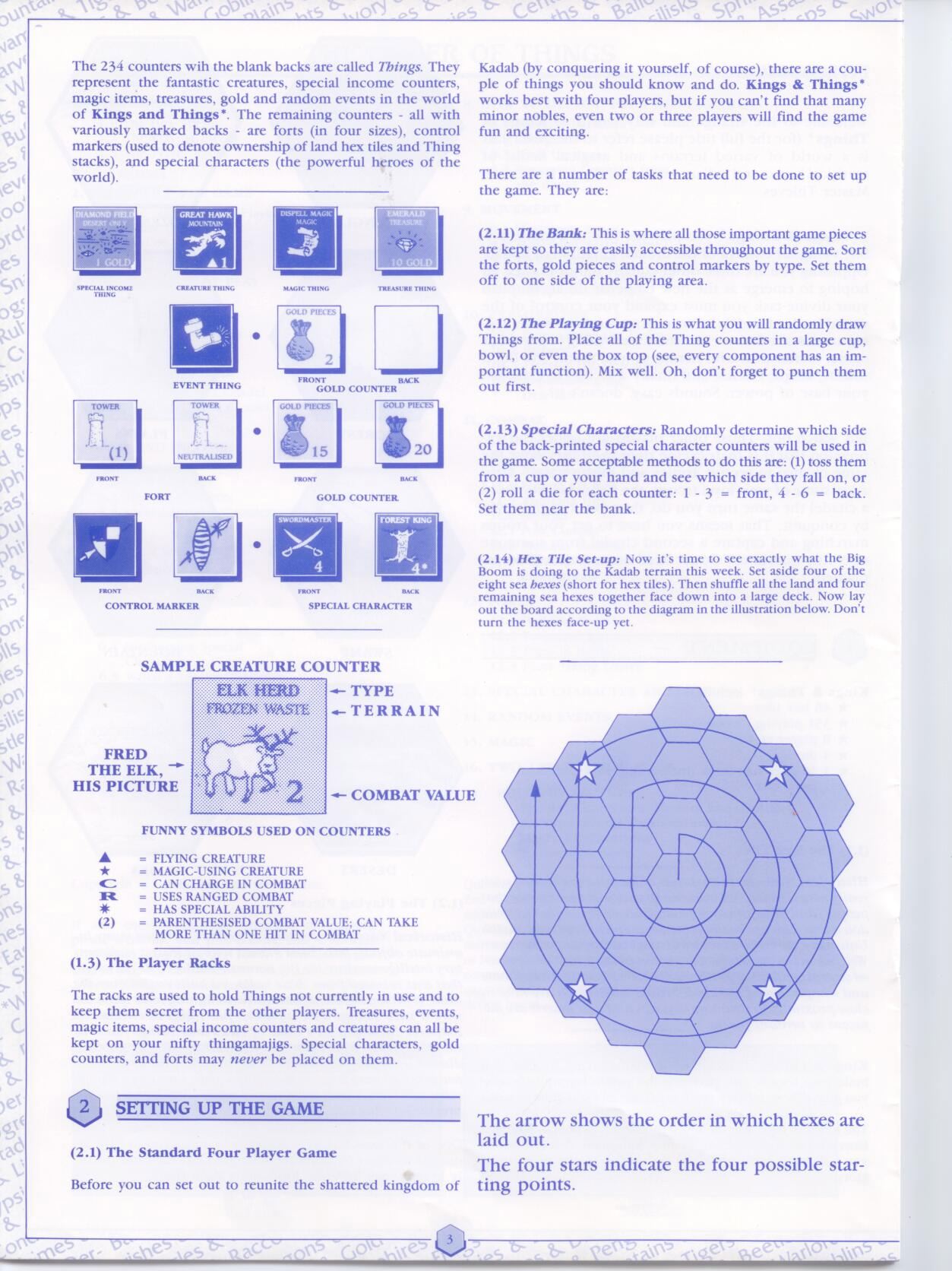
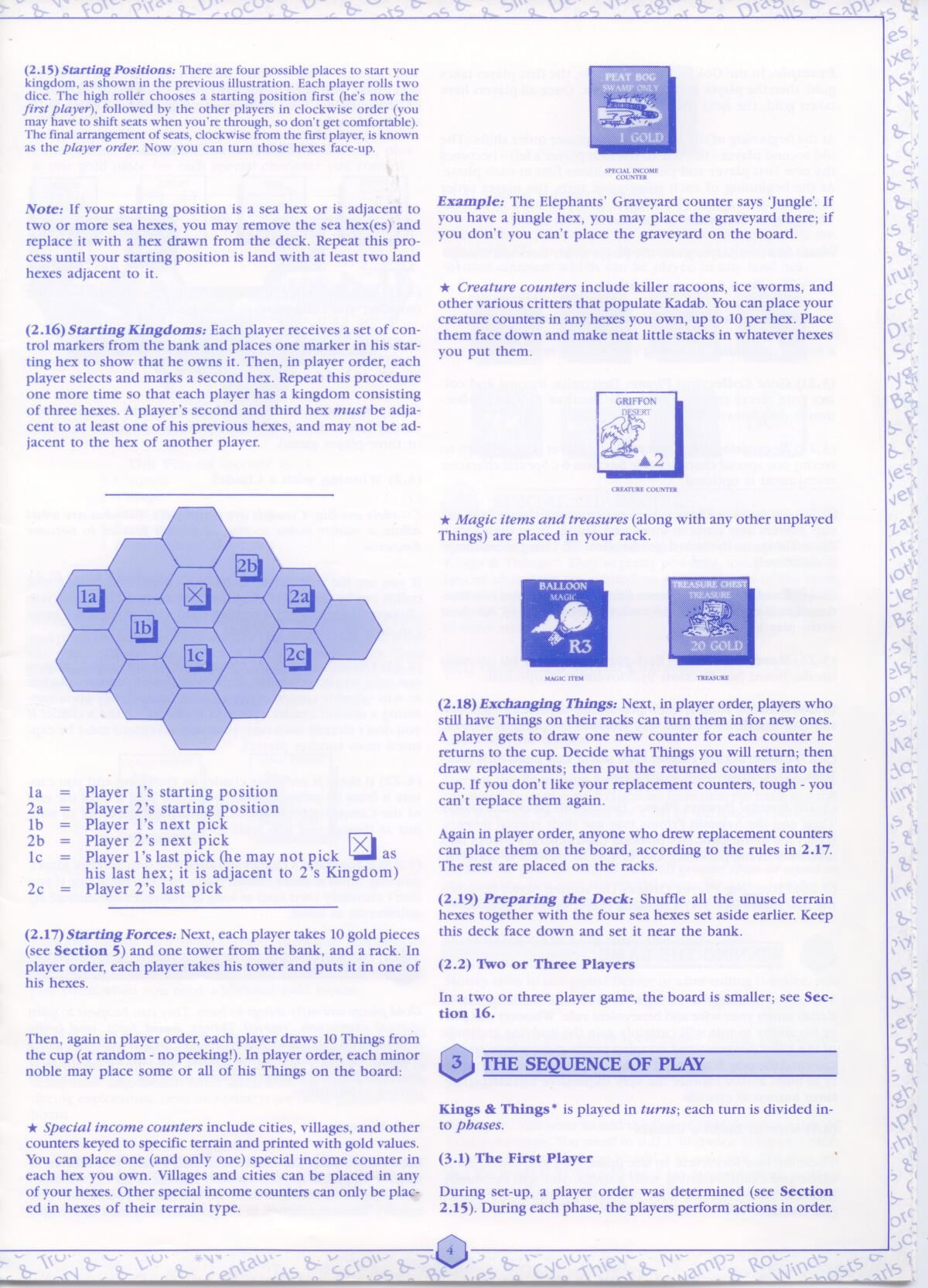
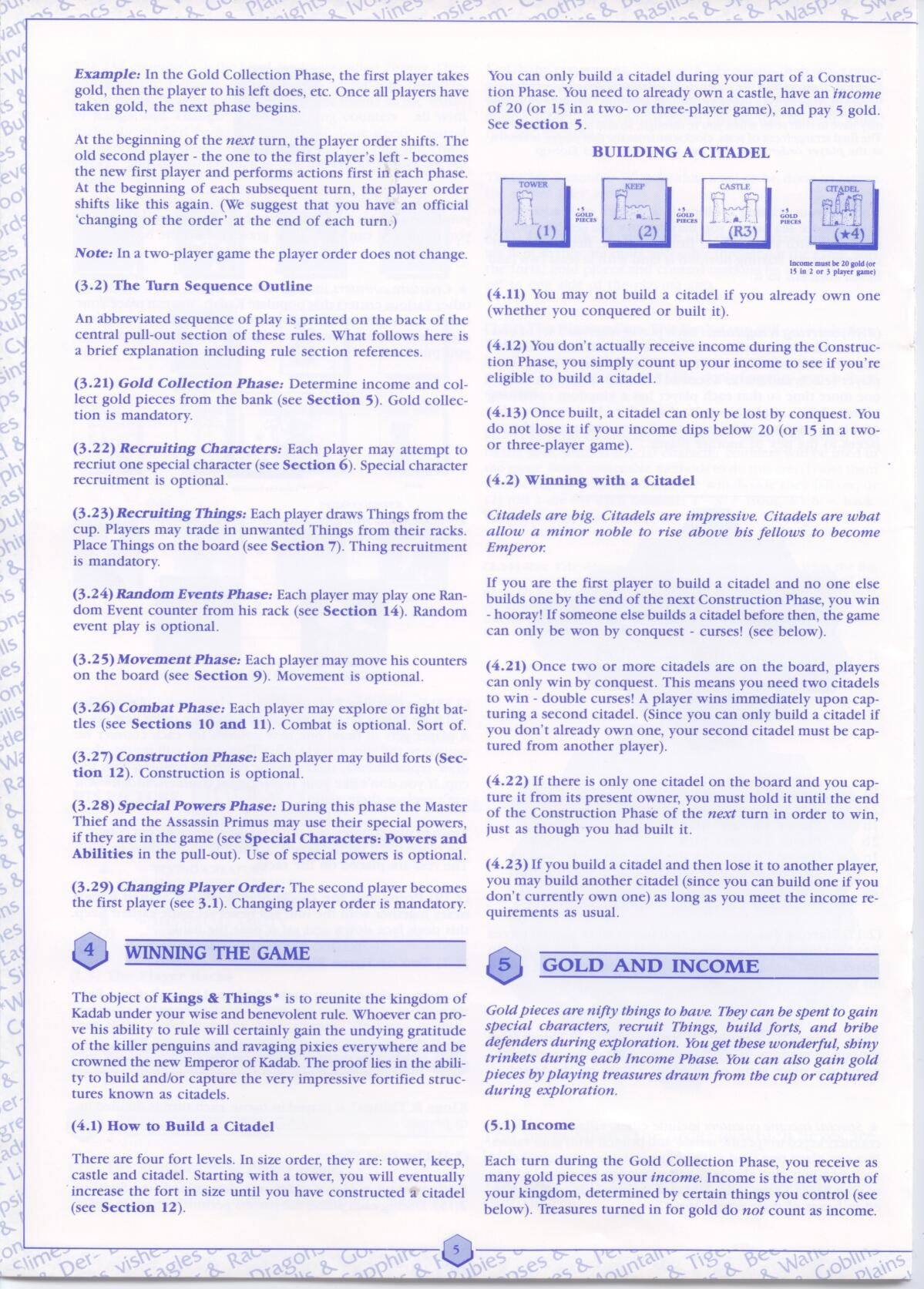
****

GR-01

GR-02

GR-03



****

GR-07

GR-06

GR-05

GR-04

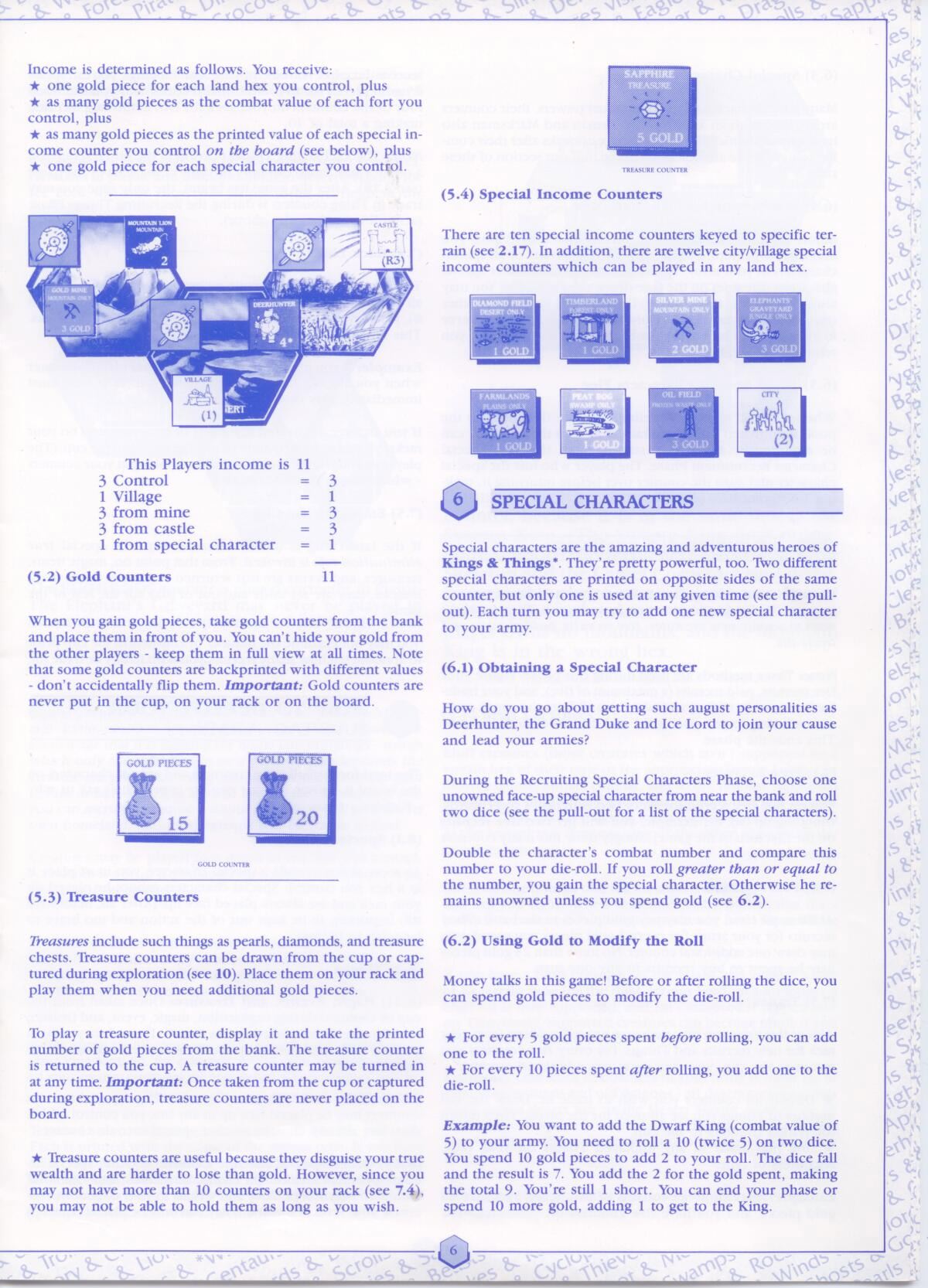
GR-12

GR-11

GR-10

GR-09

GR-08



GR-16

GR-17

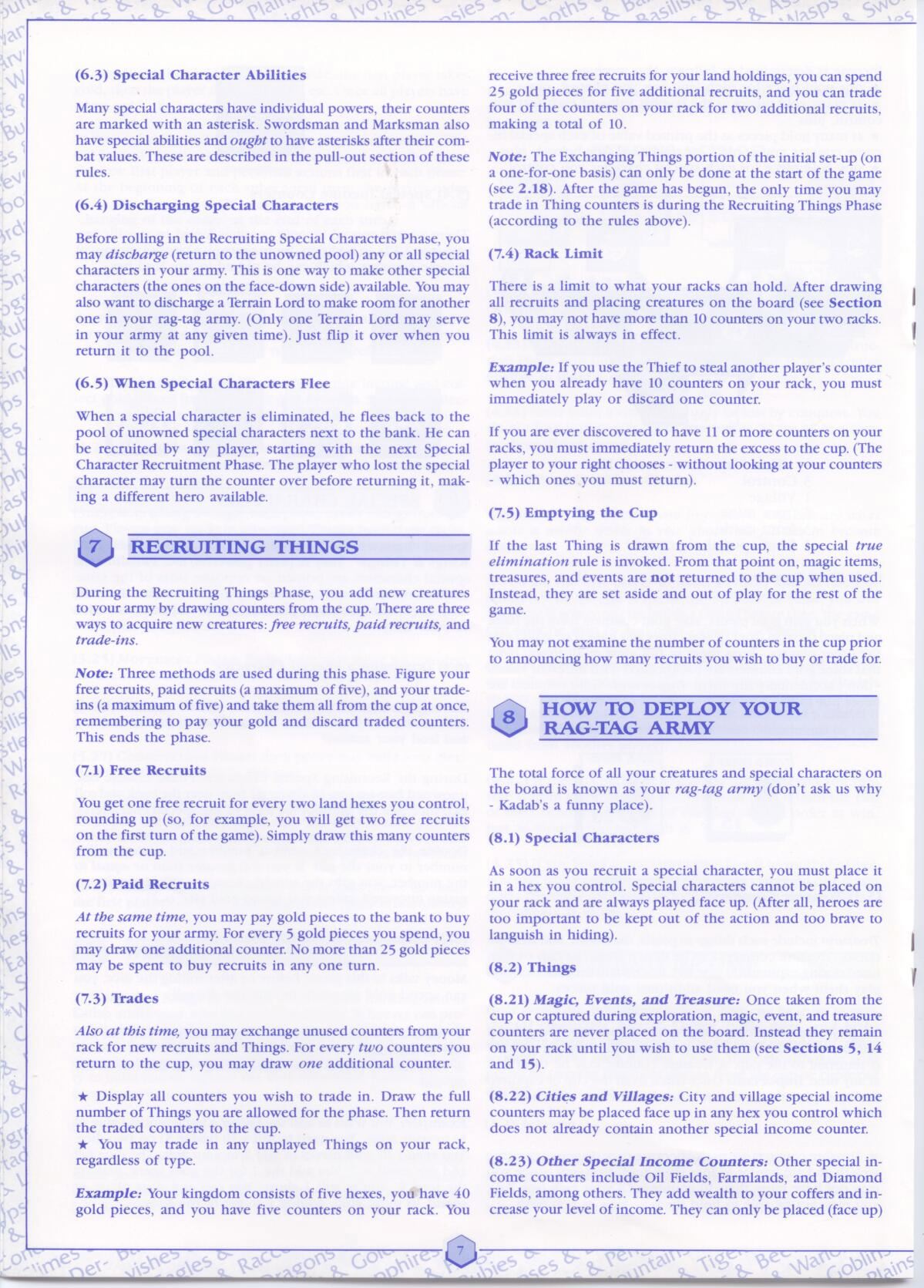
GR-15

GR-14

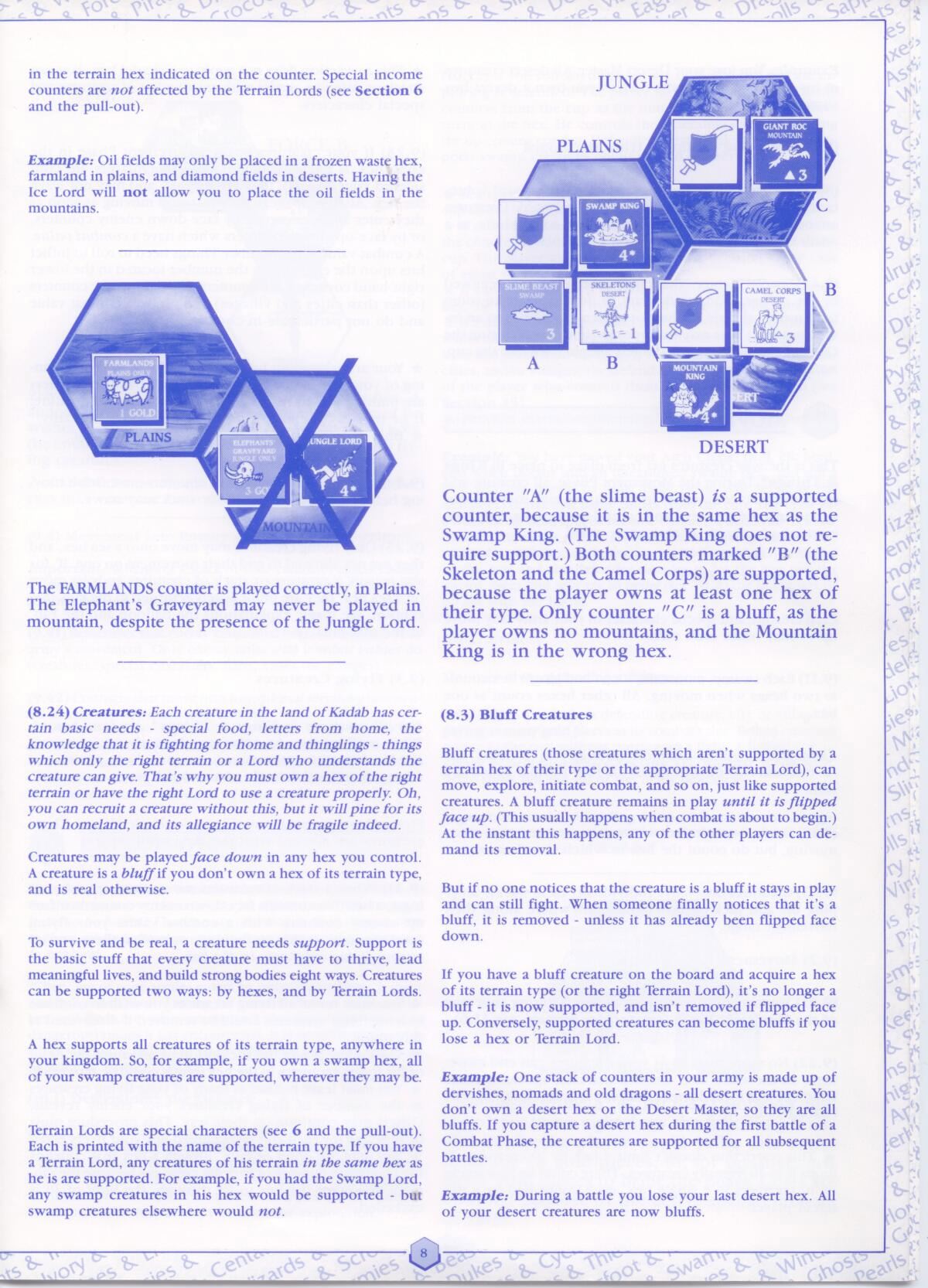
GR-13

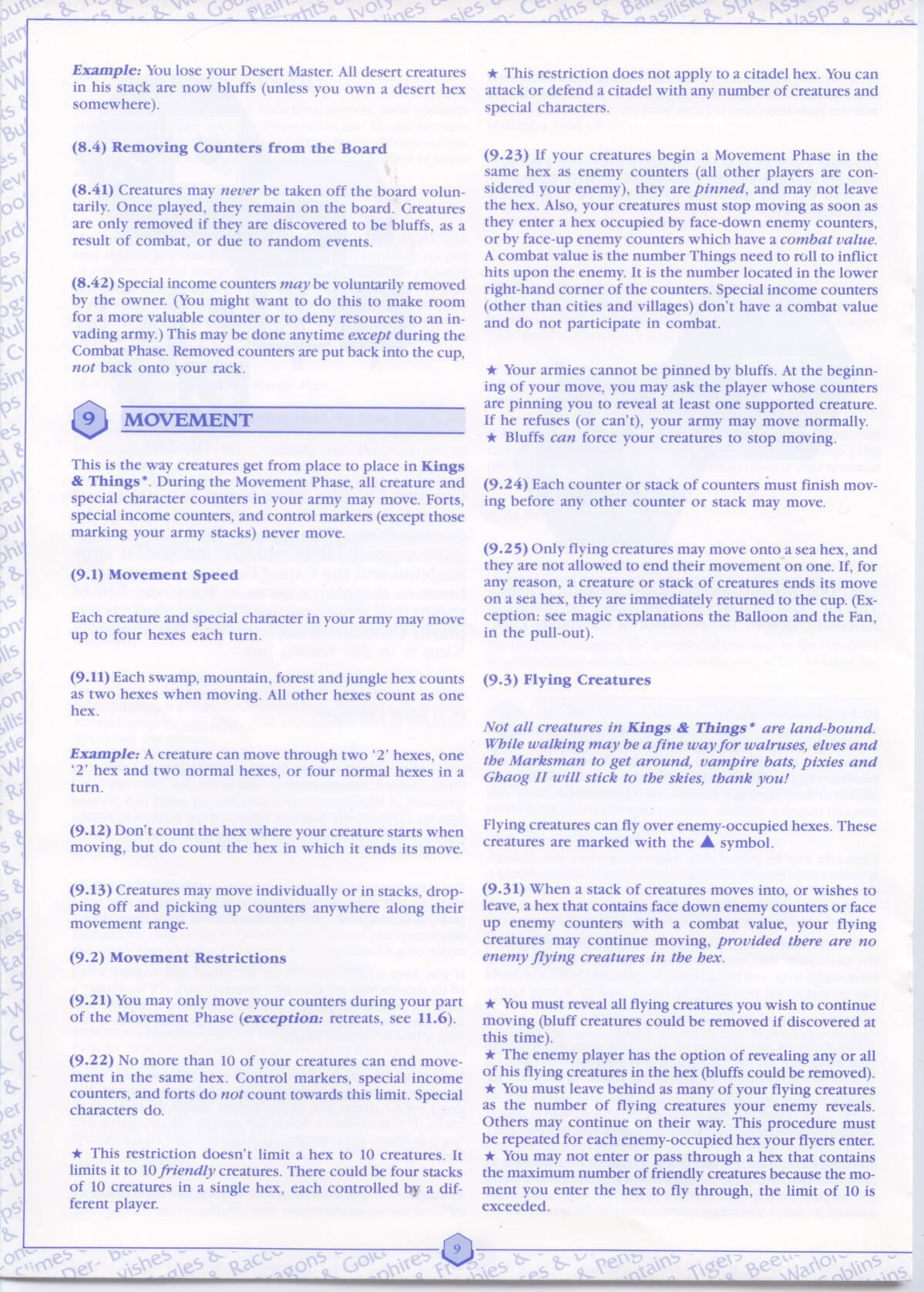
GR-12

GR-11



GR-18



****

GR-22

GR-21

GR-20

GR-19

GR-24

GR-23



GR-30

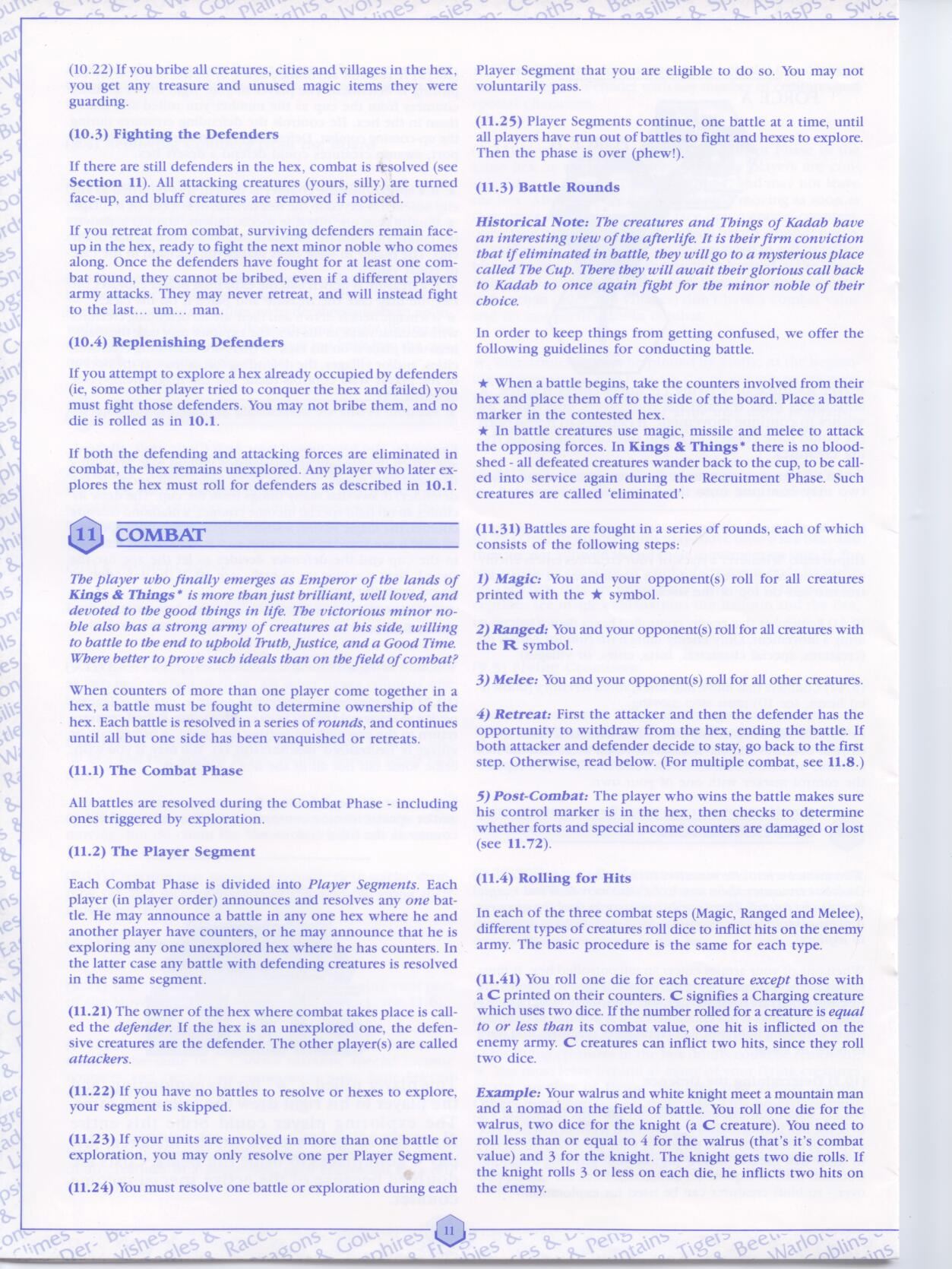
GR-29

GR-28

GR-27

GR-26

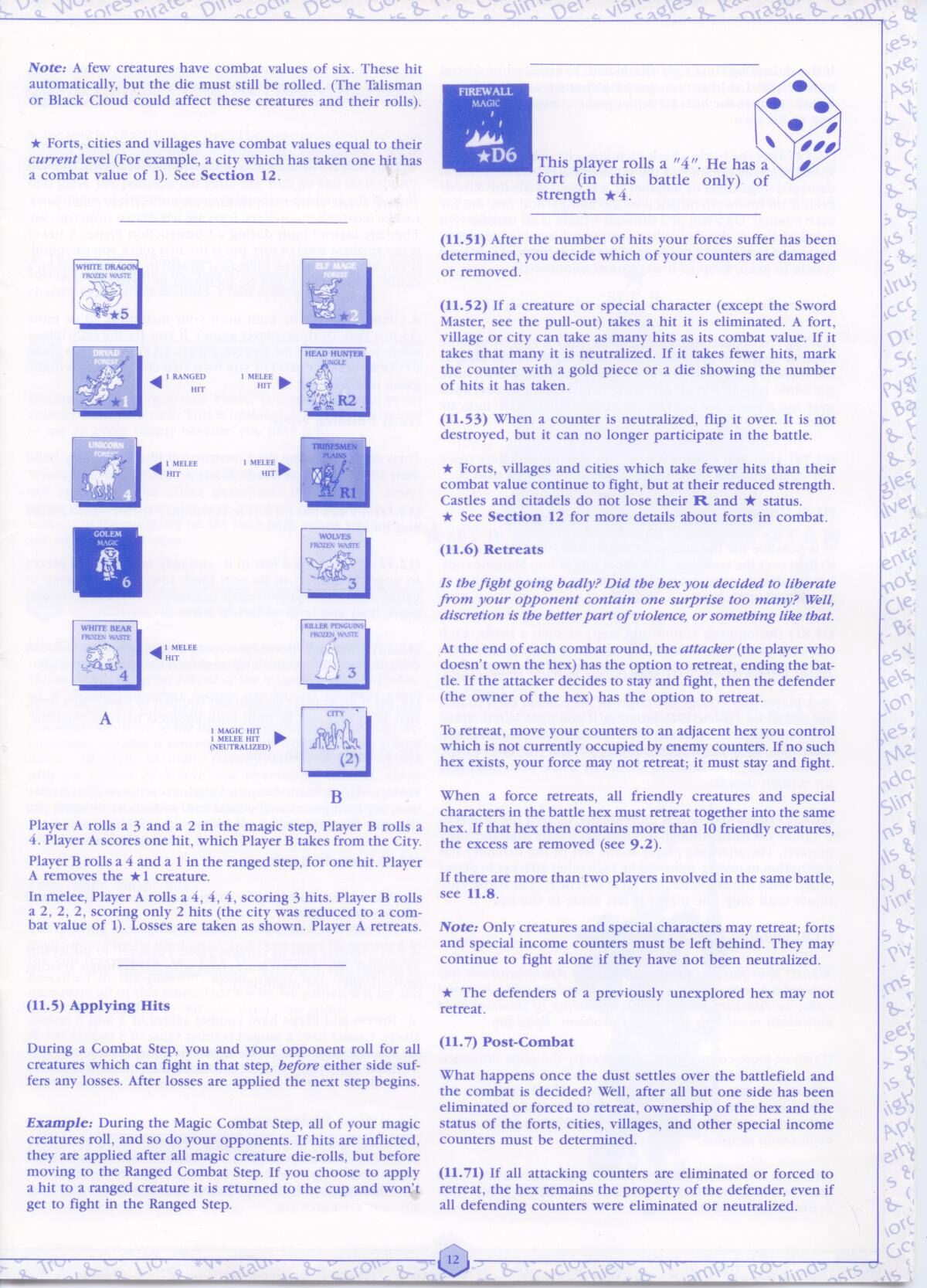
GR-25



GR-33

GR-32

GR-31



GR-38

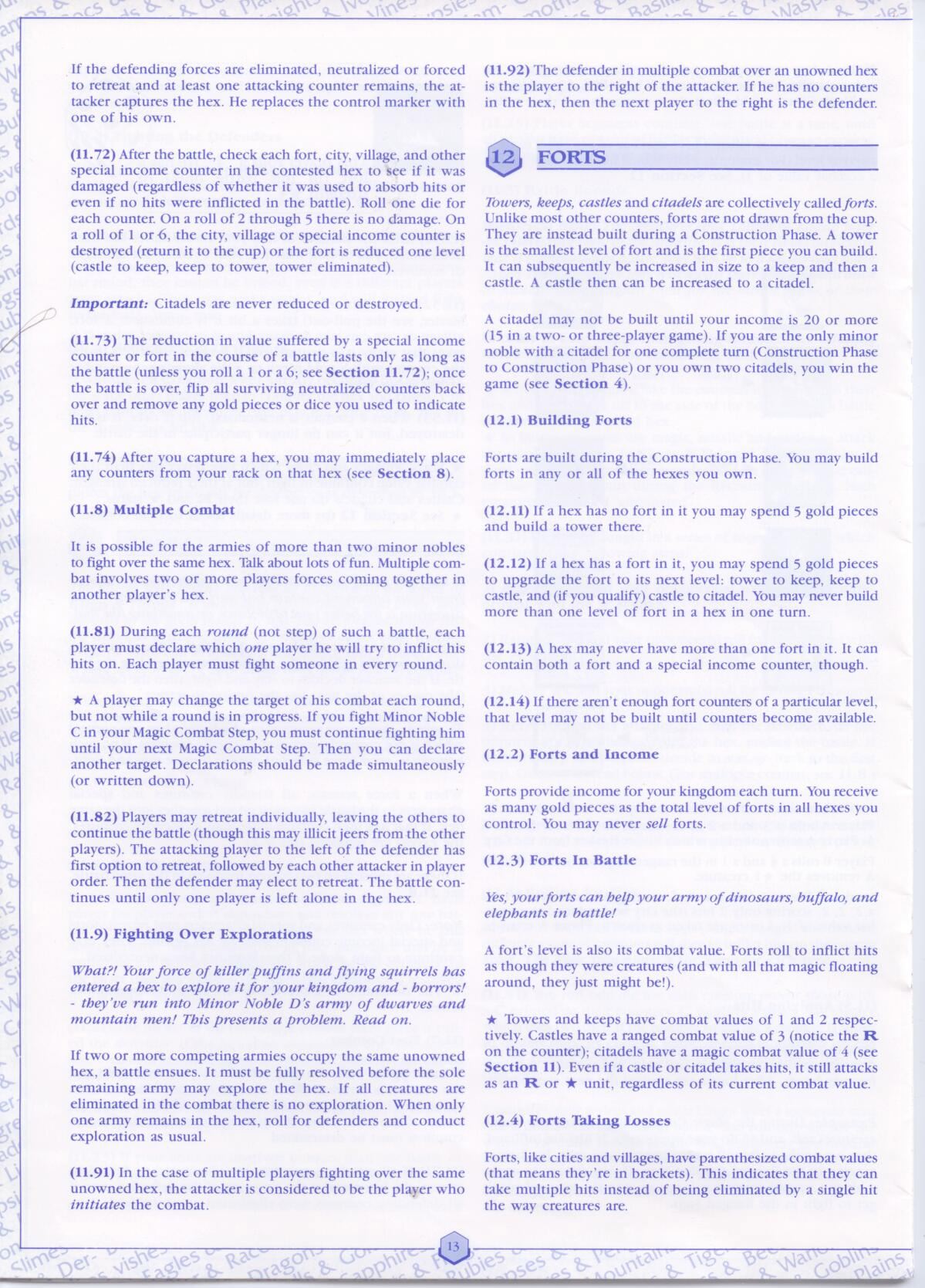
GR-37

GR-39

GR-36

GR-35

GR-34

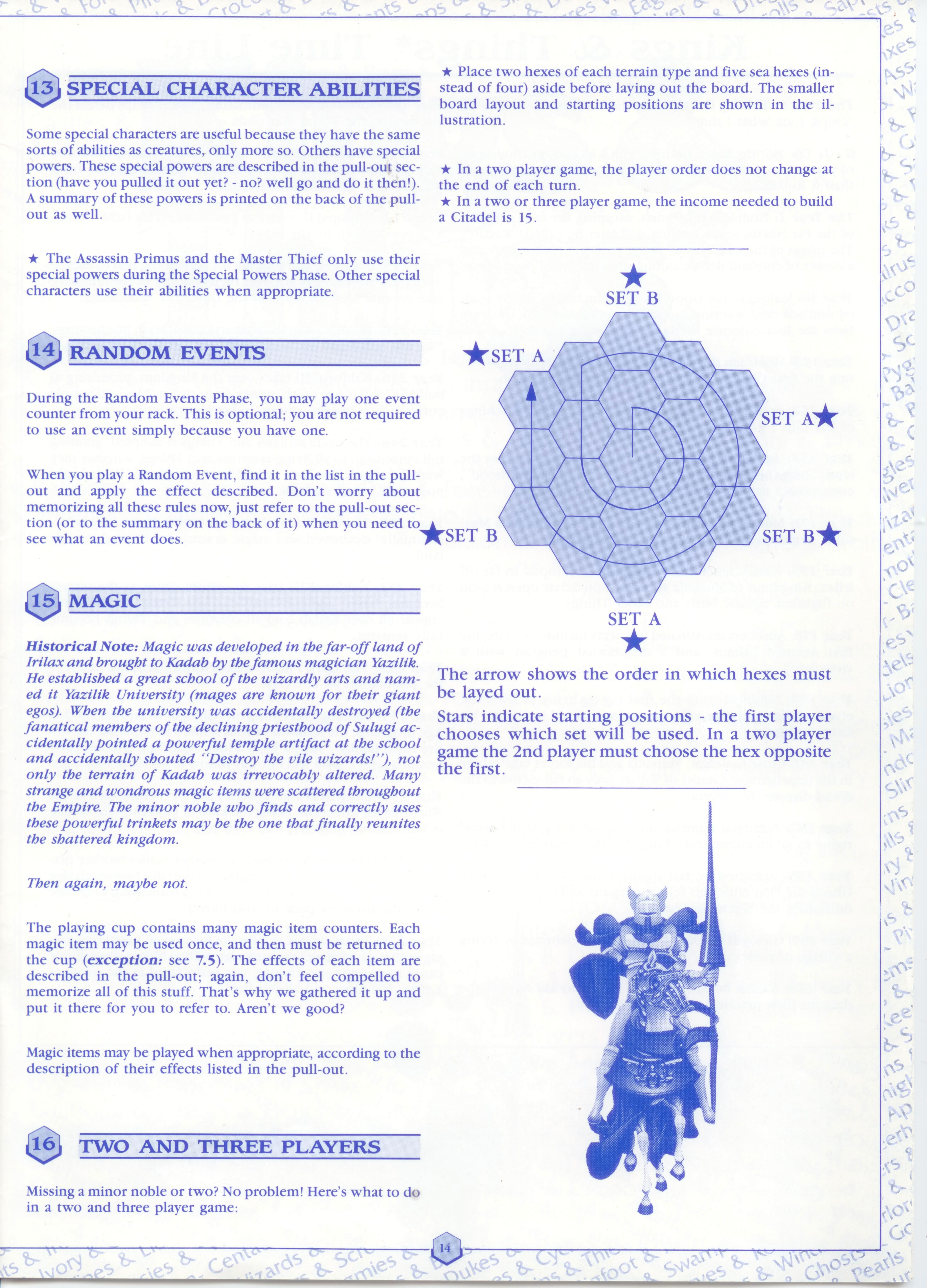


GR-41

GR-42

GR-43

GR-40



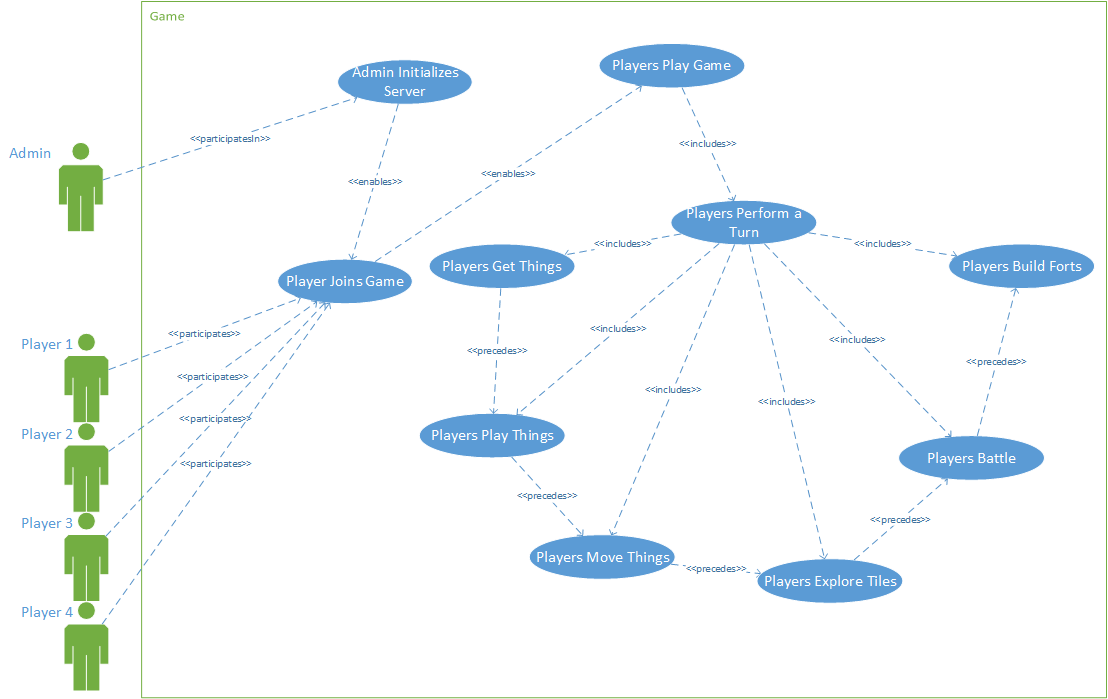
**3.2 – Non Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **ID** | **Non Functional Requirement** | **Traceability** |
| NF-01 | The system determines a winner. |  |
| NF-02 | The system responds to any valid interactions by the player. |  |
| NF-03 | The server acts as an intermediary between players. |  |
| NF-04 | The system ends the combat phase once no combat is left. |  |

**2.3 - Assumptions**

|  |  |  |
| --- | --- | --- |
| **ID** | **Assumption** | **Justification** |
| **A-01** | The game is unplayable if any player is permanently unresponsive. | The game relies on each player taking a turn, so if any player is absent the playing order is broken leaving the game in an unplayable state. |
| **A-02** | A single hex can only be explored by a single player at a time. | The exploration and combat phases are evaluated separately, so having combat occur before exploration is not possible. |
| **A-03** | The player may be able to examine anything that is on the playing board or in their racks at any time. | The player needs the most up to date information to make decisions during each sequence of play. |
| **A-04** | Actions that can be performed during any sequence (treasure claiming and income removal) can only be performed during a player’s turn. | Allowing players to perform actions when it is not their turn would add unnecessary complexity and was not possible with my server model because it is only listening on the socket of the player taking a turn. |
| **A-05** | Things in a tile remain in the order in which they enter the tile (placed or moved). | Considering that creatures are usually faced down during play it is important that the owner knows the order of things for the movement sequence. |
| **A-06** | The player may view a log of personal status updates. | Status updates are being displayed often (usually in response to a server message), so a logging mechanism allows the player to view the history of these updates. |
| **A-07** | Free recruits that would exceed your racks limit (10) are never picked from the playing cup. | Free recruitment cannot be skipped because it is a mandatory phase; therefore we can only pick from the playing cup as many free recruits that can fit in the requesting player’s racks. |
| **A-08** | Citadels can be damaged like any other fort and the single winning condition is that a player is the only one with a citadel at the end of a construction phase. | I felt as though the winning conditions where there is more than one citadel were not entirely clear. My version is certainly a simplification and makes it difficult to win a game once there is more than one citadel built. It would work better for 2 or 3 player games |
| **A-09** | Once a creature is moved into an unexplored hex, it is pinned in that hex for the remainder of the movement phase. | Creatures moving into an unexplored hex have to explore it, and since I have separated out the exploration phase, the creatures need to be pinned so that they will explore the hex after the movement phase has concluded. |
| **A-10** | Sea tiles are treated as regular tiles but are inferior since there is no special income keyed to them. | I do not support flying creatures so I decided not to create the edge case of removing creatures that move onto sea tiles. |

**3.1 - Use Case Diagram**



**3.2 - Use Case Descriptions**

|  |  |
| --- | --- |
| **UC-01** | **Admin Initializes Server** |
| **Description** | This use case describes an Admin setting up the Kings n’ Things game server, so that players can connect and be placed into a games |
| **Actors** | Admin |
| **Triggering Event** | Admin chooses to initialize the Kings N’ Things server |
| **Pre-Condition** | N/A |
| **Main Sequence** | 1. Admin runs the server 2. Admin selects number of players to play in each game 3. Admin begins hosting games of Kings n’ Things for 2, 3, or 4 Players |
| **Post-Condition** | Game server is running |
| **Resulting Event** | The game server is now awaiting connections from players |
| **Alternative Scenarios** | N/A |
| **Traceability** | GR-01, GR-02, GR-40  NF-03 |

|  |  |
| --- | --- |
| **UC-02** | **Player Joins Game** |
| **Description** | This use case describes a player joining a game of Kings n’ Things |
| **Actors** | Player |
| **Triggering Event** | Player starts the application |
| **Pre-Condition** | The admin has setup the server and is waiting for players to connect |
| **Main Sequence** | 1. Player enters lobby 2. Once a certain number of players connect, the server sets up a game 3. The server sends initial startup message to player 4. The game begins |
| **Post-Condition** | The player views the game view |
| **Resulting Event** | The player is now playing a game |
| **Alternative Scenarios** | N/A |
| **Traceability** | N/A |

|  |  |
| --- | --- |
| **UC-03** | **Players Play Game** |
| **Description** | This use case describes Players playing a game of Kings n’ Things |
| **Actors** | 2, 3, or 4 Players |
| **Triggering Event** | Final player has joined the game |
| **Pre-Condition** | 2, 3, or 4 Players have joined a Game |
| **Main Sequence** | 1. Players roll for player order 2. Winner selects starting position 3. Players claim a hex in player order (repeated once) 4. Players are given their initial gold 5. Players place their initial tower in player order 6. All Players receive initial things 7. Players play initial things in player order 8. Repeatedly:    1. Players Perform a Turn of Kings N’ Things    2. Game determines if a Player has won       1. If so, Player wins       2. else loop back to 8a |
| **Post-Condition** | Game has declared a winner |
| **Resulting Event** | Game ends |
| **Alternative Scenarios** | N/A |
| **Traceability** | NF-02  A-01, A-03, A-04, A-06 |

|  |  |
| --- | --- |
| **UC-04** | **Players Perform a Turn** |
| **Description** | This use case describes players playing a turn within a game of Kings n’ Things |
| **Actors** | 2, 3, or 4 Players |
| **Triggering Event** | First turn OR previous turn has ended without a winner |
| **Pre-Condition** | Player order is set |
| **Main Sequence** | 1. Players get things 2. Players play things 3. Players move things 4. Players explore tiles 5. Players battle 6. Players construct Forts |
| **Post-Condition** | All players have played another turn |
| **Resulting Event** | Players are ready to play another turn |
| **Alternative Scenarios** | N/A |
| **Traceability** | GR-03, GR-04, GR-19  A-01, A-04 |

|  |  |
| --- | --- |
| **UC-05** | **Players get things** |
| **Description** | This use case describes the steps for the players to get things |
| **Actors** | 2, 3, or 4 Players |
| **Triggering Event** | Players begin a new turn |
| **Pre-Condition** | New turn has begun |
| **Main Sequence** | 1. In play order, each Player collects gold based on his/her income 2. In play order, each Player obtains things from “the cup” based on the number of hex tiles they control and free space in their racks 3. In play order, each Player can optionally recruit things at the cost of gold |
| **Post-Condition** | Players have received their things |
| **Resulting Event** | Players are ready to play things |
| **Alternative Scenarios** | N/A |
| **Traceability** | GR-07, GR-09, GR-11, GR-12, GR-14  A-07 |

|  |  |
| --- | --- |
| **UC-06** | **Players Play Things** |
| **Description** | This use case describes the steps for the players to play things |
| **Actors** | 2, 3, or 4 Players |
| **Triggering Event** | Players are ready to play things |
| **Pre-Condition** | N/A |
| **Main Sequence** | 1. In player order, each player takes things from his/her rack and places them on any tile s/he owns |
| **Post-Condition** | Player’s Things have been played to their tiles |
| **Resulting Event** | Players are ready to move things |
| **Alternative Scenarios** | N/A |
| **Traceability** | GR-17  A-10, A-05 |

|  |  |
| --- | --- |
| **UC-07** | **Players Move Things** |
| **Description** | This use case describes Players moving their things |
| **Actors** | 2, 3, or 4 Players |
| **Triggering Event** | Players are ready to move things |
| **Pre-Condition** | Things are in play |
| **Main Sequence** | In playing order, in each hex tile s/he owns:   1. The current player selects up to 10 things to move from any tiles s/he owns 2. The current player attempts to move the selected things by selecting a destination 3. Game carries out the attempted move to the selected hex |
| **Post-Condition** | Things have been moved |
| **Resulting Event** | Players are ready to explore tiles |
| **Alternative Scenarios** | Alternative 1: The destination is an unexplored tile  A3.1 Game carries out movement  A3.2 Game marks the destination for exploration  Alternative 2 : The destination is occupied by another player  B3.1 Game carries out movement  B3.2 Game marks the destination for combat  Alternative 3: The destination is owned by another player but not occupied  C3.1 Game carries out movement  C3.2 Destination is claimed by player  Alternative 4: The selected hex is an invalid destination  D3.1 Game does not carry out movement |
| **Traceability** | GR-20, GR-21  A-10, A-09, A-05 |

|  |  |
| --- | --- |
| **UC-08** | **Players Explore Tiles** |
| **Description** | This use case describes Players exploring tiles |
| **Actors** | 2, 3, or 4 Players |
| **Triggering Event** | Players are ready to explore tiles |
| **Pre-Condition** | Things have been moved into unexplored tiles |
| **Main Sequence** | In playing order, repeat until there are no more tiles to explore:   1. The current player selects a tile that is marked for exploration 2. The current player rolls a die to explore the tile 3. The exploration marker is removed 4. The tile is defended by the player to the right of the current player and marked for combat |
| **Post-Condition** | No exploration markers are left |
| **Resulting Event** | Players are ready to battle |
| **Alternative Scenarios** | Alternative 1: No combatable thing is picked for defense  A4.1 The tile is claimed by the current player  A4.2 Any treasure picked is added to the current player’s gold count  Alternative 2: Player rolls a 1 or 6  B4.1 The tile is claimed by the current player |
| **Traceability** | GR-23  A-02, A-09 |

|  |  |
| --- | --- |
| **UC-09** | **Players battle** |
| **Description** | This use case describes players resolving battles |
| **Actors** | 2, 3, or 4 Players |
| **Triggering Event** | Combat markers have been placed on the game |
| **Pre-Condition** | N/A |
| **Main Sequence** | In *each* hex tile with a combat marker, repeat the following steps for each player until the game declares the current combat to be over:   1. The current player selects a tile where s/he is fighting and the battle begins 2. The game asks each participant to select another participant of this battle as a target. 3. Each participant rolls for each of his/her things involved in the current round of combat to determine how many hits s/he inflicts to his/her selected target. 4. For each hit received by a participant, the game asks that participant to select a thing to discard for its army involved in that combat, which are discarded instantly. 5. The game declares combat to be over in this tile if none or a single participant is left in this tile, and the combat marker is removed from this tile 6. The winner of the battle claims the tile |
| **Post-Condition** | No combat markers are left |
| **Resulting Event** | Players are ready for construction |
| **Alternative Scenarios** | Alternative 1: The battle had no victor  6.1 The tile’s owner remains unchanged |
| **Traceability** | GR-27 to GR-33  NF-04  A-08 |

|  |  |
| --- | --- |
| **UC-10** | **Players Build Forts** |
| **Description** | This use case describes players building forts |
| **Actors** | 2, 3, or 4 Players |
| **Triggering Event** | Combat has finished |
| **Pre-Condition** | N/A |
| **Main Sequence** | 1. In player order, each player either places a tower or upgrades an existing fort in any tile s/he owns |
| **Post-Condition** | End of turn cycle |
| **Resulting Event** | N/A |
| **Alternative Scenarios** | N/A |
| **Traceability** | GR-36 to GR-39 |

|  |  |
| --- | --- |
| **UC-11** | **Players Wins** |
| **Description** | This use case describes the result of a player winning the game |
| **Actors** | 2, 3, or 4 Players |
| **Triggering Event** | The game has determined a winner |
| **Pre-Condition** | N/A |
| **Main Sequence** | 1. The game notifies the winner and loser(s) 2. Players restart the game |
| **Post-Condition** | Players play another game |
| **Resulting Event** | The game is over |
| **Alternative Scenarios** | N/A |
| **Traceability** | GR-05, GR-06  NF-01  A-08 |

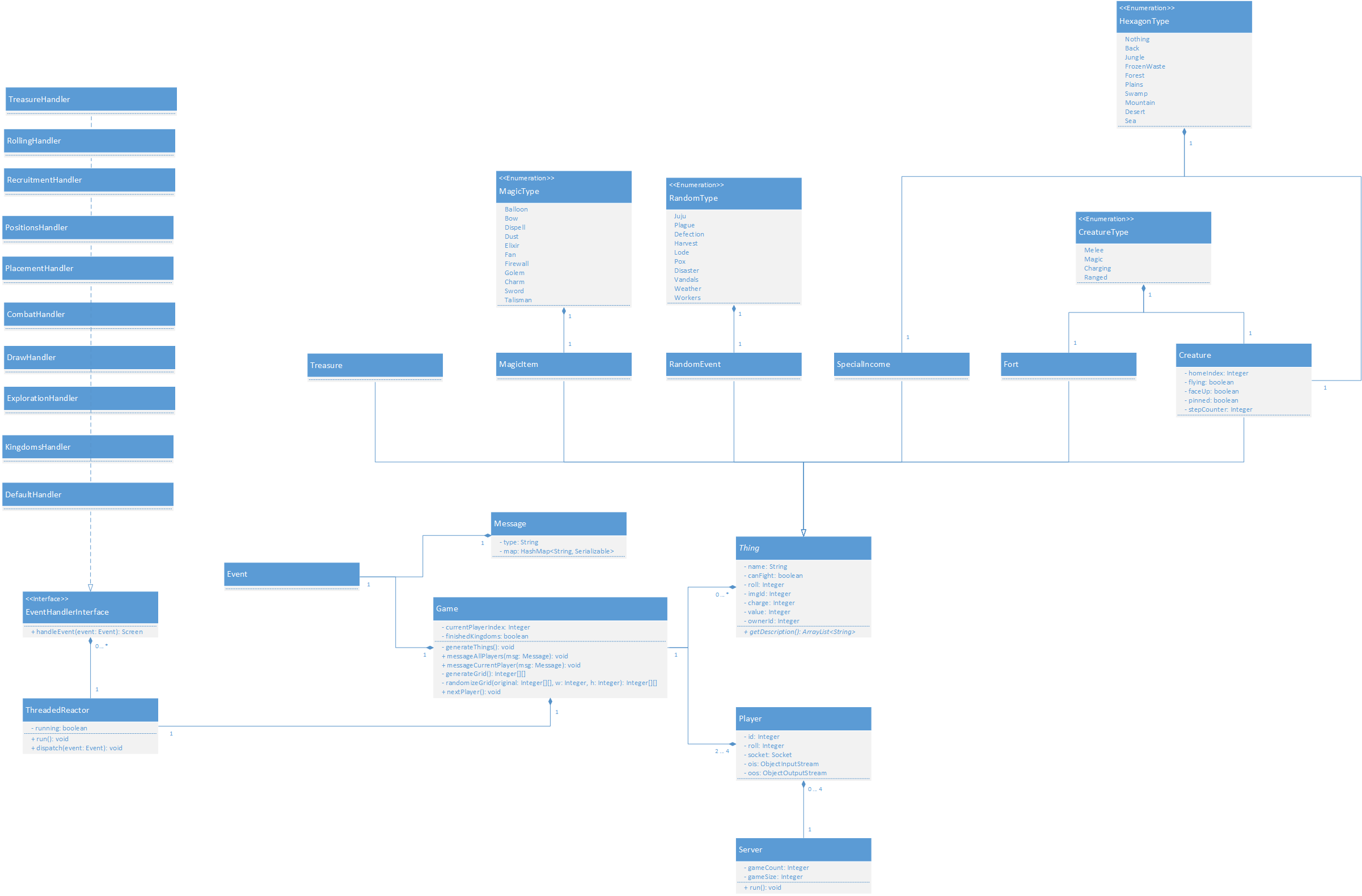
**4.1 - Responsibilities**

|  |  |  |
| --- | --- | --- |
| **ID** | **Responsibility** | **Use Case** |
| **R-1** | The system accepts connections from players. |  |
| **R-2** | The system spawns a new game for players. |  |
| **R-3** | The system randomizes the playing grid. |  |
| **R-4** | The system sends the start message to all players. |  |
| **R-5** | The system sets up the grid. |  |
| **R-7** | The players roll two dice. |  |
| **R-8** | The system moves the highest roller to the front of the turn order. |  |
| **R-10** | The player selects a valid starting position. |  |
| **R-11** | The system determines appropriate player positions. |  |
|  | The player selects two valid tiles to claim. |  |
|  | The system sets the ownership of the selected tiles. |  |
|  | The player selects an owned tile to place a tower within. |  |
|  | The system places a tower in the selected tile. |  |
|  | The system adds 10 to the player’s gold counter. |  |
|  | The system randomly selects 10 things from the playing cup and displays them in the player’s racks. |  |
|  |  |  |

**5.1 - Client Class Diagram**



**5.2 - Server Class Diagram**



**6 - Design Decisions**

|  |  |
| --- | --- |
| ID | Design Decision |
| DD-01 | **MVC - Client**  The architecture used for the client systems is a variation of the model view controller. The primary difference is that the model does not notify the view when it has changed but instead a render loop simply calls an update method in each active view for each touch event in an event buffer, which then appropriately modifies the model. The views also have a present method which is called constantly so that the most recent changes to the model are visible to the user, since we never really know when a modification to the model will be made. This design provided lots of flexibility when creating views allowing me to quickly create the appropriate view for each specific sequence and effectively encapsulate its functionality. |
| DD-02 | **Rector Pattern**  All incoming messages on both the server and client are handled through an implementation of the reactor pattern running in its own thread. The reactor pattern effectively separates the processing of events from the actual functionality of the game allowing me to write modular event handling components. |
| DD-03 | **Client Side Reactor**  The reactor on the client differs from a standard reactor in that messages received are placed on an event queue so that they can be evaluated at the top of the render loop on the main thread rather than when the message is received on the reactor thread. The reason for doing this is that state is modified from the handlers that are dispatched by the reactor and so by doing everything on the main thread I side stepped any concurrency issues that could have come up. This was not a problem on the server because almost all state manipulations are facilitated through the reactor thread so there is very little concurrency within each individual game. |
| DD-04 | **Client to Server Model – Centralized Control**  All messages that are sent from a client travel through the server, which gives the server host complete control over what is being sent to each player in the relay message(s). This allows the host to provide modifiers so that when particular messages are received (such as rolls); they can be modified before being relayed out to the players. The host could also send commands (such as shutdown) to all players at any point. Debugging the networking component was simplified since all incoming and outgoing messages can be logged directly on the server. |
| DD-05 | **Client to Server Model – Separation of State**  In the game there are certain things, such as the playing cup, for which there should be one and only one copy over the entire distributed game. The client to server model lends itself very well to this concept, because these things can simply exist on the server side which allows all players to interact with them while maintaining only one copy. |
| DD-06 | **Networking – Listening on One Socket**  The server listens on only one socket (current player’s socket) for each game that is running. I could have spawned a reactor for each player but listening on only one socket simplifies messaging functionality by not doing concurrent message handling per game. The primary reason for implementing the game server this way is that I feel as though it complies with how a board game is played in real life. Obviously players take turns but the main idea is that only one player should be performing actions at a time otherwise it would become difficult to keep track of what every player is doing. |
| DD-07 | **Networking - Locking**  Several sequences allow players to perform multiple actions before ending their turns. Each time a player performs an action a lock is set so that the user cannot do any more actions until a server response is received. The reason for doing this is that to determine if an action is valid the system does many checks on data that could potentially be stale if the previous action has not been processed by the server yet. The player still has the ability to do anything else within the application that does not involve sending the server messages, such as view racks, hex contents, stats. |
| DD-08 | **UI Component Encapsulation**  Due to the limited screen size of mobile devices I made the decision to create sub-screens which can be toggled on or off and can obviously only be displayed one at a time. These screens include the viewer for the racks, the viewer for any selected hex, and a page for stats (player gold count / status log). There is also a combat sub screen which is only toggled during the combat sequence (by the system). Creating these sub screens allowed me to encapsulate the functionalities of the varied UI components within a relatively small area. |

**General Notes**

Below is a short list of notes that I was not entirely sure where to fit in the documentation.

**Movement**

The generated path is always the shortest distance between two tiles but since there are usually multiple shortest paths, it is done arbitrarily by the system, which means it does not take into account the concept of "cheapest" or "best" path. The only way players can have control over the path is by moving their creature’s one tile at a time.

If the number of creatures in the destination hex reaches 10 (per player) all remaining moving creatures will be placed back one spot on the path to the destination hex. This concept continues to apply for all creatures moving along the path of hexagons.

**Exploration**

If the hex is not defended, any treasure(s) that were drawn are returned to the cup immediately and their combined worth is automatically added to the explorer’s gold count. If multiple special income counters are drawn then the priority is neutral (city over village) over any other type, then the correct type, and then the value. If the hex is not defended then the explorer conquers the hex and no special income gets returned from the drawing.

**Combat**

In battle, Special income (village or city) and forts can take multiple hits and once these hits are applied, they are mostly permanent (upgrading a fort will clear the applied hits). This means that you receive a reduced income if your stuff is damaged. Also with a reduced combat value the thing is less useful in combat (need to roll lower to hit). If the combat value of the special income or the fort is reduced to 0, it is removed from the game board and in the case of special income returned to the cup.