# **Ivanhoe**

**Rex Chiu (100904495) Hector Zhang (100947935) Team 12 April 6, 2016**

Table of Contents

Ivanhoe 1

1. Introduction 4

1.1 Motivation 4

1.2 Terminology 4

2. Networking Strategy 5

2.1 Server Strategy 5

2.2 Client Strategy 5

2.3 Message Strategy 5

2.4 Pros and cons Strategy 5

**2.4.1** **Pros of the Networking** 5

**2.4.2** **Cons of the Networking** 5

3. Game Design 6

3.1 Overall Architecture 6

3.2 Design Pattern 6

**3.2.1** **Motivation** 6

**3.2.2** **Design Pattern - MVC** 6

3.3 Refactoring and Implementation 6

**3.3.1** **Elimination of duplicated code** 6

**3.3.2** **Redistribution of responsibilities** 6

3.4 Pros and cons Game Design 7

**3.4.1** **Pros of the MVC** 7

**3.4.2** **Cons of the MVC** 7

3.5 Bells and Whistles 7

4. Game Rules 8

5. Requirements 15

5.1 Functional Requirements 15

5.2 Non-Functional Requirements 16

5.3 Assumptions 16

6. Use Cases 17

6.1 Use Case Diagram 17

6.2 Use Cases 17

6.3 Responsibilities 17

6.4 Unbound Use Case Maps 18

6.5 Bound Use Case Maps 18

7. Design Decisions 19

7.1 Decisions 19

7.2 Structural Model (UML) 19

8. Object Specification 21

9. Interaction Diagrams 22

# **Introduction**

Designed by Rex Chiu and Hector Zhang

Introduction of Ivanhoe

## **1.1 Motivation**

Learning Software Design

Understanding Agile Development

Family with TDD and JUnit test

Experience Design Pattern

Study Scenario and Identify the test cases

Practicing Networking Implementation

## **1.2 Terminology**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Server |  |
| Client |  |
| Rule Engine |  |
| Ivanhoe |  |
| Player |  |
| Game |  |
| Deadwood |  |
| Display |  |
| Hand |  |
|  |  |

# Networking Strategy

## **2.1 Server Strategy**

Server is the host of the Ivanhoe. Server would run the game and control the procedure of the game. The responsibility of the server would be report the result of client’s response to the Rule Engine of Ivanhoe. The key feature of server is communicating with clients over the network. Ivanhoe is a multiple player game. Therefore, the server be handling multi clients in every time. No matter how many requests is received, the server could handle it and report it to Rule Engine. Server would not response to clients if the Rule Engine recognize the request is not valid. When any clients are quit or accidently exit the game, the server could be terminated the game by send the message to rest of the clients. The message is the information of the server is down due to client loss. The server is well implemented with the setting of the network which is allowed to change the setting of the number players in the game, the IP address and port number. It is implemented for running the game over the network in different devices. It is allowed the player play the game in anywhere.

## **2.2 Client Strategy**

Client is the side of Ivanhoe player. Client is able to join the game and play the game and quit the game. The responsibility of the client is participated the game of Ivanhoe. Client is able to join the game and play. The key feature of client is communicating with server over the network. Client is treat as the player side of the Ivanhoe. Therefore, the client is able to do any corresponding action depend on the state of the game which means, client is able to update the newest information to Rule Engine. No matter what the client do, the client is not able to update anything unless server’s response. Client is automatically received the any message from the server. And automatically update the message from the server to the UI. Client is free to report any actions from the players to the server. But the action is not response immediately unless the action is corresponding to the request action from the server. When the server is shut down. Client could receive the message of the server is down and the client could be automatically quit the server and terminated. The client is well implemented with the setting of the network which is allowed to change the setting of the IP address or port number. It is implemented for joining the game over the network. It is allowed the player to join if any game is open.

## **2.3 Message Strategy**

The message is used for communicating between the network which record the data of the current state of Rule Engine. The message is serialized object that is secure to transfer over network. The responsibility of the message is carrying the data packet of Rule Engine. The key feature of the message is encapsulating the header and the body. The header represents the sender, receiver, the state of the game and the type of the state. And the body is player information which is designing by the data collection of HashMap. It is really easier to manipulate for the program. The message could be generated by the Rule Engine which recording the current state of the game. It is including the information of each players and the state of the game. The message is designed by carrying the information. Therefore, the client will be able to use the message to update UI immediately.

## **2.4 Pros and cons Strategy**

### **2.4.1 Pros of the Networking**

* Server could update the information to client immediately
* Client could send message of any action in any time to server.
* Either Server or Client could not interrupt the Rule Engine in any time
* Server and client takes less control of the Rule Engine

**2.4.2 Cons of the Networking**

* Client could send lot of garbage messages to the server
* Server need handle some garbage messages from the client who is not current player

# **Game Design**

## **3.1 Overall Architecture**

Server

Server would receive and send

Client would receive and send

Rule Engine Filter Message

## **3.2 Design Pattern**

### **3.2.1 Motivation**

Why we choose the design pattern of MVC?

Separate UI from Game.

Set Networking could be only one who has access to control the game

Our target is letting player who can play the game as controller

MVC is much clear to understand the game easily

### **3.2.2 Design Pattern - MVC**

How the design pattern MVC is implementing to Ivanhoe?

Rule engine contains everything data of the game

Model could handle different state of game (Process Message)

UI would display the data message between Rule Engine and Client

UI would update everything data according to Rule Engine State

Client Action Events would update UI and Send message to Rule Engine

## **3.3 Refactoring and Implementation**

### **3.3.1 Elimination of duplicated code**

How we refactor the duplicated code?

Remove the unused Game functionality from Display and hand

Remove all unused game state and type in Game Configuration

### **3.3.2 Redistribution of responsibilities**

How the game logic is redundancy?

Networking was integrated with Rule Engine

Server gave the control to clients

Client has some control of whole game over state

Server Process Message by different state

Communication between server, client and Rule Engine

## **3.4 Pros and cons Game Design**

### **3.4.1 Pros of the MVC**

MVC would never clash with each other.

Each other could swapped out

Easy to unit test

Separates game logic from UI Design

### **3.4.2 Cons of the MVC**

Difficult to convert from existing code

They are not considering everything

Complexity

View and Controller closely coupled

Changes in model format would need change in VC

## **3.5 Bells and Whistles**

Right click any face-up cards to display larger image of the card

Robustness networking: handling loss of a player

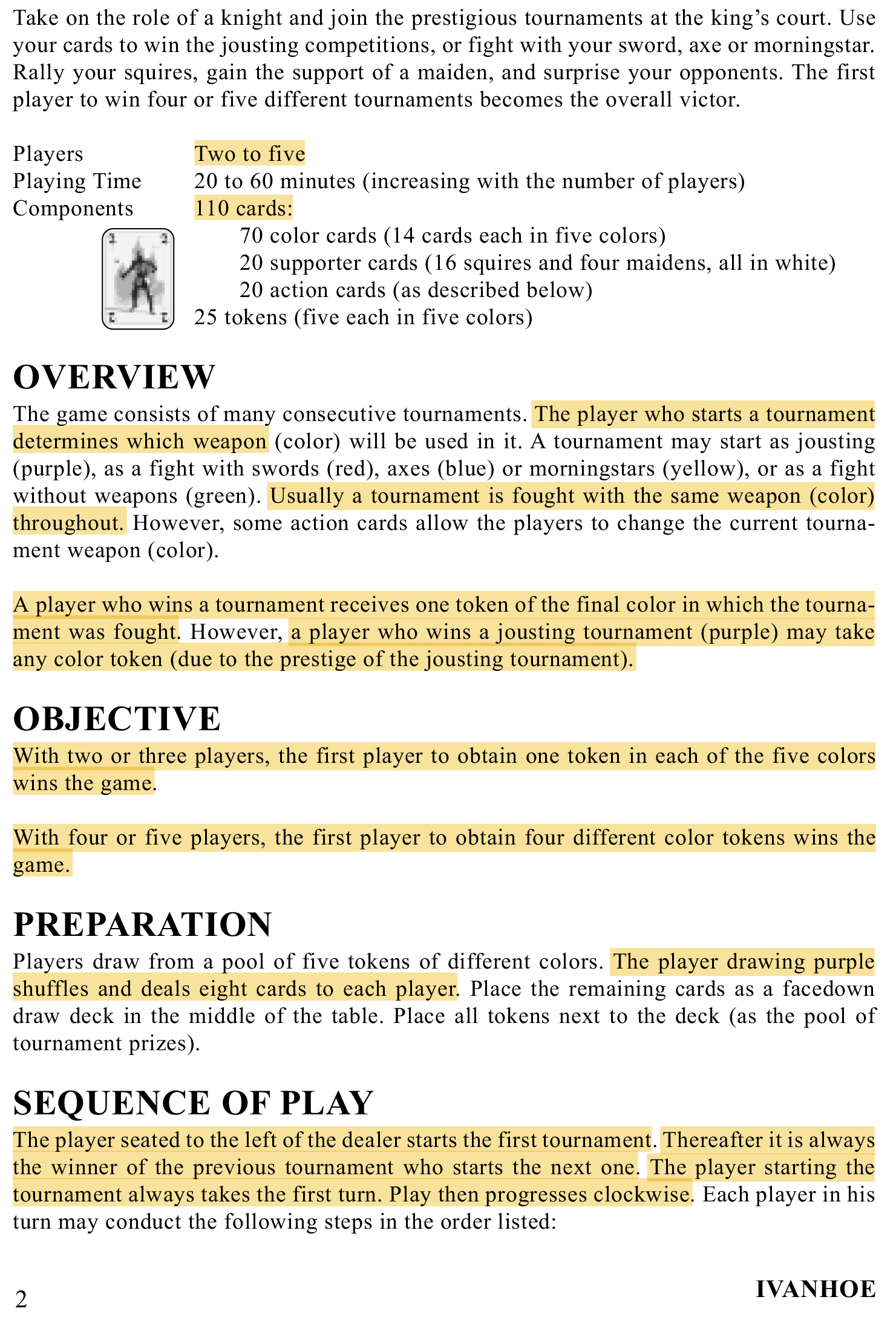
Robustness of game: preventing playing out of turn or invalid card

Tournament Panel: Display players order, status and display total

# **Game Rules**

Below are pdf pages of the official game rule for Ivanhoe.

Detail Information



**GR-01**

**GR-02**

**GR-10**

**GR-12**

**GR-11**

**GR-09**

**GR-08**

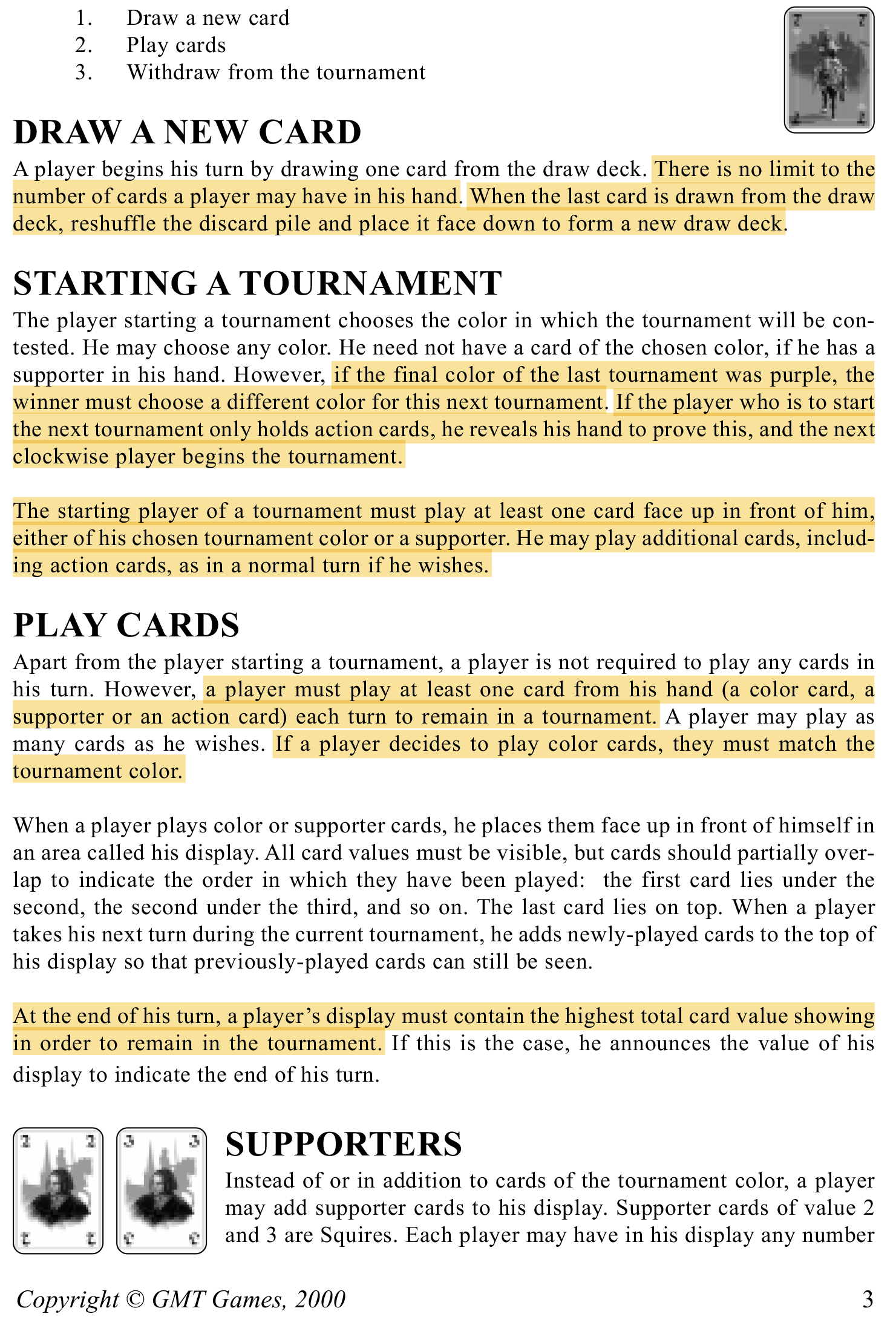
**GR-07**

**GR-06**

**GR-05**

**GR-04**

**GR-03**



**GR-20**

**GR-19**

**GR-18**

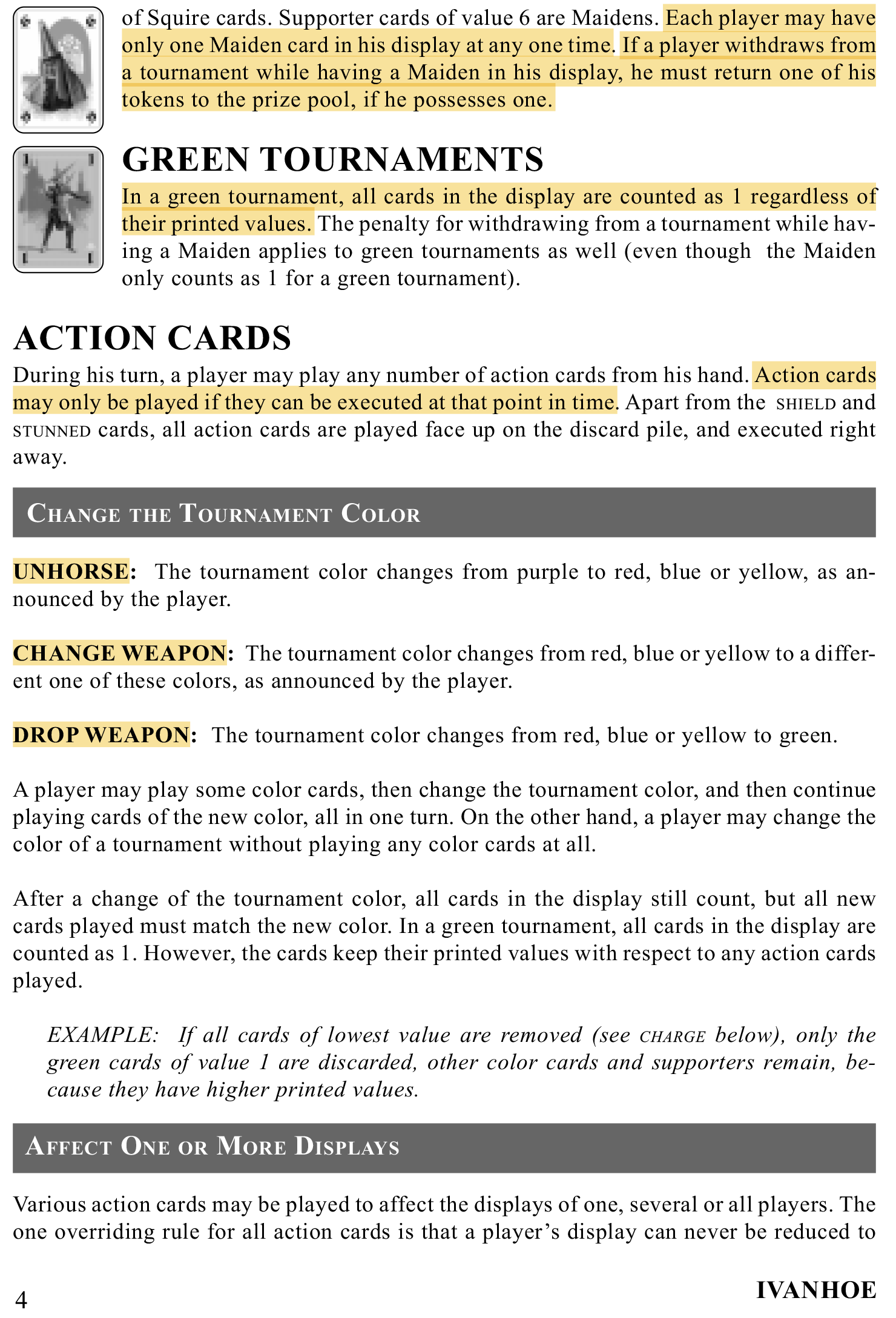
**GR-17**

**GR-16**

**GR-15**

**GR-14**

**GR-13**



**GR-27**

**GR-26**

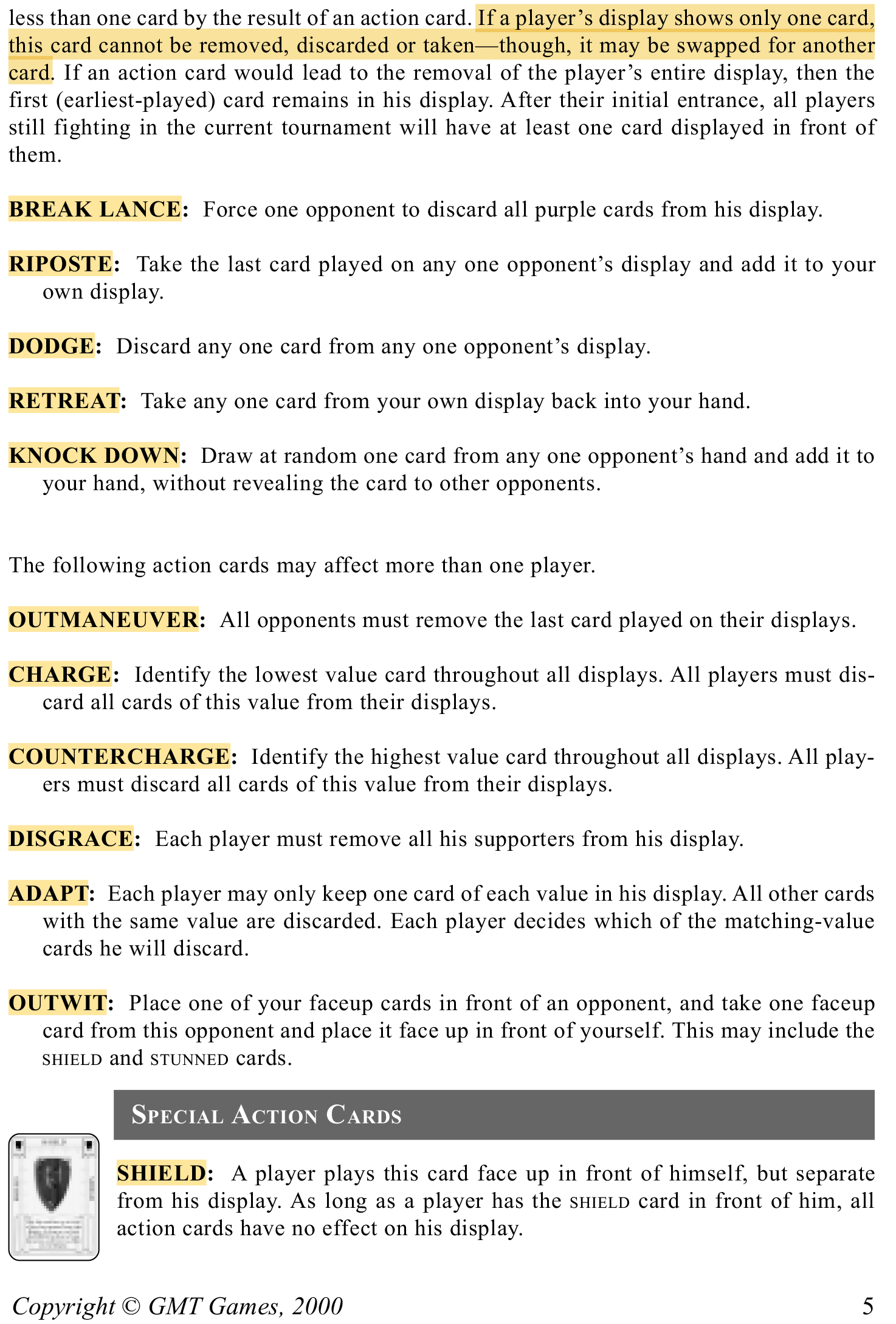
**GR-25**

**GR-24**

**GR-23**

**GR-22**

**GR-21**



**GR-40**

**GR-39**

**GR-38**

**GR-37**

**GR-36**

**GR-35**

**GR-34**

**GR-33**

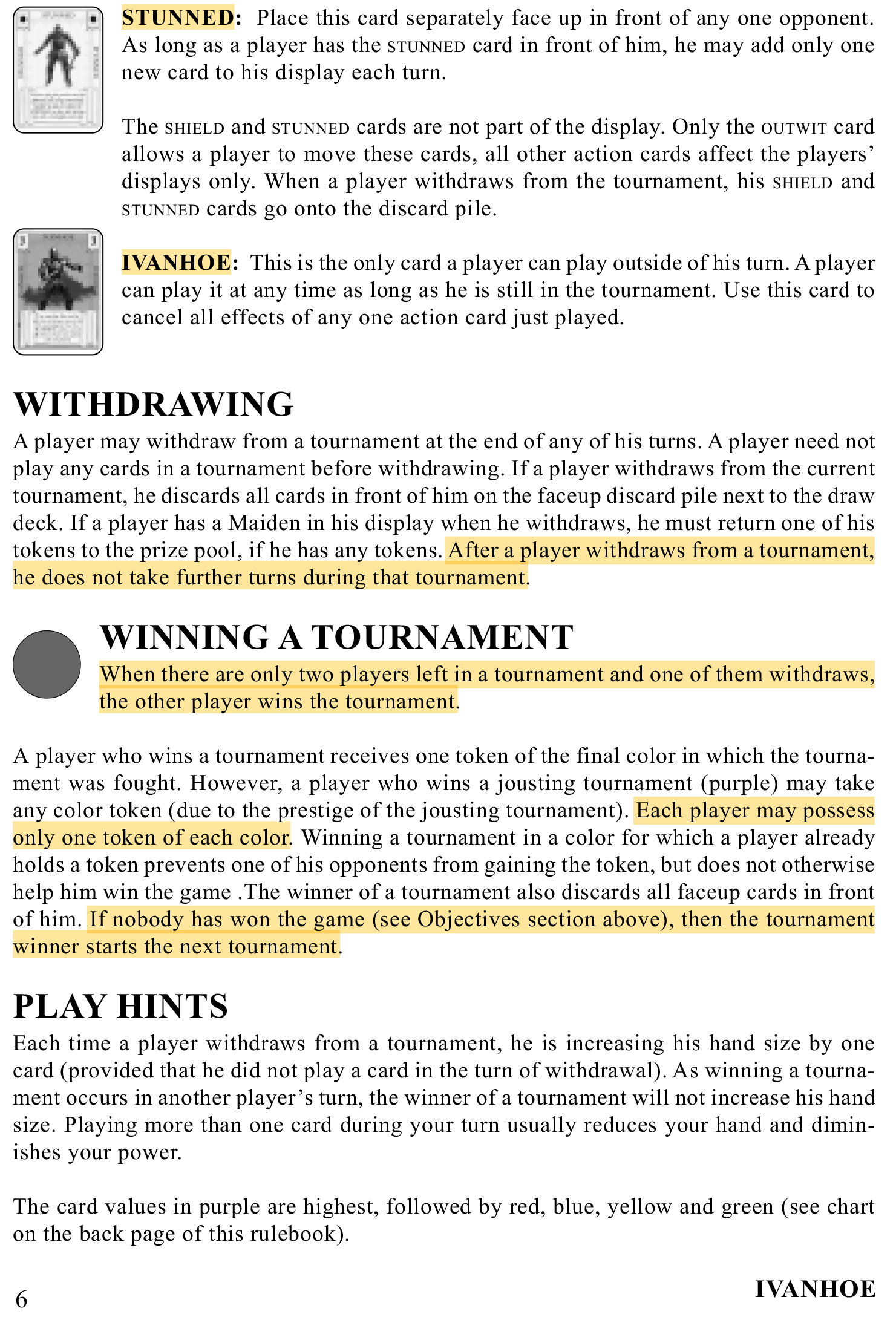
**GR-32**

**GR-31**

**GR-30**

**GR-29**

**GR-28**



**GR-46**

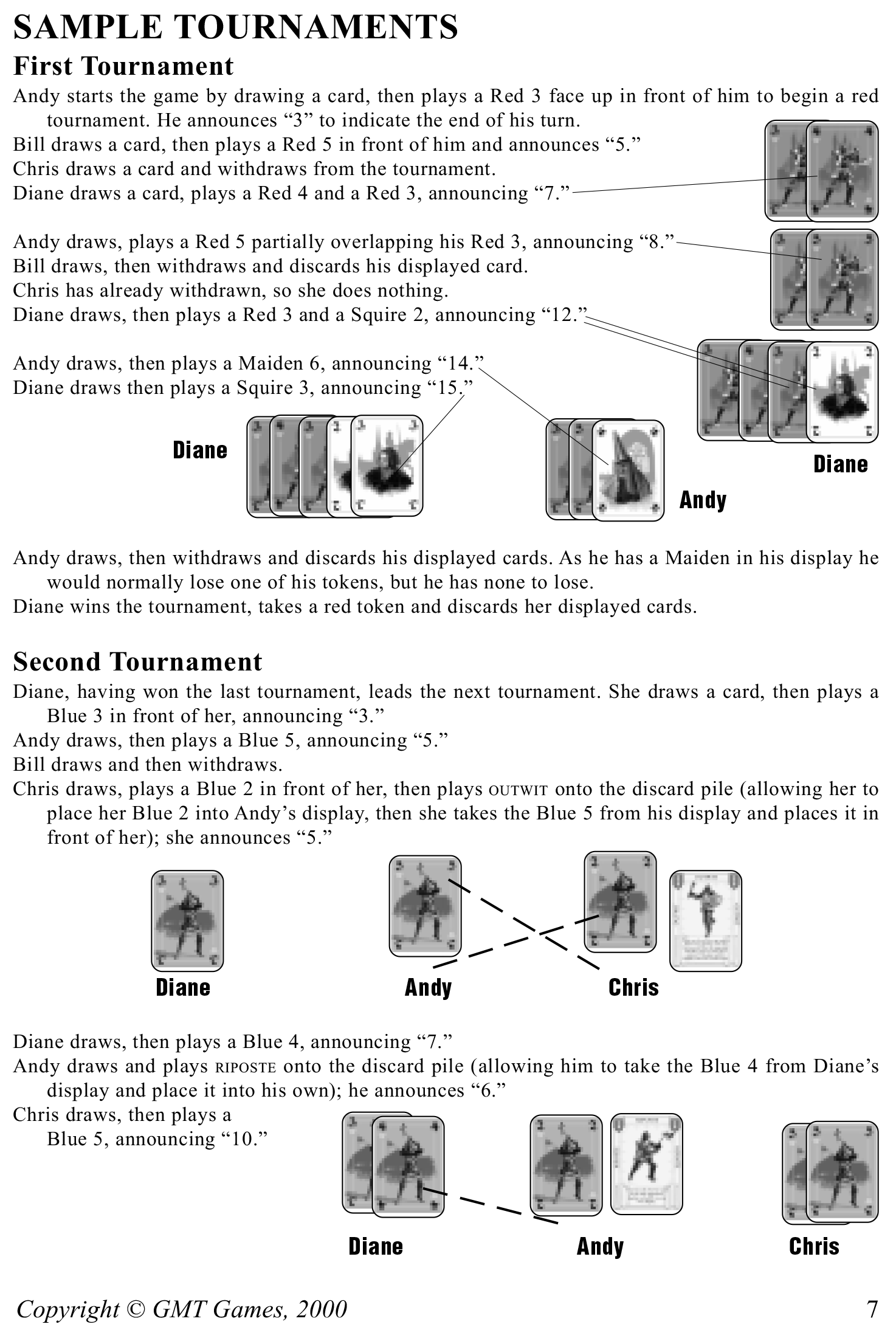
**GR-45**

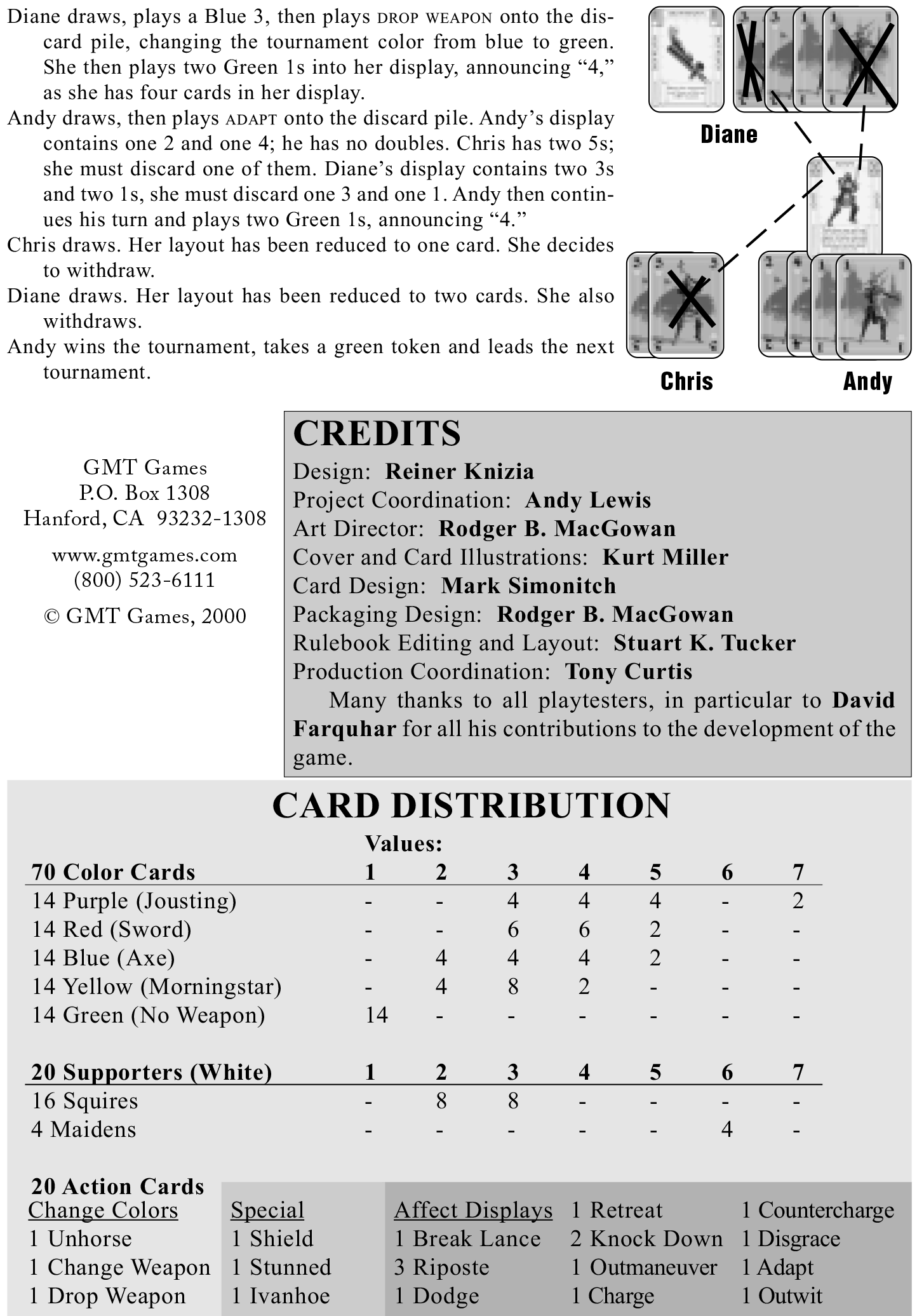
**GR-44**

**GR-43**

**GR-42**

**GR-41**





# **Requirements**

This section contains what features must be implemented. Each requirement has its own unique identifier, description, plus where the decision to come up with it can be traced to. The source of traceability may either be from the official game rules, assumptions, other requirements, or team decisions.

## **5.1 Functional Requirements**

Functional requirements define what behavior and functionality the software must have. They have ben categorized appropriately below, based on different game states of functionality

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Functional Requirement** | **Traceability** | |
| **Network Setup** | | | |
| FR-NS-01 | Network would initialize the Ivanhoe And open the port for player join | GR-09 |
| FR-NS-02 | Each Player would be able to join the network to play Ivanhoe | GR-01 |
| FR-NS-03 | Network would check the total of players joined the game | GR-01 |
| FR-NS-04 | Server would stop waiting for new players if all players joined | GR-01 |
| **Setup** | | |
| FR-S-01 | Initialize the deck to 110 cards | GR-02 |
| FR-S-02 | Deal eight cards to each players | GR-09 |
| FR-S-03 | Initialize the players order | GR-13 |
| FR-S-04 | Initialize the first player | GR-10 |
| **Start Tournament** | | |
| FR-ST-01 | The player who starts a tournament choose tournament color | GR-03 |
| FR-ST-02 | The winner of the previous tournament start next one | GR-11 |
| FR-ST-03 | The player starting tournament takes the first turn | GR-13 |
| FR-ST-04 | If previous color is purple, new tournament color cannot be purple again | GR-16 |
| FR-ST-05 | If player only have actions cards, he reveals his hand to prove this, and the next clock wise player begins the tournaments | GR-17 |
| **Draw Card** | | |
| FR-DC-01 | There is no limit to the number of cards a player can have | GR-14 |
| FR-DC-02 | When the last card is drawn from the deck, reshuffle the deadwood pile as a new deck | GR-15 |
| **Play Card** | | |
| FR-PC-01 | Starting player of a tournament must play chosen tournament color or a supporter card first | GR-18 |
| FR-PC-02 | Player can only play the card match the tournament color | GR-04 |
| FR-PC-03 | Player must play one card from his hand, each turn to remain in a tournament | GR-19 |
| FR-PC-04 | Player decides to play color card, they must play a tournament color card or support card | GR-20 |
| FR-PC-05 | Each player may have only one Maiden card in the display at any time | GR-22 |
| FR-PC-06 | In green tournament, all cards in display are counted as value of one | GR-24 |
| FR-PC-07 | Action cards may only be played if it can be executed at that point in time | GR-25 |
| FR-PC-08 | Player’s display shows only one cards, this card cannot be removed, discarded or taken-though, it may be swapped for another card | GR-29 |
| FR-PC-09 | Playing Unhorse: The tournament color changes from purple to red, blue or yellow, as announced by the player | GR-26 |
| FR-PC-10 | Playing Change Weapon: The tournament color changes from red, blue or yellow to different one of these colors, as announced by the player | GR-27 |
| FR-PC-11 | Playing Drop Weapon: The tournament color changes from red, blue or yellow to green | GR-28 |
| FR-PC-12 | Playing Break Lance: Force one opponent to discard all purple cards from his display | GR-29 |
| FR-PC-13 | Playing Riposte: Take the last card played on any one opponent’s display and add it to your own display | GR-30 |
| FR-PC-14 | Playing Dodge: Discard any one card from any one opponent’s display | GR-31 |
| FR-PC-15 | Playing Retreat: Take any one card from your own display back into your hand | GR-32 |
| FR-PC-16 | Playing Knock Down: Draw at random one card from any one opponent’s hand and add it to your hand, without revealing the card to other opponents | GR-33 |
| FR-PC-17 | Playing Outmaneuver: All opponents must remove the last card played on their displays | GR-34 |
| FR-PC-18 | Playing Charge: Identify the lowest value card throughout all displays. All players must discard all cards of this value from their displays | GR-35 |
| FR-PC-19 | Playing Countercharge: identify the highest value card throughout all displays .All players must discard all cards of this value form their displays | GR-36 |
| FR-PC-20 | Playing Disgrace: Each player must remove all his supporters from his display | GR-37 |
| FR-PC-21 | Playing Outwit: Place one of your face-up cards in front of an opponent, and take one face-up card from this opponent and place it face up in front of yourself. This may include the shield and stunned cards | GR-38 |
| FR-PC-22 | Playing Shield: A player play this card face up in front of himself, but separate from his display. As long as a player has the shield card in front of him, all action cards have o effect on his display | GR-39 |
| FR-PC-23 | Playing Stunned: Place this card separately faces up in front of any one opponent. As long as a player has the stunned card in front of him, he may add only one new card to his display each turn | GR-40 |
| FR-PC-24 | Playing Ivanhoe: This is the only card a player can play outside of his turn. A player can play it at any time as long as he is still in the tournament. Use this card to cancel all effects of any one action card just played | GR-41 |
| **End Turn** | | |
| FR-ET-01 | Player must have the highest total card value to remain in the tournament | GR-21 |
| **Withdraw** | | |
| FR-W-01 | Player withdrawing with a Maiden in the display must remove a token if there is one | GR-23 |
| FR-W-02 | After a player withdraws from a tournament, player cannot take further turns during that tournament | GR-42 |
| FR-W-03 | There are only two players left in a tournament and one of them withdraws, the other player wins the tournament | GR-43 |
| **Win Tournament** | | |
| FR-WT-01 | A player who wins the tournament gets tournament color token | GR-05 |
| FR-WT-02 | A player winning purple tournament can chose any color tokens | GR-06 |
| FR-WT-03 | Each player may process only one token of each color | GR-44 |
| FR-WT-04 | Nobody has won the game then winner starts the next tournament | GR-45 |
| **Game Over** | | |
| FR-GO-01 | With two or three players, the first players to obtain one token in each of the five colors win the game | GR-07 |
| FR-GO-02 | With four or five players, the first player to obtain four different color tokens wins the game | GR-08 |

## **5.2 Non-Functional Requirements**

Non-functional requirements deal with the quality of which the program must satisfy. Concerning aspects such as accessibility, usability, and maintainability

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Non-Functional Requirement** | **Traceability** | |
| NFR-01 | Server would start and initialize the rule engine | FR-NS-01 |
| NFR-02 | Client would be able to join the game of Ivanhoe | FR-NS-02 |
| NFR-03 | Sever would wait for all player to join the game | FR-NS-03 |
| NFR-04 | Server would close the port when all players have joined | FR-NS-4 |

## **5.3 Assumptions**

Throughout figuring out the problem space, assumption needed to be made in order to capture software requirements. Many of the requirements above would therefore trace back to items below. The following table identifies them along with their justifications.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Assumption** | **Justifications** | |
| A-01 | Player must start a new tournament if they can | It is pointless to have a game with no one wants to start |
| A-02 | After winning a purple tournament, the previous tournament is purple regardless of the token chosen | The winning token color does not affect to previous tournament color |
| A-03 | It is possible to withdraw without entering a tournament | Entering the tournament is optional for players |
| A-04 | Can remain in the tournament without adding a card to display | Playing action card can raise your display total to highest and remain in tournament |
| A-05 | Can play action card on a player who is not in the tournament | Some action card would be affect to their hand or display |
| A-06 | Ivanhoe can be used when Shield is played | Shield does not protect against Ivanhoe |
| A-07 | Can play Stunned on a player with Shield | Stunned can bypass shield |
| A-08 | Shield only protect against action from other people | Your own action cards would affect your own display |
| A-09 | Player can only play one card at a time | Overcome the special case of Ivanhoe playing, easier to implement robustness check |

# **Use Cases**

Use case depicts a scenario, the game engine, and the path of game logic that may follow. In this cases of Ivanhoe, it including the player over the networking.

## **6.1 Use Case Diagram**

The diagram of the Ivanhoe over the networking to demonstrate how the Ivanhoe work



## **6.2 Use Cases**

Each use case is detailed in its own table, describing their sequence of events

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Description** |
| UC-01 | Create Server | System create a game of Ivanhoe  Actors:  System  Triggering Event:  Server starts running the Ivanhoe Game  Pre-Condition(s):  No game is created yet.  Main Sequence:   1. Initialize the server setting    1. Set the number of players    2. Set the default IP and port 2. Run the server 3. Waiting for the player to join the game   Resulting Event:  Server is ready for player join  Traceability:  FR-NS-01  NFR-01  NFR-02  GR-01  GR-09 |
| UC-02 | Join Server | Player join the server to play  Actors:  Player, Rule Engine, System  Triggering Event:  Ivanhoe Server is running  Pre-Condition(s):  Player is not already in the Ivanhoe Server  Main Sequence:   1. Player joins the Ivanhoe Server 2. System add player to Rule Engine 3. Ivanhoe initialize the player information 4. System wait for all players to joined 5. System closes all waiting ports when all players joined   Resulting Event:  System updates players list to Rule Engine  Alternative Scenario:  Not all players have joined: System waits until all players have joined  Traceability:  FR-NS-02  NFR-03  NFR-04  GR-01 |
| UC-03 | Setup | Rule Engine set up the game  Actors:  System, Rule Engine  Triggering Event:  All players have joined the server  Pre-Condition(s):  Game is not set up yet  Main Sequence:   1. System tells Rule Engine to setup Ivanhoe 2. Rule Engine initialize the deck 3. Rule Engine initialize the player orders 4. Rule Engine randomly choose first player 5. Rule Engine deal cards to each players 6. Deal card to first player   Result Event:  Ivanhoe is setup  Traceability:  FR-S-01  FR-S-02  FR-S-03  FR-S-04  FR-S-05  GR-02  GR-09  GR-10  GR-13 |
| UC-04 | Start Tournament | A player starts a new tournament  Actors:  Player, Ivanhoe, System  Triggering Event:  Ivanhoe is set up  Pre-Condition(s):  The Tournament did not start yet  Main Sequence:   1. Player prompted to choose a tournament color    1. Ivanhoe tells System to start game    2. System forwards message to Player 2. Player chooses a tournament color 3. Player sends tournament color choice to System 4. System forwards message to Ivanhoe 5. Ivanhoe update the tournament color   Resulting Event:  Player has chosen a tournament color  Alternative Scenarios:   1. Previous color is purple: new tournament color cannot be purple 2. Player only have action cards: next clockwise player start tournament   Traceability:  FR-ST-01  FR-ST-02  FR-ST-04  FR-ST-05  GR-03  GR-11  GR-16  GR-17 |
| UC-05 | Draw Card | Player draws a card  Actors:  Player, Ivanhoe, System  Triggering Event:  Player has chosen a tournament color  Pre-Condition(s):  When it is player’s turn  Main Sequence:   1. Ivanhoe deals card to Player 2. Ivanhoe sends updated information to System 3. System forwards message to Player   Resulting Event:  The player has drawn a card  Alternative Scenarios:   1. When the last card is drawn from the deck: Reshuffle the deadwood pile as the new deck   Traceability:  FR-DC-02  GR-15 |
| UC-06 | Play Card | Player wants to play a card  Actors:  Player, Ivanhoe, System  Triggering Event:  The player has drawn a card  Pre-Condition(s):   1. It is player’s turn 2. Player did not end turn yet 3. Player is not withdrawn   Main Sequence:   1. Player sends card request to System 2. System forwards message to Ivanhoe 3. Ivanhoe handles the card effect 4. Ivanhoe sends updated information to System 5. System forwards message to all players   Resulting Event:  Player play a card  Alternative Scenarios:   1. Card does not match tournament color: Card is not played 2. First card played in the tournament: Card must be tournament color or support 3. Player has Maiden card in the display: Cannot play another Maiden 4. In Green Tournament: All card in the display are counted as value of one 5. Playing action card: Can only be played if it can be executed at that point in time 6. Last card in display: No action card can remove this card 7. Shield in the display: No opponent’s action card can affect display 8. Stunned in the display: Cannot add more than one card to display per turn 9. Ivanhoe: Cancel the effect of any action card   Traceability:  FR-PC-01  FR-PC-02  FR-PC-04  FR-PC-05  FR-PC-06  FR-PC-07  FR-PC-08  FR-PC-22  FR-PC-23  FR-PC-24  GR-18  GR-04  GR-20  GR-22  GR-24  GR-25  GR-29  GR-39  GR-40  GR-41 |
| UC-07 | End Turn | Player ends their turn  Actors:  Player, Ivanhoe, System  Triggering Event:  Player wants to end their turn  Pre-Condition(s):   1. It is player’s turn 2. Player did not end turn yet   Main Sequence:   1. Player sends end turn to System 2. System forwards message to Ivanhoe 3. Ivanhoe check the total of player’s display and finds next player 4. Ivanhoe checks for a winner 5. Ivanhoe sends “Play or Withdraw” to System 6. System forwards message to Player   Resulting Event:  Player ends the turn  Alternative Scenarios:   1. Player display’s total value is not highest: Ivanhoe withdraws the player   Traceability:  FR-ET-01  GR-21 |
| UC-08 | Withdraw | Player withdraw the tournament  Actors:  Player, Ivanhoe, System  Triggering Event:  Player wants/needs to withdraw  Pre-Condition(s):   1. It is player’s turn 2. Player did not withdraw yet   Main Sequence:   1. Player sends withdraw to System 2. System forwards message to Ivanhoe 3. Ivanhoe check if player has Maiden 4. Ivanhoe check if there is a winner 5. Ivanhoe sends next player “Play or Withdraw” to System 6. System forwards message to next Player   Resulting Event:  Player withdraw the tournament  Alternative Scenarios:   1. Player withdraw with Maiden: Player must remove a token if there is one 2. Second last player withdraws from the tournament: The last player wins the tournament.   Traceability:  FR-W-01  FR-W-02  FR-W-03  GR-23  GR-42  GR-43 |
| UC-09 | Win Tournament | A player wins the tournament  Actors:  Player, Ivanhoe, System  Triggering Event:  The second last player withdraw the tournament  Pre-Condition(s):  Last non-withdrawn player in the tournament  Main Sequence:   1. Ivanhoe gives the tournament color token to player 2. Ivanhoe check if player win game 3. Ivanhoe sends “start new tournament” message to System 4. Server forwards message to Player   Resulting Event: Player wins the tournament  Alternative Scenarios:   1. Purple tournament: Player choose any tournament color token 2. Player wins the game: Ivanhoe game is over 3. Player already has tournament color token: Player does not get the tournament color token   Traceability:  FR-WT-01  FR-WT-02  FR-WT-03  FR-WT-04  GR-05  GR-06  GR-44  GR-45 |
| UC-10 | Game Over | A player wins the game  Actors:  Player, Ivanhoe, System  Triggering Event:  A player wins the tournament  Pre-Condition(s):   1. A player has all five tournament color tokens in 2-3 player game 2. A player has any four tournament color tokens in 4-5 player game   Main Sequence:   1. Ivanhoe update the winner information 2. Ivanhoe send the winner information to System 3. System forwards message to Players   Resulting Event:  Ivanhoe game is over  Traceability:  FR-GO-01  FR-GO-02  GR-07  GR-08 |

# **Design Decisions**

This section documents design decisions that have been taken with respect to classes and objects chosen for the system. Included is a Unified Modeling Language (UML) diagram.

## **7.1 Decisions**

|  |  |  |
| --- | --- | --- |
| **ID** | **Design Decision** | **Traceability** |
| DD-01 | **Use a peer-to-peer model for networking** | Group Decision |
| DD-02 | **Separate the networking from Rule Engine** | Group Decision |
| DD-03 | **Client has no control over game state** | Group Decision |
| DD-04 | **MVC is the design pattern for Rule Engine** | Group Decision |
| DD-05 | **Convert object using Object-Oriented Programing** | Group Decision |

## **7.2 Structural Model (UML)**

UML Class Diagram



# **Object Specification**

This section is demonstrating Object Specification of the Project. Below of the table are descripted the all objects for the program of Ivanhoe

|  |  |
| --- | --- |
| Class Name: ClientIvanhoe | |
| Responsibilities:  Create New GUI Client for Ivanhoe | Collaborators:  ClientPanel |

|  |  |
| --- | --- |
| Class Name: ServerIvanhoe | |
| Responsibilities:  Create New GUI Server for Ivanhoe | Collaborators:  HostPanel, AppServer |

|  |  |
| --- | --- |
| Class Name: AppClient | |
| Responsibilities:  Initialize Client Network | Collaborators:  ClientPanel, Message, ClientThread |

|  |  |
| --- | --- |
| Class Name: AppServer | |
| Responsibilities:  Initialize Server Network | Collaborators:  HostPnael, Message, Ivanhoe, Player, ServerThread |

|  |  |
| --- | --- |
| Class Name: ClientThread | |
| Responsibilities:  Initialize Client Thread | Collaborators:  AppClient |

|  |  |
| --- | --- |
| Class Name: ServerThread | |
| Responsibilities:  Initialize Server Thread | Collaborators:  AppServer |

|  |  |
| --- | --- |
| Class Name: Message | |
| Responsibilities:  Initialize Header and Body | Collaborators:  Header, Body |
| Class Name: Header | |
| Responsibilities:  Initialize Header | Collaborators: |

|  |  |
| --- | --- |
| Class Name: Body | |
| Responsibilities:  Initialize Body | Collaborators: |

|  |  |
| --- | --- |
| Class Name: Token | |
| Responsibilities:  Display Information of Token | Collaborators: |

|  |  |
| --- | --- |
| Class Name: Tokens | |
| Responsibilities:  Add Token to Tokens  Remove Token from Tokens  Check if Tokens Contain Token  Clean Tokens  Check if Tokens Has Four Unique Token  Check if Tokens Has All Token  Display Information of Tokens | Collaborators:  Token  Token  Token  Token  Token  Token  Token |

|  |  |
| --- | --- |
| Class Name: Player | |
| Responsibilities:  Withdraw Player  Clear the Display  Check If Has Token  Give Token to Player  Add Card to Display  Add Card to Hand  Check If Player is Withdrawn  Display Information of Player | Collaborators:  Display  Tokens, Token  Tokens, Token  Display, Card  Hand, Card  Tokens, Hand, Display |

|  |  |
| --- | --- |
| Class Name: Card | |
| Responsibilities:  Check Two Card Equal  Is Action Card  Is Supporter  Is Maiden  Is Ivanhoe  Display Information of Card | Collaborators: |

|  |  |
| --- | --- |
| Class Name: Deck | |
| Responsibilities:  Initialize the Deck  Add Multiple Same Cards to Deck  Add Single Card to Deck  Remove Given Card from Deck  Shuffle the Deck  Check If Empty Deck  Clear the Deck  Check If Contain Card  Copy Cards from Different Deck | Collaborators:  Card  Card  Card  Card  Card  Card  Card  Card  Card |

|  |  |
| --- | --- |
| Class Name: Display | |
| Responsibilities:  Increase Number of Card Played  Reset Number of Card Played  Check If Display is Empty  Check Display for Purple Card  Check Display for Value Card  Check Display for Support Card  Check Display for Shield  Check Display for Stunned  Add Card to Display  Remove Card from Display  Check Display for Maiden Card  Update Display with Status  Is Tournament green  Does Display has Shield  Does Display has Stunned  Display Information of Display | Collaborators:  Card  Card  Card  Card  Card  Card  Card  Card  Card |

|  |  |
| --- | --- |
| Class Name: Hand | |
| Responsibilities:  Check If Only Last Card in Hand  Add Card to Hand  Play Card from Hand  Check If Only Action Cards in Hand  Check If Ivanhoe is in Hand  Check If Maiden is in Hand  Display Information of Hand | Collaborators:  Card  Card  Card  Card  Card  Card  Card |

|  |  |
| --- | --- |
| Class Name: Ivanhoe | |
| Responsibilities:  Initialize the Ivanhoe Game  Initialize the Hand of all players  Initialize the Player Orders  Select the First Player  Setup the Game of Ivanhoe  Deal Card to Current Player  Select Tournament Color  Check If Player Play or Withdraw  Check Which Token to Remove  Check If There is A Winner  Play A Card  Check If Player Play Ivanhoe  Change Tournament Color  End Turn  Win Purple Tournament  Process Message  Update Current Player to Next Player  Discard Display from All Players  Check If Player Total is Highest  Player Play Card  Update State  Update Tournament Color  Add Player  Remove Player | Collaborators:  Player, Deck  Player, Deck, Card  Player  Player  Player, Deck, Card  Player, Deck, Card  Player, Message, Display  Player, Message, Tokens, Token, Deck, Display, Card  Player, Message, Tokens, Token  Player, Message, Tokens, Token, Deck, Display, Card  Player, Message, Deck, Hand, Display, Card  Player, Message, Deck, Hand, Card  Player, Message, Deck, Card  Player, Message, Deck, Hand, Display, Card  Player, Message, Tokens, Token  Player, Message, Tokens, Token, Deck, Hand, Display, Card  Player, Display, Card  Player, Display  Player, Hand, Display, Card  Player, Display  Player  Player |

# **Interaction Diagrams**

Contained below are UML 2.0 Interaction Diagrams as Message Sequence Chart. Each correspond to the previous bound use case maps

**Corresponding to UC diagram and each use case**