. The modifier card is added to the top of the handling player's modifier card stack (see 3.3.4.1.2).

---

<sup>4</sup>Otherwise, modifiers that are kept would live a turn shorter than modifiers that are given due to the lifetime decrement at the end of the keeping player's turn.

**3.4.2.3.4 Changing score** When changing a player's score according to a gametype, the following happens:

1. A score change is calculated from the gametype's score change distribution with the player's cherry count as parameter (see section 3.3.1).
2. The score change is run through the player's modifier card stack (see section 3.4.2.3.4.1).
3. The resulting number is added to the player's score (see section 3.3.4.1.4). If the score is less than 0 after adding the result, the score is set to 0.

**3.4.2.3.4.1 Modifying change** When a player's cherry count (see section 3.3.4.1.3) or score (see section 3.3.4.1.4) is about to be changed (see sections 3.4.2.3.3.1, 3.4.2.3.3.2, and 3.4.2.3.4), the change (which is a number in  $\mathbb{Z}$ ) is run through that player's modifier card stack (see section 3.3.4.1.2) to calculate the actual change<sup>5</sup>. This is done in the following way:

1. Execute the relevant function (cherry or score) of the modifier (see section 3.3.2.1.1.3) at the bottom of the modifier card stack (see section 3.3.4.1.2) with the change number as parameter to calculate the next result.
2. Repeat the following steps until there are no more unused modifiers (see section 3.3.2.1.1.3) in the modifier card stack (see section 3.3.4.1.2):
  - a) Execute the relevant function (cherry or score) of the next modifier (see section 3.3.2.1.1.3) in the modifier card stack (see section 3.3.4.1.2) with the last result as parameter to calculate the next result.

The last result is the actual change number that has to be used. Figure 3.8 shows an activity diagram of a run through a modifier card stack.

**3.4.2.3.5 Aging modifiers** When aging a player's modifiers (see section 3.3.2.1.1.3), the following happens:

1. The player's modifiers that have a lifetime (see section 3.3.2.1.1.3) higher than 0 have their lifetimes decreased by one.
2. The player's modifiers that now have a lifetime (see section 3.3.2.1.1.3) of 0 are placed on top of the player's received card stack (see section 3.3.4.1.1). The order in which they are placed on the received card stack is the same as their (relative) order in the modifier card stack (see section 3.3.4.1.2)

---

<sup>5</sup>Note that a change of 0 is still a change, and is run through the modifier card stack.

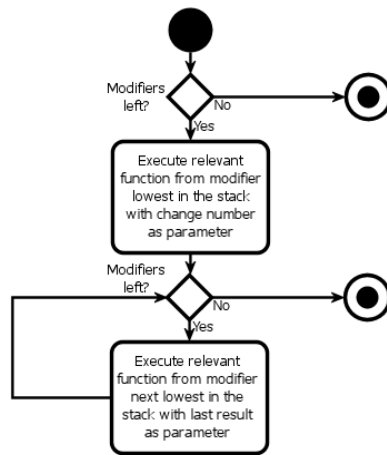


Figure 3.8: Activity diagram of a run through a modifier card stack.

### 3.4.3 Game end

A Cherries game ends when one of the following occurs:

- A player satisfies the winning condition according to the played gametype (see section 3.3.1). That player has then won the game. This can only occur after handling a card (see section 3.4.2.3.3) or at the end of a winning player’s turn (see section 3.4.2.3.4), so it is not be possible for more than one player to win a game.
- There are no more players left as described in section 3.4.2.1.

## 3.5 Conflict resolution

Due to the simple nature of Cherries, conflicts about the execution of the rules should seldom occur<sup>6</sup>. If a conflict occurs, it is most likely an error in the game, and this should be reported so it can be fixed. Therefore, the player that just drew the card that causes the conflict should remove it from the game, and should draw a new card from the deck instead.

## 3.6 House rules

Cherries can also be played with house rules. A number of house rules are imagineable:

---

<sup>6</sup>In fact, conflicts cannot occur in any software implementation because such an implementation would be deterministic.

- Restrictions can be made on when players can leave or join. For example, players may not be allowed to join once the game is in progress, or leave until the game is over.
- Custom gametypes (providing custom scoring or winning condition parameters) can be played.

# 4 Software Design

## 4.1 Overview

## 4.2 Requirements analysis

### 4.2.1 Functional requirements

### 4.2.2 Non-functional requirements

### 4.2.3 Pseudo-requirements

## 4.3 System design

### 4.3.1 Design goals

### 4.3.2 Architecture

### 4.3.3 Design decisions

## 4.4 Object design

### 4.4.1 Class model

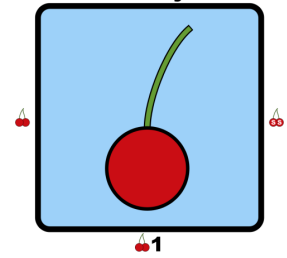
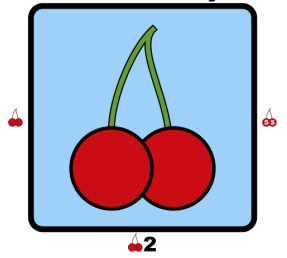
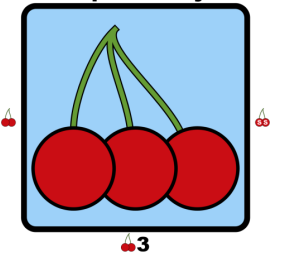
### 4.4.2 Object model

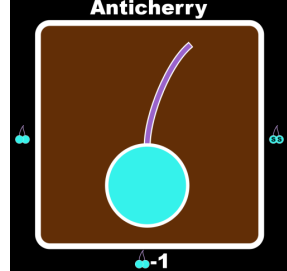
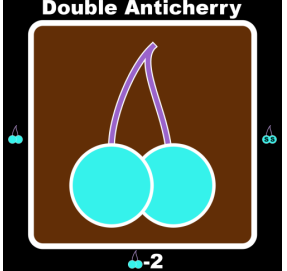
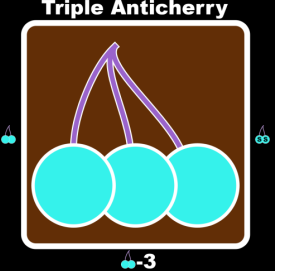
### 4.4.3 Dynamic model

## 4.5 Implementation

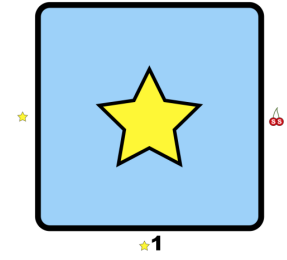
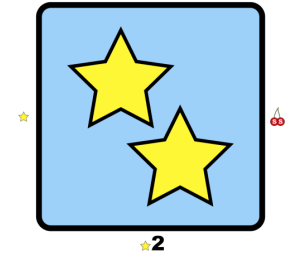
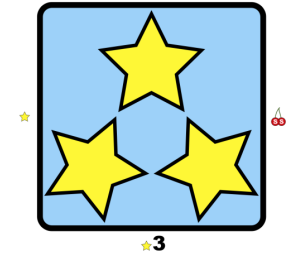
# Appendix A: Standard Set card list




## Cherry cards

	<b>Cherry</b>	<b>Double Cherry</b>	<b>Triple Cherry</b>
<b>Picture</b>			
<b>Name</b>	Cherry	Double Cherry	Triple Cherry
<b>Type</b>	Cherry	Cherry	Cherry
<b>Change</b>	1	2	3
<b>Amount</b>	10	5	2

	<b>Anticherry</b>	<b>Double Anticherry</b>	<b>Triple Anticherry</b>
<b>Picture</b>			
<b>Name</b>	Anticherry	Double Anticherry	Triple Anticherry
<b>Type</b>	Cherry	Cherry	Cherry
<b>Change</b>	-1	-2	-3
<b>Amount</b>	10	5	2

## Score cards

	<b>1 Point</b>	<b>2 Points</b>	<b>3 Points</b>
<b>Picture</b>			
<b>Name</b>	1 Point	2 Points	3 Points
<b>Type</b>	Score	Score	Score
<b>Change</b>	1	2	3
<b>Amount</b>	4	2	1

	<b>-1 Point</b>	<b>-2 Points</b>	<b>-3 Points</b>
<b>Picture</b>			
<b>Name</b>	-1 Point	-2 Points	-3 Points
<b>Type</b>	Score	Score	Score
<b>Change</b>	-1	-2	-3
<b>Amount</b>	4	2	1

## Modifier cards

Picture			
	<b>Name</b>	+1 Hat Of Fruit	-1 Hat Of Unfruit
<b>Type</b>	Modifier	Modifier	Modifier
<b>Lifetime</b>	3	3	2
<b>Cherry function</b>	$next = last + 1$	$next = last - 1$	$next = 0$
<b>Score function</b>	$next = last$	$next = last$	$next = last$
<b>Amount</b>	4	4	2

Picture			
	<b>Name</b>	Cherry Doubler	Cherry Halver
<b>Type</b>	Modifier	Modifier	Modifier
<b>Lifetime</b>	2	2	2
<b>Cherry function</b>	$next = 2last$	$next = \text{floor}(\frac{1}{2}last)$	$next = -last$
<b>Score function</b>	$next = last$	$next = last$	$next = last$
<b>Amount</b>	2	2	2

	<b>Point Boost</b>	<b>Point Drop</b>	<b>Score Shield</b>
<b>Picture</b>			
<b>Name</b>	Point Boost	Point Drop	Score Shield
<b>Type</b>	Modifier	Modifier	Modifier
<b>Lifetime</b>	2	2	1
<b>Cherry function</b>	$next = last$	$next = last$	$next = last$
<b>Score function</b>	$next = last + 1$	$next = last - 1$	$next = 0$
<b>Amount</b>	2	2	2

	<b>Score Doubler</b>	<b>Score Halver</b>	<b>Score Inverter</b>
<b>Picture</b>			
<b>Name</b>	Score Doubler	Score Halver	Score Inverter
<b>Type</b>	Modifier	Modifier	Modifier
<b>Lifetime</b>	1	1	2
<b>Cherry function</b>	$next = last$	$next = last$	$next = last$
<b>Score function</b>	$next = 2last$	$next = floor(\frac{1}{2}last)$	$next = -last$
<b>Amount</b>	1	1	2