CombatAlrik-Benutzerhandbuch

1. Einführung

CombatAlrik ist eine Kampfverwaltung für DSA. Das Programm verwaltet Kämpfer, Initiative, Aktionen, Waffen, Distanzklassen, Manöver, Licht, Zustände, Wunden, Schaden und längerfristige Handlungen.

Spieler würfeln selbst. Für Nichtspielercharaktere würfelt CombatAlrik. Zusätzliche Würfelbefehle (w6, w20) stehen dem Spielleiter als Hilfsmittel zur Verfügung. CombatAlrik hilft danach beim Verwalten der Kampfsituation und beim Ausführen der Regeln, die aus den eingegebenen Ergebnissen folgen.

CombatAlrik macht insbesondere:

- Kämpfer und Kampfgruppen in eine Kampfliste aufnehmen
- Initiative, Aktionen, Reaktionen und freie Aktionen verwalten
- Nahkampf-, Fernkampf- und Verteidigungsabläufe auswerten
- Distanzklassen `H`, `N`, `S`, `P`, `X` berücksichtigen
- Waffen ziehen, auch Nebenhandwaffen
- Schaden, Rüstung, Trefferzonen und Wunden verwalten
- Lichtverhältnisse und Sichtmodifikatoren anwenden
- längerfristige Aktionen wie Zaubern oder Fernkampfvorbereitung verwalten
- Debug-Ausgaben für Kampfproben anzeigen

CombatAlrik macht nicht:

- Spielerentscheidungen ersetzen
- alle DSA-Sonderregeln automatisch erkennen
- freie Texte mit Leerzeichen in jedem Parserfeld verstehen
- Würfelergebnisse der Spieler automatisch erzeugen, außer bei programminternen Proben

Grundprinzip: Die GUI erzeugt meist Parser-Kommandos. Wer möchte, kann dieselben Kommandos direkt im Eingabefeld des Fensters `Combatalrik` schreiben.

2. Programmfenster

Combatalrik

Das Hauptfenster `Combatalrik` enthält:

- Menüpunkte `Laden` und `Speichern`
- Auswahl für Einzelkämpfer
- Auswahl für Kampfgruppen und Anzahl
- ein Eingabefeld für direkte Kommandos

Typische direkte Eingaben:

```
kr
1 z
1 N w4
a2 N 8 br p2
tp2 6 br
```

Kämpfer

Das Fenster `Kämpfer` zeigt die Kampfübersicht. Dort sieht man die Kämpfernummern, Namen, Lebensenergie, AT/PA beziehungsweise FK, Waffen, Rüstung, Status, Initiative und verfügbare Aktionen.

Die Kämpfernummer ist wichtig: Fast alle Kommandos beginnen mit dieser Nummer oder beziehen sich auf sie.

Status

Das Fenster `Status` zeigt Textausgaben. Dort erscheinen Ergebnisse wie:

```
Söldner zieht seine Waffe.
Söldner attackiert erfolgreich mit dem Säbel!
Piratenkapitän pariert nicht!
```

Auch Fehlermeldungen und Debug-Ausgaben erscheinen dort.

Controller

Der `Controller` bietet Eingabefelder und Auswahlboxen für häufige Aktionen. Er erzeugt Parser-Kommandos, zum Beispiel für Nahkampf, Fernkampf, langfristige Aktionen, Verteidigung, Schaden und Licht.

GUI und Parser sind zwei Wege zum selben Ziel: Wenn die GUI eine Attacke auslöst, wird intern ein Kommando wie `1 N w4` oder `a2 N 8 br p2` erzeugt.

3. Erster Kampf in fünf Minuten

Ein einfacher Ablauf:

1. Im Hauptfenster einen Einzelkämpfer auswählen und `Hinzufügen` drücken.
2. Weitere Kämpfer oder eine Kampfgruppe hinzufügen.
3. Mit `bereit` oder `b` NSC bereit machen. Dabei ziehen NSC ihre Waffen und eine neue Kampfrunde beginnt.
4. Einen Kämpfer angreifen lassen.
5. Eingehende Angriffe mit `a# ...` abwickeln.
6. Schaden mit `tp#` oder `sp#` eintragen, falls nötig.
7. Mit `kr` die nächste Kampfrunde starten.

Beispiel:

```
b
1 z
1 N
a2 N 8 br p2
tp2 6 br
kr
```

Wenn ein Angriff ohne Manöver ausgeführt werden soll:

```
1 N
```

Wenn Kämpfer 2 einen eingehenden Treffer in der Brust mit normaler Parade abwehren soll:

```
a2 N 8 br p
```

Ohne Verteidigungstoken verwendet der Parser den normalen Paradepfad:

```
a2 N 8 br
```

4. Kämpfer verwalten

Über die GUI

Einzelkämpfer werden im Hauptfenster aus der Liste `Einzelkämpfer` gewählt und mit `Hinzufügen` eingefügt.

Kampfgruppen werden im Bereich `Kampfgruppen` gewählt. Daneben wird die Anzahl eingestellt. `Hinzufügen` fügt die Gruppe hinzu.

Direktes Kommando add

Mit `add` wird ein Kämpfer mit freiem Namen und fester Initiative erzeugt:

```
add Alrik 14
add Boromea 18
add Gegner 12 bk2
```

Syntax:

```
add <name> <ini> [extra]
```

`extra` ist optional und wird für zusätzliche Kampfstile wie `bk2`, `sk2` oder `pw2` verwendet, wenn der Kämpfer passend angelegt ist.

Gruppen

Gruppen werden über die GUI hinzugefügt. Es gibt kein direktes Parserkommando für `add_group`. Die Gruppendaten kommen aus den Kampfgruppendateien.

Laden und Speichern

Laden und Speichern laufen über das Menü `Datei` im Hauptfenster:

- `Laden`
- `Speichern`

Spielstände werden als CombatAlrik-JSON-Dateien gespeichert. Alte Pickle-Spielstände werden vom aktuellen Ladepfad nicht mehr unterstützt.

5. Initiative

Die Initiative steht in der Kampfübersicht. CombatAlrik unterscheidet:

- `initiative`: aktueller Initiativewert
- `actionini`: Initiativeposition für Aktionen innerhalb der Runde
- Aktionen, Reaktionen und freie Aktionen

Direkte Änderung:

```
ini1 +4
ini1 -3
```

Syntax:

```
ini# <wert>
```

Beispiele:

```
ini2 -4
ini3 +2
```

Freie Aktionen

Humanoide Kämpfer erhalten bei positiver Initiative ihre normalen Aktionen und freien Aktionen. Zusätzlich gibt es Initiative-Schwellen.

Initiative-Schwellen

Bei Initiative 21, 31, 41, 51, ... erhält ein humanoider Kämpfer pro Schwelle:

- eine zusätzliche freie Aktion
- einen Punkt Verteidigungserleichterung über `PAinimod`

Beispiele:

```
INI 20 -> kein Schwellenbonus
INI 21 -> +1 freie Aktion, +1 PAinimod
INI 31 -> +2 freie Aktionen, +2 PAinimod
INI 41 -> +3 freie Aktionen, +3 PAinimod
INI 51 -> +4 freie Aktionen, +4 PAinimod
```

Die Berechnung erfolgt beim Aktualisieren der Aktionen, insbesondere zu Beginn einer neuen Kampfrunde. Eine Initiativeänderung mitten in der Runde verändert nicht rückwirkend bereits verbrauchte Aktionen.

Neue Kampfrunde:

```
kr
```

6. Waffen und Ziehen

Waffen liegen intern in mehreren Slots:

- `right`: rechte Hand
- `left`: linke Hand
- `rightholster`: rechte Scheide
- `leftholster`: linke Scheide
- `side`: Gürtel oder Ersatzwaffe

Standard: z zieht die rechte Waffe.

```
1 z
1 ziehen
```

Linke Hand:

```
1 z la
1 ziehen la
```

Beide Hände:

```
1 z beide
1 ziehen beide
```

Mit Ziehdauer:

```
1 z 2
1 z 2 la
1 z 2 beide
1 ziehen 2 beide
```

`z beide` ist eine Bequemlichkeitssyntax. Sie führt zwei normale Ziehvorgänge aus:

```
1 z
1 z la
```

Dadurch bleiben Aktionskosten und `Schnellziehen` unverändert. Das Ziehen beider Waffen wird nicht automatisch kostenlos.

Wenn ein Kämpfer `Schnellziehen` hat, reduziert `do_weapondraw( )` wie bisher die Kosten. `z beide` nutzt diese bestehende Logik zweimal.

7. Distanzklassen

CombatAlrik unterstützt diese Distanzklassen:

- `H`
- `N`
- `S`
- `P`
- `X`

Von nah nach weit:

```
H → N → S → P → X
```

Normale Attacke in Distanzklasse `N`:

```
1 N
```

Attacke in Distanzklasse `X`:

```
1 X
```

Distanzklasse vergrößern:

```
1 dk+
```

Distanzklasse verkleinern:

```
1 dk- +2
```

`dk-` akzeptiert einen optionalen signierten Modifikator mit explizitem Vorzeichen.

8. Angriffe

Nahkampfangriff

Grundform:

```
# [hand] [DK] [manöver]
```

Beispiele:

```
1
1 N
1 ra N
1 la N
1 N w4
1 la N w2,f3
```

`ra` ist die rechte beziehungsweise Haupthand. `la` ist die linke beziehungsweise Nebenhand. Ohne Handangabe verwendet CombatAlrik die Standardhand.

Manöver werden als ein Token geschrieben. Mehrere Manöver werden mit Komma getrennt, ohne Leerzeichen:

```
1 N w4,f2
```

Fernkampfangriff

Fernkampf wird direkt über die Kämpfernummer gestartet.

Im Gegensatz zum Nahkampf wird nach der Kämpfernummer die Entfernung angegeben.

CombatAlrik berücksichtigt dabei Entfernung, Zielgröße, Zielzonen, Schnellschüsse und weitere Modifikatoren.

Typischer Fernkampfablauf

Ein Fernkampfangriff besteht meist aus drei Schritten:

1. Schuss abgeben
2. Treffer auswerten
3. Schaden eintragen

Beispiel:

```
1 20 m
```

CombatAlrik führt den Fernkampfangriff aus.

Falls der Schuss trifft:

```
a2 N 8 br p
```

Misslingt die Verteidigung:

```
tp2 8 br
```

Dadurch werden Rüstung, Wunden und Zustände automatisch berücksichtigt.

Zielgröße

Zielgröße	Direktkommando	Modifikator
winzig	`winzig` oder `w`	+8
sehr klein	`sk`	+6
klein	`klein` oder `k`	+4
mittel	`mittel` oder `m`	+2
groß	`gross` oder `g`	0
sehr groß	`sg`	-2

Die Zielgröße beeinflusst die Trefferwahrscheinlichkeit des Fernkampfangriffs (wie in der Tabelle angegeben). Hinweis: In direkten Parser-Kommandos müssen mehrteilige Größen als Kürzel geschrieben werden. Verwende also `sk` für „sehr klein“ und `sg` für „sehr groß“.

Grundform:

```
# <distanz> [größe] [zone|ansage] [mod] [s]
```

Beispiele:

```
1 20
1 20 m
1 20 m ko
1 20 m 4
1 20 m ko -2 s
```

s steht für Schnellschuss. Größen und Zonen sind Parser-Tokens, zum Beispiel `m`, `ko`, `br`, `ba`.

Komplexes Fernkampfbeispiel

Ein Bogenschütze schießt auf ein mittelgroßes Ziel in 30 Schritt Entfernung.

```
1 30 m ko -2 s
```

Bedeutung:

- 30 Schritt Entfernung
- mittelgroßes Ziel
- Kopf als Zielzone
- zusätzlicher Modifikator -2
- Schnellschuss

Möglicher Ablauf:

```
1 30 m ko -2 s
a2 fk 10 ko a
tp2 10 ko
```

Für eingehende Fernkampftreffer wird die Syntax `a# fk ...` verwendet. Dabei werden keine Distanzklassenmodifikatoren angewendet.

CombatAlrik berechnet anschließend automatisch Rüstung, Wunden und weitere Folgen.

9. Verteidigung

Eingehende Angriffe werden mit a# abgewickelt. Die Nummer hinter a ist der Zielkämpfer.

Grundform:

```
a# [DK] [angriffsmanöver] <schaden> <trefferzone> [verteidigung]
```

Beispiele:

```
a2 N 8 br
a2 N 8 br p
a2 N 8 br p2
a2 N w4 12 br p2
a2 N 10 ko g18
```

Wenn kein Verteidigungstoken angegeben wird, verwendet CombatAlrik den normalen Paradepfad.

Fernkampftreffer

Für eingehende Fernkampftreffer steht eine eigene Syntax zur Verfügung:

```
a# fk <schaden> <trefferzone> [verteidigung] [mod]
```

Beispiele:

```
a2 fk 10 ko
a2 fk 10 ko a
a2 fk 10 ko m +4
a2 fk 10 ko p -2
```

Dabei werden keine Distanzklassenmodifikatoren angewendet.

Normale Parade

```
a2 N 8 br p
```

p steht im Verteidigungsteil für Parade. Ohne Zahl ist der eigene Modifikator `0`.

Meisterparade

```
a2 N 8 br p4
```

p4 bedeutet Parade mit eigenem Meisterparade-Modifikator 4. Misslingt ein Manöver mit eigener Erschwernis (zum Beispiel Meisterparade, Wuchtschlag oder Finte), wird diese als `nextmod` auf die nächste relevante Kampfprobe übertragen.

Gegenhalten

```
a2 N 10 ko g18
```

g18 steht für Gegenhalten mit Vergleichswert 18. Der Wert wird gegen Paradequalität plus Initiative verglichen.

Ausweichen

```
a2 N 8 br a
a2 N 8 br a2
```

a verwendet Ausweichen.

Meisterliches Ausweichen

```
a2 N 8 br m
a2 N 8 br m2
```

`m` steht für gezieltes beziehungsweise meisterliches Ausweichen im Verteidigungstoken.

10. Trefferzonen

Bei direktem Schaden und eingehenden Angriffen werden Trefferzonen als Token angegeben.

Gängige Trefferzonen:

Token	Bedeutung
`ko`	Kopf
`br`	Brust
`ba`	Bauch
`ra`	rechter Arm
`la`	linker Arm
`rb`	rechtes Bein
`lb`	linkes Bein
`ru`	Rücken
`rü`	Rücken

Beispiele:

```
tp1 6 br
sp1 4 ko
a2 N 8 rü p
```

`ru` und `rü` werden für Rücken akzeptiert. Intern verwendet CombatAlrik den Schlüssel `ru`.

Fernkampf-Zielzonen verwenden eine eigene Token-Tabelle. Dort sind unter anderem `kopf`, `ko`, `k`, `brust`, `br`, `bauch`, `ba`, `arm`, `bein`, `hand`, `fuss`, `auge` und `herz` möglich.

11. Manöver

Angriffsmanöver werden im Angriff als kurzer Token angegeben:

Token	Manöver
`w`	Wuchtschlag
`f`	Finte

Token	Manöver
g	Gezielter Stich
s	Sturmangriff
h	Hammerschlag
p	Passierschlag
t	Todesstoß
e	Entwaffnen
n	Niederwerfen
b	Befreiungsschlag
d	Doppelangriff oder Doppelschlag bei Bestien
a	Ausfall

Beispiele:

```
1 N w4
1 N f2
1 N w4,f2
1 la N d
```

Verteidigungsmanöver stehen am Ende eines eingehenden Angriffs:

Token	Verteidigung
p	Parade / Meisterparade
e	Entwaffnen aus Parade
a	Ausweichen
m	Gezieltes Ausweichen
b	Binden
g	Gegenhalten
w	Windmühle

Beispiele:

```
a1 N 8 br p2
a1 N 8 br a
a1 N 8 br m2
a1 N 10 br g18
```

12. Zweihand- und Nebenhandkampf

Nebenhandangriffe verwenden `la`:

```
1 la N
1 la N w2
1 la S f3
```

Die linke Waffe muss gezogen sein:

```
1 z la
1 la N
```

Beide Waffen ziehen:

```
1 z beide
```

Wenn ein Kämpfer eine Zweihandwaffe in der rechten Hand hat, verhindert `do_weapondraw( )` das Ziehen einer linken Waffe.

Beispiel mit rechter und linker Waffe:

```
1 z beide
1 N
1 la N
```

`la` wird auch in der GUI benutzt, wenn die Nebenhand ausgewählt wird.

13. Bestienangriffe

Bestien verwenden nummerierte Angriffsformen. Bei Bestien kann nach der Kämpfernummer eine Zahl als Angriffsform stehen:

```
1 1 N
1 2 N
1 2 N w2
```

Bei normalen humanoiden Kämpfern startet eine Zahl nach der Kämpfernummer dagegen den Fernkampfparser.

Bestien können auch Manöver verwenden, wenn die Angriffsform und die Daten es erlauben:

```
1 2 H
1 3 N d
```

14. Lichtverhältnisse

Licht wird global für den Kampf gesetzt:

```
l normal
l mondlicht
l dunkelheit
licht dichter nebel
```

Benannte Werte:

```
l normal
l sonnenschein
l fackel
l lagerfeuer
l dämmerung
l mondlicht
l dunst
l nebel
l dichter nebel
l undurchdringlicher nebel
l sternenlicht
l dunkelheit
```

Numerische Werte sind ebenfalls erlaubt:

```
l -2
l 0
l +1
l +2
```

Gültig sind `-8` bis `0` sowie `+1` und `+2`.

Die effektive Lichtstrafe wird pro Kämpfer berechnet:

- `Nachtsicht` halbiert den Lichtmodifikator und begrenzt ihn auf höchstens 2.
- `Dämmerungssicht` halbiert den Lichtmodifikator.
- `lichtmod` oder `lightmod` kann als numerische Minderung in den Kämpferdaten oder Mods vorhanden sein.

Beispiele für `lichtmod`:

```
globaler Lichtmodifikator 4, lichtmod 0  -> effektiv 4
globaler Lichtmodifikator 4, lichtmod 2  -> effektiv 2
globaler Lichtmodifikator 4, lichtmod 99 -> effektiv 0
```

Die effektive Lichtstrafe wird nie unter `0` gesenkt.

15. Langfristige Aktionen

Langfristige Aktionen belegen Aktionen über mehrere Ticks beziehungsweise Aktionen.

Syntax:

```
# lang <name> <zeit>
```

Beispiele:

```
1 lang Zauber 4
2 lang Liturgie 6
3 lang Nachladen 2
```

Der Name ist ein einzelnes Token. Leerzeichen im Namen werden nicht unterstützt.

Einige Aktionen erzeugen intern ebenfalls längerfristige Aktionen, zum Beispiel:

- Waffe ziehen mit Dauer größer 1
- Orientieren
- Fernkampfangriffe mit Vorbereitungszeit

Fernkampfwaffen mit Ladezeit können ebenfalls über langfristige Aktionen verwaltet werden.

Beispiel:

```
1 lang Nachladen 2
```

Dadurch wird eine Nachladezeit von zwei Aktionen verfolgt.

16. Direkte Aktionen

Direkte Aktionen wirken ohne den normalen Angriffspfad.

Ausfall beenden

```
q1
aus1
```

`q#` und `aus#` beenden den Ausfall eines Kämpfers.

Entwaffnen aus Parade

```
e1
e1 4
e1 +4
e1 -2
```

`e# [mod]` ruft die Entwaffnen-Logik mit optionalem Modifikator auf.

Initiative ändern

```
ini1 -4
ini1 +2
```

Entfernen

```
rem1
```

`rem#` blendet einen Kämpfer aus der Kampfliste aus.

17. Schaden

CombatAlrik unterscheidet Trefferpunkte `tp` und Schadenspunkte `sp`.

Trefferpunkte

```
tp1 10
tp1 10 br
tp1 10 br +2
```

Syntax:

```
tp# <schaden> [trefferzone] [wundschwellenmod]
```

`tp` zieht Rüstung ab und erzeugt daraus Schadenspunkte.

Schadenspunkte

```
sp1 8
sp1 8 ko
sp1 8 br +2
```

`sp` wird direkt als Schaden angewendet.

Wunden

Wunden werden anhand von Schaden, Trefferzone und Wundschwellen verwaltet. Trefferzonen wie `ko`, `br`, `ba`, `ra`, `la`, `rb`, `lb`, `ru` und `rü` können verwendet werden.

Schadensmultiplikatoren

Schadensmultiplikatoren werden mit `x` angegeben und vor Rüstung angewendet. Schadensmultiplikatoren dienen dazu, Resistenzen und Verwundbarkeiten abzubilden.

Beispiele:

```
tp1 10 x2
tp1 10 x0.5
tp1 10 x0,5
tp1 10 br x2
tp1 10 br +2 x1.5
sp1 8 x2
```

Ohne Multiplikator gilt `x1`.

Gültige Dezimaltrennzeichen:

```
x0.5
x0,5
x1.5
x2
x2.0
```

Der Multiplikator wird einmal angewendet und danach gerundet, bevor Rüstung verarbeitet wird.

18. Debug-Modus

Der Debug-Modus zeigt bei Kampfproben eine zusätzliche Zeile. Normal ist er ausgeschaltet.

Einschalten:

```
debug an
```

Ausschalten:

```
debug aus
```

Beispielausgabe:

```
Söldner trifft nicht mit dem Säbel!
(DEBUG: AT 17, nextmod -17, effektiv 0, Wurf 10)
```

Debug-Ausgaben gibt es für:

- Attacke
- Passierschlag
- Parade
- freies Ausweichen
- meisterliches Ausweichen
- Fernkampfangriff

Nützlich ist der Debug-Modus vor allem für `nextmod`, also übertragene Erschwernisse aus misslungenen Manövern.

19. Häufige Beispiele

Kampfrunde starten:

```
kr
```

Alle NSC bereit machen:

```
b
bereit
```

Waffe ziehen:

```
1 z
1 z la
1 z beide
```

Nahkampfangriff:

```
1 N
1 N w4
1 la N f2
```

Eingehenden Angriff abwickeln:

```
a2 N 8 br
a2 N 8 br p2
a2 N w4 12 br p4
```

Ausweichen:

```
a2 N 8 br a
a2 N 8 br m2
```

Gegenhalten:

```
a2 N 10 br g18
```

Schaden:

```
tp2 9 br
tp2 9 br x2
sp2 4 ko
```

Licht:

```
l normal
l mondlicht
l dunkelheit
l -2
```

Direkte Aktionen:

```
q3
aus3
e3 +2
ini3 -4
rem3
```

Debug:

```
debug an
debug aus
```

20. Fehlersuche

Das Kommando wird nicht erkannt

Prüfe:

- Kämpfernummer vorhanden?
- Leerzeichen zwischen Kämpfernummer und Befehl?
- Manöver ohne Leerzeichen geschrieben, zum Beispiel `w2,f3`?
- Bei signierten Modifikatoren Vorzeichen verwendet, zum Beispiel +2?
- Schadensmultiplikator mit x, zum Beispiel x2, nicht *2?

Falsch:

```
1z
1 N w2, f3
ini1 x
tp1 5 n x
```

Richtig:

```
1 z
1 N w2,f3
ini1 -2
tp1 5 br x2
```

Eine Waffe ist nicht gezogen

Prüfe:

```
1 w
1 z
1 z la
1 z beide
```

Wenn ein Angriff ohne gezogene Waffe ausgeführt wird, fordert CombatAlrik zum Ziehen oder Bereitmachen auf.

Die linke Hand greift nicht an

Prüfe:

```
1 z la
1 la N
```

Die Nebenhand muss gezogen sein und der Kämpfer muss die passenden Voraussetzungen erfüllen.

Rücken verursacht Probleme

Verwende:

```
tp1 5 ru
tp1 5 rü
```

Beide Formen werden akzeptiert.

Debuggen von Manövererschwernissen

Wenn unklar ist, ob eine misslungene Meisterparade oder ein misslungenes Angriffsmanöver die nächste Probe erschwert:

```
debug an
```

Danach zeigt CombatAlrik `nextmod`, effektiven Wert und Wurf.

21. Kommandoreferenz

Globale Befehle

Kommando	Bedeutung
`help [thema] [unterthema]`	Hilfe anzeigen
`kr`	neue Kampfrunde
`w6`	W6 ausgeben
`w20`	W20 ausgeben
`clear`	Kampfliste leeren
`bereit` / `b`	NSC bereit machen und neue Runde starten
`licht <wert>` / `l <wert>`	Licht setzen
`debug an`	Debug-Ausgaben einschalten
`debug aus`	Debug-Ausgaben ausschalten

Kämpferbefehle

Kommando	Bedeutung
`# [hand] [DK] [manöver]`	Nahkampfangriff
`# <distanz> [größe] [zone|ansage] [mod] [s]`	Fernkampfangriff
`# lang <name> <zeit>`	langfristige Aktion
`# ziehen [aktionen] [hand|beide]`	Waffe ziehen
`# z [aktionen] [hand|beide]`	Kurzform für Ziehen
`# aktion` / `# ak`	Aktion verbrauchen
`# reaktion` / `# rk`	Reaktion verbrauchen
`# aufstehen` / `# auf`	aufstehen
`# runter` / `# ru`	hinlegen
`# ru knie`	hinknien
`# orient` / `# o`	orientieren
`# klingenwand` / `# kw`	Klingenwand
`# klingensturm` / `# ks`	Klingensturm
`# raserei` / `# ras`	Raserei
`# info` / `# i`	Beschreibung
`# desc`	Beschreibung
`# attribute` / `# char` / `# c`	Werte anzeigen
`# rüstung` / `# rü`	Rüstung anzeigen
`# waffen` / `# w`	Waffen anzeigen
`# sfs`	Sonderfertigkeiten anzeigen
`# uat`	Parade in Attacke umwandeln
`# upa`	Attacke in Parade umwandeln
`# dk+ [mod]`	Distanzklasse vergrößern
`# dk- [mod]`	Distanzklasse verkleinern

Eingehende Angriffe

`a# [DK] [angriffsmanöver] <schaden> <trefferzone> [verteidigung]`

Beispiele:

```
a1 N 8 br
a1 N w4 8 br p2
a1 X 10 ko a
a1 N 10 ko g18
a1 fk 10 ko
a1 fk 10 ko a
a1 fk 10 ko m +4
```

Direkte Zielbefehle

Kommando	Bedeutung
`tp# <schaden> [zone] [wundmod] [x...]`	Trefferpunkte zufügen
`sp# <schaden> [zone] [wundmod] [x...]`	Schadenspunkte zufügen
`ini# <wert>`	Initiative ändern
`rem#`	Kämpfer entfernen
`e# [mod]`	Entwaffnen aus Parade
`q#`	Ausfall beenden
`aus#`	Ausfall beenden

Distanzklassen

```
H
N
S
P
X
```

Angriffsmanöver

Token	Manöver
w	Wuchtschlag
f	Finte
g	Gezielter Stich
s	Sturmangriff
h	Hammerschlag
p	Passierschlag
t	Todesstoß
e	Entwaffnen
n	Niederwerfen
b	Befreiungsschlag
d	Doppelangriff / Doppelschlag
a	Ausfall

Verteidigungsmanöver

Token	Manöver
p	Parade / Meisterparade
e	Entwaffnen
a	Ausweichen
m	Gezieltes Ausweichen
b	Binden
g	Gegenhalten
w	Windmühle

Lichtwerte

```
l normal
l sonnenschein
l fackel
l lagerfeuer
l dämmerung
l mondlicht
l dunst
l nebel
l dichter nebel
l undurchdringlicher nebel
l sternenlicht
l dunkelheit
l -8
l 0
l +1
l +2
```