Olympia II

Rules of Play

by Rich Skrenta
Questions, comments, to join: info@pbm.com

Distribution

Olympia, its rules, documentation and game text are Copyright © 1991-1994 Rich Skrenta. All rights reserved.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the same conditions as for modified versions.

1 Introduction

Olympia is an open-ended computer moderated fantasy simulation. Characters move, battle, explore and study in the Olympian world. Each week, players submit orders for their units. After the turn runs, Olympia sends reports to the players detailing what happened.

Olympia is set in a low technology fantasy world. Characters do not have fixed goals. They may study whatever skills they believe will be useful and pursue goals as they find them. Olympia has no victory conditions; no winner is ever declared.

An aspiring heroic fighter may purchase weapons, study combat and slay monsters. Business-minded individuals can establish a profitable trading empire. The study of magic could furnish tools to advance dark plans. An explorer could trade his gold to a shipbuilder for a galley, and set out on the high seas to find adventure and treasure.

Olympia turn reports tend to be long, detailed and dense. Since your player character may hire other units which can study, move and fight on their own, large factions of characters may be organized. Supplying orders for many units in a large faction can be a lot of work!

A turn report for a new player controlling three characters is about five (66 line) pages. Average turn reports tend to be 15-25 pages. The size of the report will depend on whether a player chooses to concentrate development in a few characters, or to pursue an empire- building strategy of acquiring many units and controlling as much territory as possible.

1.1 Overview

Each player begins with control of one character and a sum of gold, the Olympian currency. Characters may hire subordinates, study skills, trade items, build things, and explore. Each turn a player submits orders for all of the characters they control. These include the initial character, known as the Player Character, and any vassals of the player character, which belong to the player's faction.

The Olympia program will execute orders for characters in parallel. Each order may take no time or some number of game days to complete. As many orders will be processed for each character as game time allows. Orders not completed by the end of the turn may continue into the next. Each turn covers one month of game time. An Olympian month lasts 30 game days.

1.2 Turn schedule

Turns are run once per week, on Wednesday. Orders are due at 6:00pm, Eastern U.S. time. Players should receive their turn reports by the following morning.

Late orders will queue for the next turn. Beware! Even if mailed before the deadline, your message may take some time to get to the order scanner. Send your orders in a day early to avoid grief. This will also give you time to correct mistakes reported by the order scanner.

New player additions are performed just after the turn is run on Wednesday.

1.3 Olympia Times

Players may submit articles or notices to *The Olympia Times*. The *Times* is published once per week, if there is sufficient material. Submissions may be creative stories about your characters, advertisements for goods to be sold within the game, proclamations, etc. Generally anything in good taste with some relevance to the game will be accepted.

Any reasonable submission will earn 25 gold. To receive payment, the '`Subject:`' header of your Times article **must** say '`credit` *who*', where *who* is the entity-number of the character to receive the 25 gold.

For example:

```
To: times@g1.pbm.com
Subject: credit 501

Plans are being drawn up to build a place of worship just outside of
Drassa.  Worshipers of the Twilight Goddess are gathering for the
blessing.  The worship circle shall not be the temple of some dogmatic
religion, we are not seeking followers, only fellowship with friends.
```

Please limit articles to a line length of 72 characters, to prevent lines from wrapping after formatting.

1.4 Diplomatic email forwarding service

Players can send mail to the player controlling an entity by mailing to *entity*@*game*.**pbm.com**.

For example, suppose you saw the following characters:

```
Seen here:
   Circe [2225], with three peasants, accompanied by:
      Hephaestus [3446], with 11 peasants
```

Circe's lord isn't identified, so you don't know what player is controlling her. However, it is still possible to send Circe's owner a message. Mail to **2225@g1.pbm.com** and the message will be forwarded to whatever player is controlling Circe [2225].

Forwarding works for character and player entities. Mail sent to other entity numbers, or to unaffiliated characters, will be silently discarded.

Forwarded mail is not anonymous. The original headers on your message will be preserved.

1.5 Compensation for errors

What if there's a bug which ruins your whole turn? What if an error in the combat system lets all of your men be killed in one day?

- I will not re-run turns. Turns can't be rerun for one player, the entire simulation has to be run for everyone. This takes some time and produces a large amount of data which is mailed out. Only for the most serious of game disasters would I consider rerunning a turn.
- If you are stung by a bug, I may grant items for compensation, but I won't alter the universe. I might give you 1,000 gold if you are the victim of some heinous bug, or perhaps even an NP, but I won't move a unit that should have been in Pactra to Drassa.

1.6 Cheating

1. A player may not have more than one faction in the game. Also, multiple players from the same account are not allowed. Each player must have their own unique email address.

2. Players should not send in orders for another player's faction, to ruin that person's turn or otherwise benefit.
3. Players must inform the GM of any game bugs found.
4. Anti-social behavior, including harassing telephone calls, sending obscene/obnoxious unwanted communications, mail bombing, etc. will not be tolerated.

Punishment for serious violations is generally banishment from Olympia. So don't cheat, play fair, and be a good sport.

All decisions of the GM are final.

1.7 Olympian Calendar

The Olympia calendar has two months for each season, for a total of eight months per Olympian year. Each month is 30 game days long.

Fierce winds
: Spring

Snowmelt
: Spring

Blossom bloom
: Summer

Sunsear
: Summer

Thunder and rain
: Fall

Harvest
: Fall

Waning days
: Winter

Dark night
: Winter

1.8 Game additions and rule changes

Features may be added to Olympia every so often, such as new areas in the map, NPC races, magical artifacts, skills, spells, etc. It is also sometimes necessary to make slight alterations to existing game mechanics to correct bugs or flaws in game balance.

While it is inevitable that some players will be affected by rule changes, every effort is made to not disrupt existing game positions.

This is mentioned only as a warning that the rules may evolve over time. Changes are announced via email to all players.

1.9 How do I join?

Get a signup sheet by sending mail to **info@pbm.com**, or from the **ftp.pbm.com** ftp server. Fill it out, and mail it to **info@pbm.com**.

Turns cost US$2.50 each. Players may send a US check to Shadow Island Games or elect to have their credit card billed US$30 quarterly. (We are looking into accepting international money orders — ask first).

1.10 Email address, etc.

Joining, info
info@pbm.com

Order scanner
orders@*game***.pbm.com**

The Olympia Times
times@*game***.pbm.com**

Diplomatic forwarding
*entity***@***game***.pbm.com**

The latest rules, back issues of the *Olympia Times*, and other game related material may be downloaded via anonymous ftp from **ftp.pbm.com**.

Visit the Olympia WWW server at **http://www.pbm.com/oly-top.html**.

```
Shadow Island Games
P.O. Box 6435
Somerset, NJ 08875-6435

Telephone:  908-469-1992 (answers 24 hours; leave message)
```

2 Important terms

Entity

Unit Everything in the Olympian world has a unique code. Nobles, items, locations, and skills all are referenced with an *entity number*. The code is shown in brackets after the name. Some examples:

Rich Skrenta [501]
: a player

Osswid the Destroyer [5499]
: a character

Shipcraft [120]
: a skill

City of the Lost [gx14]
: a place

Gold [1] : an item

Scroll [yq12]
: an item

Noble

Character Used interchangeably. These are the individuals under the control of players. All player orders are given to characters. Players start with one character. Others may be hired or persuaded to join the player's faction. Characters may possess items, travel through locations, learn skills, engage in combat, cast magical spells, etc.

Faction All of the units controlled by a player are called the player's faction. A player starts with only one character, but the faction may grow to have many units.

Player character

The player character, or *PC*, is the top-most unit in the faction's command hierarchy. The player's first character is their PC. If the PC is killed, an heir will be installed in the lost PC's place.

Item

Possession Characters may hold items, such as gold, scrolls, weapons, magic potions, jewels, lumber, rugs, etc.

Men Characters may also have non-descript men in their employ. These men are represented as possessions for simplicity. They include peasants, workers, sailors, and different kinds of soldiers. These men may not learn skills, hold any items, or act independently from the noble they are with.

For example, one might see:

```
Seen here:
   Law Netexus [2020], with three peasants
```

Law Netexus is a character; the three peasants are non-descript men accompanying him.

Characters obtain peasants with the `recruit` order.

Skills Characters may learn skills, which are used to perform tasks. For instance, Sailing [9502] must be known in order to sail a ship.

Skills are grouped into the following categories:

- Alchemy
- Beastmastery
- Combat
- Construction
- Forestry
- Mining
- Persuasion
- Shipcraft
- Stealth

There are also six schools of magic.

For information about learning skills, see [study], page 141 and [research], page 135.

Noble Points

A player starts with twelve Noble Points (*NP*). Each player gets an additional NP every eight turns (one each game year). NPs are used to buy nobles with the `form` command. They are also required to learn some advanced skills, and to swear characters to oath loyalty.

Stack A group of characters joined such that they move and fight together.

Province A location on the map. Provinces may have sub-locations within them, such as cities, bogs, caves, etc. Provinces are either forest, swamp, mountain, desert, plains, or ocean.

Month Each turn is a game month, or 30 game days.

Safe Haven

New players start in a randomly selected Safe Haven city. Combat is not permitted in safe havens. New players may acclimate themselves in safety before venturing out into the world.

3 Command summary

command	arguments	time	priority
admit	*who or what* [ALL] [*units*]	0 days	0
attack	*target* [*flag*]	1 day	3
banner	[*unit*] "*message*"	0 days	1
behind	*number*	0 days	1
board	*ship* [*maximum fee*]	0 days	2
bribe	*who amount* [*flag*]	7 days	3
build	*structure* "*name*" [*max-days*]	varies	3
buy	*item qty price* [*have-left*]	0 days	3
catch	[*number of horses*] [*days*]	as given	3
collect	*item* [*number*] [*days*]	as given	3
contact	*who*	0 days	0
decree	*decree*	0 days	0
default	*who*	0 days	0
defend	*who*	0 days	0
drop	*item qty* [*have-left*]	0 days	1
email	*new email address*		
execute	[*prisoner*]	0 days	1
exhume	[*body*]	7 days	3
explore		7 days	3
fee	[*gold per 100 wt*]	0 days	1
ferry		0 days	1
fish	[*number of fish*] [*days*]	as given	3
flag	*string*	0 days	1
fly	*direction or destination* [...]	varies	2
forget	*skill*	0 days	1
form	*unit* "*name of new character*"	7 days	3
format	*num*		
garrison	*castle*	1 day	3
get	*who item* [*qty*] [*have-left*]	0 days	1
give	*to-who item* [*qty*] [*have-left*]	0 days	1
guard	*flag*	0 days	1
honor	*amount*	1 day	3
hostile	*who*	0 days	0
lore	*sheet*		
make	*item* [*qty*]	as given	3

command	arguments	time	priority
message	*# of lines of text to-who*	1 day	3
move	*direction or destination* [...]	varies	2
name	[*unit*] "*new name for unit*"	0 days	1
neutral	*who*	0 days	0
notab	*num*		
oath	*level*	1 day	3
pay	*to-who* [*amount*] [*have-left*]	0 days	1
password	["*password*"]		
pillage	[*flag*]	7 days	3
players			
post	*# of lines of following text*	1 day	3
pledge	*who*	0 days	1
promote	*who*	0 days	1
quarry	[*number of stones*] [*days*]	as given	3
quit			
raze		varies	3
recruit	[*days*]	as given	3
repair	[*days*]	as given	3
research	*skill*	7 days	3
resend	[*turn*]		
sail	*direction or destination*	varies	4
seek	*who*	7 days	3
sell	*item qty price* [*have-left*]	0 days	1
stack	*character*	0 days	1
stop			
study	*skill*	7 days	3
swear	*who*	0 days	1
take	*who item* [*qty*] [*have-left*]	0 days	1
terrorize	*who severity*	7 days	3
train	*kind days*	as given	3
ungarrison	[*garrison*]	1 day	3
unload		0 days	3
unstack	[*who*]	0 days	1
use	*skill* [*arguments ...*]	varies	3
wait	*conditions*	varies	1

4 Submitting orders

Orders should be sent to **orders@*game*.pbm.com**. For instance, if you are in game 1 of Olympia, send orders to **orders@g1.pbm.com**.

The '`Reply-To:`' header on turn reports is set to the correct orders, so using the reply feature of your mailer should send orders to the right place.

The orders address is read by an automated daemon which scans the orders for syntax errors and queues them for your units. It will send a reply as soon as it receives mail from you, showing whether or not there were any errors with the orders it received.

Orders must be of the following form:

```
begin player-number password
  .
  .  email, lore, password commands
  .

unit player-number
  .
  .  commands for player entity:  name, format or quit
  .

unit unit-number
  .
  .  commands for unit
  .

unit unit-number
  .
  .  commands for unit
  .

end
```

The '`Subject:`' line on your message is ignored.

The `begin` keyword tells the order scanner what your player number is. If you have not set a password, you do not need to supply one.

The `unit` command replaces a set of orders for a unit. Any pending orders for the unit will be cleared, and the new orders sent in will queue up. Orders that are still executing for the unit will not be interrupted unless the first order queued is the `stop` order (see [stop], page 140).

Do not match an `end` for every `unit` command! There should only be one `end`, at the end of all of the `unit` sections. The Olympia order parser will not read beyond the `end`.

For example, here is a set of orders for player Fate [812], who has two characters, Osswid [5499] and Candide [1269]:

```
begin 812
  password sneaky

unit 5499
  explore
  move east
  explore
  study 120

unit 1269
  move north
  stack 5499

end
```

The parser tries to be as flexible as possible. It is case insensitive and is not strict about spaces on a line, so you may use indentation to make your orders more readable.

Orders may be commented with the '`#`' character. Everything from a '`#`' to the end of the line will be ignored by the parser:

```
move north    # Head to Drassa to meet up with Osswid
stack 5499    # stack with him
```

No one will read the comments but you. Neither the GM nor the Olympia engine will try to interpret them for any reason.

Note that arguments must be enclosed in quotes if they are more than one word:

```
name "Osswid the Constructor"
```

The acknowledgement will show any errors that occurred while the orders were being parsed, and list the current pending commands for all of your units.

4.1 Interrupting orders

Suppose the turn report shows the following orders queued:

```
unit 5499
   # > study 160 (executing for three more days)
   recruit 10
   explore
```

Sending in new orders for this unit will not disturb the still-running study command unless the first order is `stop`. For example:

If this were sent in:

```
unit 5499
   move south
```

This would be the result:

```
unit 5499
   # > study 160 (executing for three more days)
   move south
```

To interrupt the study and get on with the move right away, instead send in:

```
unit 5499
   stop
   move south
```

This will show:

```
unit 5499
   # > study 160 (executing for three more days)
   stop
   move south
```

Note that the stop queues like any other order; it does not actually interrupt the executing command until the turn runs. This means that the stop itself can be replaced by sending in another set of orders later.

4.2 Units not controlled by you (yet)

Orders may be sent in for units which are not yet under control, such as characters that you intend to `bribe` or `terrorize` into switching to your faction.

As soon as the unit comes under your control, the orders queued for it will begin to execute.

Orders may also be sent in for new nobles which will be formed during the turn. First choose one of the possible unit numbers from the choices listed near the beginning of the turn report:

```
The next five nobles formed will be:  5717 3215 4902 4489 5628
```

Supply one of these numbers as the first parameter to the `form` order:

```
form 5628 "Feasel the Wicked"
```

Then queue some orders for Feasel to execute as soon as he appears:

```
unit 5628
  unstack
  study 160
  move out
  recruit
  ...
```

4.3 Use the order template

An order template listing all of the units for a player and showing any pending orders for those units, appears at the bottom of the turn sheet.

```
Order template
--------------

begin 812  # Master Bogomil's Family

unit 2508  # Tudor
   # > make 74 (still executing)

unit 2947  # Milo

unit 4375  # Beorn
   # > move s (executing for one more day)
   pillage
   recruit

unit 4763  # Sylvia

unit 5977  # Drango
   # > collect 87 0 0 (still executing)
   sail e
   sail s
   fish
   explore

unit 5418  # Comte de le Sang

end
```

Note that the layout of the order template matches the syntax the order scanner expects. Many players find it convenient to edit this template to add or change commands for their units. Mail everything from the `begin` to `end` (inclusive) to the order scanner. It is wise to save a copy of the orders you submit in case there are errors and they need to be resent.

4.4 Be careful

Beware of sending in different sets of orders too quickly. Sometimes messages sent within a short time of each other will arrive out of order. This can wreak havoc on your turn if the wrong orders arrive last. Compose your orders offline and review them before mailing. A simple typographical error in your orders could ruin your whole turn!

4.5 Failed orders

Commands that fail generally take zero time. For instance, if `study` is issued for a skill which the location does not offer, it will fail immediately, and take zero time. The failed `study` order will not take a week, and it will not count toward the diminishing study returns for the month.

Production commands fail immediately if none of their input resources are available. For instance, `recruit` in a location with no peasants will immediately fail, taking zero time. However, resources may sometimes become depleted while the command is being executed. In such cases, the command may fail even after it has spent some time executing.

4.6 Player Entity

Each Olympia player faction has a number. This number is represented by an entity in the game. However, unlike a character, this entity is mostly used as a place holder for the faction. No one can see the faction entity, and it can issue very few orders.

For example, suppose player Fate [501] has one character Osswid [5499]. Fate does not exist in any location, so it does not receive a location report, and no one can see it. However, Osswid, being the player character for the faction, is sworn to Fate [501]. Fate may execute only a limited set of administrative orders.

Characters, not factions, issue most orders. Do not try to `form` or `recruit` with the player entity. For most turns, the player entity will have no orders queued for it.

The orders a player entity may issue are:

```
admit
format
name
quit
```

4.7 Quitting

To drop out of the game, issue the `quit` order for your player entity. For instance, player 501 would quit by sending in the following orders:

```
begin 812 password
unit 812                # don't forget unit for the player unit!
   quit

end
```

No turn report will be sent for the turn in which a player quits.

4.8 More examples

You can replace orders for some units, but leave pending orders for other units alone, by only including `unit` sections for the ones you want to change.

If you want to see what orders are queued, but not change anything send in:

```
begin 999               # whatever your player number is ...
end
```

To change your email address, send in

```
begin 999
email new@address.com   # give your new address
end
```

As a security measure, the confirmation will be sent to both the new and old addresses.

To change the name of your faction, issue the `name` order for the faction's player entity:

```
begin 999
unit 999
   name "Seekers of Fame and Power"
   ...
```

Important: Don't forget the `unit` command for the player entity.

4.9 Order of characters in a location

When a character enters a location, it is added to the end of the list of characters already there. The unit that has been in a location the longest will appear at the top of the list.

If a character leaves a location and later returns, it will be put at the end of the list again.

For example:

```
Seen here:
   Candide [1269]
   Osswid [5499]
   Feasel the Wicked [1109]
```

Candide has been here longest, followed by Osswid, then Feasel. If Candide were to leave and return, he would appear at the end of the list.

If a character unstacks from beneath another unit, the character will appear just after the unit, rather than at the end of the list.

For example:

```
Seen here:
   Candide [1269], accompanied by:
      Osswid [5499]
   Feasel the Wicked [1109]
```

Osswid is stacked beneath Candide. If Osswid unstacks, he will appear after Candide, not after Feasel:

```
Seen here:
   Candide [1269]
   Osswid [5499]
   Feasel the Wicked [1109]
```

4.10 Command priority

All orders have a priority of 0-4.

Permission commands (`admit`, `hostile`, etc.) are priority 0.

Zero time commands[1], and `wait`, are priority 1.

`move` and `fly` are priority 2.

The `sail` command is priority 4.

All other commands are priority 3.

The order scheduler will first try to start all priority 1 orders. Only when no more priority 1 orders are ready to start will a priority 2 order be started.

In other words, the order scheduler will not start an order at a higher priority when an order may be started at a lower priority.

Orders at the same priority are resolved in location order. If two units in a location are both waiting to start a `move` order, the first unit in the location will go first.

The above description of order priorities may seem complicated, but the intent is to let players ignore same-day synchronization issues in most cases. Rather that needing `wait` to guarantee that `give` happens before `move`, the lower priority of `give` makes this happen naturally.

For example, consider three units stacked together, top, mid and bot:

```
top:    move ec69
        yew

mid:    unstack
        recruit

bot:    recruit
```

These should be executed in the following order:

[1] A zero time command is an order which always takes zero time. This does not include an order which may sometimes take zero time. For instance, `unstack` is always a zero time order. However, `recruit` is not, even though `recruit` may terminate immediately under some conditions.

```
mid: unstack                    # unstack is prio-1
top: move ec69                  # move is prio-2
mid: recruit                    # recruit is prio-3

[top and bot arrive at ec69]

top: yew                        # yew is prio-3
bot: recruit                    # recruit is prio-3
```

The `unstack` happened first since it's a priority 1 command. The `move` went second. When top and bot finished moving, there were only priority three commands left, so they ran in location order.

4.11 Command precedence within a location

```
Seen here:
   Candide [1269]
   Osswid [5499]
   Feasel the Wicked [1109]
```

Order precedence within a location is an advantage for commands or skill uses which obtain resources from the location. For instance, if Candide and Osswid both attempted to harvest all of the lumber available in their location, Candide would have precedence, since his harvest order would finish before Osswid's, if they were started on the same day.

4.12 At the same time . . .

No two things ever happen at exactly the same instant in Olympia. Someone always goes first.

Suppose two characters were outside of a building (which nobody is inside), and both wanted to enter, to claim it:

```
Seen here:
   Osswid [5499]
   Candide [1269]

Inner locations:
   Hooting Own Inn [ep76], inn
```

Both Osswid and Candid issue '`move ep76`' as their first order on day 1 of the month. What happens?

Osswid's command begins before Candide's, since Osswid appears before Candid in the location list. Therefore, Osswid will enter the inn first.

Order precedence between units in different provinces is undefined.

```
Osswid              ?             Candide
------- 8 days -------- 8 days --------
city A           city B           city C
```

If Osswid and Candide both leave for city B on the same day, we cannot predict who will get their first.

5 Olympian geography

Olympia's map is a large grid of locations called provinces. Groups of provinces form continents, islands and oceans. These collections are called regions, and are usually named.

A province's description will include a list of the directions in which a character may travel:

```
Plain [ae48], plain, in region Tollus

Routes leaving Plain [ae48]:
   North, to Plain [ad48], 7 days
   East, to Plain [ae49], 7 days
   South, to Ocean [af48], Tymaerian Sea, 1 day
   West, to Ocean [ae47], Tymaerian Sea, 1 day
```

This is a non-descript province in the Tollus region (Tollus is a small continent in the northeast corner of the map.) From this province, a character may travel north or east on foot or by horse, or may sail by ship to the south or west.

```
move north  -or-   move n   -or-   move ad48
move east   -or-   move e   -or-   move ae49

sail south  -or-   sail e   -or-   sail af48
sail west   -or-   sail w   -or-   sail ae47
```

Land movement will automatically use the fastest available mode. For example, if a character has enough horses for all of the members in the party to ride, then the travelers will go on horseback.

Ocean movement requires that the character be in a ship.

Route distances are rated for the number of days it normally takes to traverse them. Land distances are rated for a lightly loaded character walking, and ocean distances are given for an ordinary ship traveling in normal weather.

Actual travel times may differ from times given in the route listing. Land distances depend on the surrounding terrain and the modes of transport available. For example, horses often speed up movement, but over especially rough or treacherous terrain, they may actually slow travel because they must be led and managed. A stiff wind may speed ocean vessels, while lack of wind may slow their progress.

5.1 Inner Locations

A province may contain sub-locations within its borders. Sub-locations may usually only be entered from the surrounding province. They will be listed separately in the location description:

```
Inner locations:
   Carim [em28], city, 1 day
```

The city Carim may be entered with the '`move em28`' order. Travel into a city requires one day.

(`move in` may be used to enter a sub-location, although this order may be ambiguous if the location contains more than one sub-location. In such a case, the first sub-location in the Inner locations list will be entered. Using `move in` is not recommended if the entity number of the sub-location is known.)

Characters in a sub-location will receive a report for the surrounding province. However, characters in the outer province will not normally be able to see into an inner location without entering it.

5.2 Inside a City

```
Carim [em28], city, in province Plain [ae48]

Routes leaving Carim [em28]:
   Out, to Plain [ae48], 1 day

Inner locations:
   Hooting Own Inn [ep76], inn
```

Characters in the city Carim may `move out` (or '`move ae48`') to the surrounding province. They may also attempt to enter the inn, which is a sub-location of the city. Notice that no travel time rating is listed for the inn; entering it takes no time (zero days).

```
move out    -or-    move ae48
move ep76
```

A character in Carim will receive a location report both for the city as well as the surrounding province Plain [ae48]. Characters in the Hooting Own Inn will receive a location report for the inn and one for Carim, but will not get a report for Plain [ae48]. A character in the city may not see inside the inn without entering.

Characters in a sub-location receive a report for the immediate surrounding location.

Characters are not able to see into inner locations without going into them.

5.3 Who else is here?

Characters spotted will be listed in the location report:

```
Seen here:
   Fighters of Pelenth [2019], "carrying a gold banner"
   Osswid the Constructor [5499]
```

All characters listed as being seen in the location may interact without requiring any travel. Thus, the Fighters of Pelenth and Osswid are considered to be in essentially the same place.

This is true whether the characters are in a province, a city, a ship, an inn, or some other sub-location.

However, a character in a sub-location may not interact with characters in the surrounding area. A noble in the city must first enter the inn before he may interact with those inside.

5.4 More about geography

Olympian provinces are arranged in a square grid. Travel is possible in the four main compass points. Thus, to move diagonally, two `move` orders are required. To move northwest, for instance, one would first need to '`move n`', then '`move w`'.

The map coordinates for a province may be read from the province's entity number. The row is represented by the leading two letters, the column by the two digits. The northwest corner is [aa01], with rows increasing to the south, and columns increasing to the east.

	00	01	02	03	04	...	98	99	00	01	...
aa	aa00	aa01	aa02	aa03	aa04	aa...	aa98	aa99	aa00	aa01	aa...
ab	ab00	ab01	ab02	ab03	ab04	ab...	ab98	ab99	ab00	ab01	ab...
ac	ac00	ac01	ac02	ac03	ac04	ac...	ac98	ac99	ac00	ac01	ac...
ad	ad00	ad01	ad02	ad03	ad04	ad...	ad98	ad99	ad00	ad01	ad...
...	...00	...01	...02	...03	...04	...	...98	...99	...00	...01	...
du	du00	du01	du02	du03	du04	du...	du98	du99	du00	du01	du...
dv	dv00	dv01	dv02	dv03	dv04	dv...	dv98	dv99	dv00	dv01	dv...

Entity numbers for sub-locations do not correspond to any coordinate system.

The map wraps at the east and west edges. Traveling west from ac00 will lead to ac99, for instance.

5.5 Holes in the Map

The map has some *holes*, representing impassable provinces. Routes into some provinces may also be hidden.

```
Plain [cd21], plain, in region Tollus

Routes leaving Plain [cd21]:
   North, to Plain [cc21], 7 days
   East, to Plain [cd22], 7 days
   West, to Plain [cd20], 7 days
```

Notice the lack of a southern exit. This means that there is no known southern route from Plain [cd21], into what should be Plain [ce21]. Exploration may find a southern route, but it is possible that none may ever be found, and the terrain to the south is completely impassable.

5.6 Hidden routes

If exploration finds a hidden route, any noble in the player's faction will be able to use it.

```
> explore
A hidden route has been found!

   South, to Plain [ce21], 7 days
```

The location description for this place will now include the hidden route:

```
Plain [cd21], plain, in region Tollus

Routes leaving Plain [cd21]:
   North, to Plain [cc21], 7 days
   South, to Plain [ce21], 7 days, hidden
   East, to Plain [cd22], 7 days
   West, to Plain [cd20], 7 days
```

However, units from other factions, even if they know that the hidden route's entity number is [ce21], will not be able to travel across it.

All factions with units in a stack traveling across a hidden route, with the exception of units being held prisoner, will learn of its existence. Nobles from factions wanting to learn how to use the hidden route can stack with a noble about to move across the route.

5.7 Ocean ports

A ship in an ocean province may sail into an adjoining land province.

```
Ocean [cw12], ocean, in South Sea

Routes leaving Ocean [cw12]:
   North, to Ocean [cv12], Atnos Sea, 4 days
   East, to Mountain [cw13], West Camaris, impassable
   South, to Plain [cx12], West Camaris, 1 day
   West, to Ocean [cw11], 4 days

Inner locations:
   Island [eb97], island, 1 day
```

A ship sailing in this ocean province may dock by sailing to Plain [cx12] or Island [eb97].

Ships may not dock in mountain provinces, as the rocky cliffs are too dangerous to approach. Routes between ocean and mountain provinces are marked '`impassable`'.

5.8 Port Cities

A city in a province adjoining an ocean will have been founded on the best spot for an ocean port. The ocean will only be accessible through the port city in this case, and not through the surrounding region.

```
Plain [ae48], plain, in region Tollus

Routes leaving Plain [ae48]:
   ...
   West, to Ocean [ae47], Tymaerian Sea, impassable

Inner locations:
   Carim [em28], port city, 1 day
```

Note that from the province surrounding the port city, access to the ocean is not possible.

```
Carim [em28], port city, in province Plain [ae48]

Routes leaving Carim [em28]:
   West, to Ocean [ae47], Tymaerian Sea, 1 day
   Out, to Plain [ae48], 1 day
```

However, ships may sail into and out of the port city itself. From the Tymaerian Sea, this looks like:

```
Ocean [af48], ocean, in Tymaerian Sea

Routes leaving Ocean [af48]:
   North, city, to Carim [em28], Tollus, 1 day
   South, to Ocean [ag48], 3 days
   ...
```

5.9 An example city description

```
Drassa [ew66], port city, in province Forest [cu26], safe haven

Routes leaving Drassa [ew66]:
   East, to Ocean [cu27], Atnos Sea, 1 day
   South, to Ocean [cv26], Atnos Sea, 1 day
   Out, to Forest [cu26], 1 day

Skills taught here:
   Shipcraft [120]
   Combat [121]
   Construction [125]

Seen here:
   Kosar the Indefectible [2022], with six peasants, one archer,
   two soldiers, accompanied by:
      Dr. Pangloss [3682]
      Law Netexus [2020], prisoner
      Alion Krysaka [2785], prisoner

Ships docked at port:
   HMS Pinafore [ib18], galley, owner:
      Captain McCook [2019], with five workers

Market report:
   No goods offered for trade.
```

5.10 Wilderness and civilization

Every province has a civilization level. Provinces with no civilization (a level of zero) are considered wilderness. Civilization levels for provinces are shown in the turn report:

The civilization level is shown for every province appearing in the turn report:

```
Mountain [cq24], mountain, in Lesser Atnos, civ-1
Forest [ac35], forest, in Torba Bacor, wilderness
```

The civilization level of a province is determined by the presence of cities and buildings, or half of the maximum civilization level of its surrounding provinces, whichever is higher.

feature	contribution
safe haven	2
castle	1.5
city	1
tower	1
temple	1
inn	0.5
mine	0.5

Any fractional remainder is dropped after the contributions are summed.

Examples:

A province with no cities or buildings adjacent to a civ-2 province would have a level of civ-1.

A city and a castle would result in a level of civ-2 (the .5 is truncated). A safe haven city, and inn, and a mine would yield a level of civ-3.

A province with a safe haven city surrounded by provinces with no cities or buildings would have a level of civ-2. The surrounding provinces would then rise to civ-1.

A province containing a castle would have a level of civ-1. However, if a neighbor had a level of civ-5, then the province's level would rise to civ-2.

5.11 Dangerous places

Players should take care when exploring the Olympian world. There are many dangers, both from non-player characters (NPC's) as well as from other players. While the threat of death to nobles is always present, the following dangerous areas warrant extra caution.

5.12 Wilderness

Units in the wilderness run the risk of attack by bandits, fiends, or monsters. Stacks with a combined defense value less than 200 should avoid wilderness areas. While some monsters will attack any unit they encounter, strong stacks generally face less danger than weaker ones.

5.13 Hades

Hades, also known as The Land of the Dead, is a subterranean world populated with demons, ghouls and spirits thirsty for the blood of the living. Only the bravest warriors should consider walking these dark paths.

name	combat rating
Greater Demon	(500,500,0)
Lesser Demon	(250,250,0)
Ghostly presence	(100,100,0)
Tortured spirits	(50,50,0)

5.14 Faery

The Faery world lies in a nearby, but separate reality. Occasionally a Faery hill will exist simultaneously in both Faery and the outside world. During this time, mortals may cross between the two worlds.

Faery is protected by *Faery Hunts*, tough bands of elves armed with magical bows. Each hunt group consists of 10-50 elves, each with a combat rating of (50,50,100).

Rumors speak of a magical talisman, the elfstone, which allows mortals to pass unharmed in Faery, and to summon Faery hills to the mortal world.

5.15 The Cloudlands

The Cloudlands is a small region which floats over Mt. Olympus and the Imperial City. It is generally only accessible by flight.

The Cloudlands is home to three cities: Nimbus, Stratos and Aerovia. Weather magic is taught in these three cities.

5.16 Guard monsters

Monsters are sometimes found guarding sublocations, such as a lairs, caves, or graveyards. These locations may be hidden, requiring the noble to `explore` to find them. Often these monsters will have some rare or useful possession, or in some cases may have a prisoner. Nobles who attack and defeat these monsters can expect to obtain the monster's treasure.

Such treasures include:

gold
a quantity of some rare tradegood
a magical artifact
a book which teaches a rare skill
a prisoner who is grateful for being rescued

6 Health

Nobles have a health rating of 1-100, which indicates how wounded they are. A noble with health 100 has no wounds; a noble with a health of zero dies. Nobles also have a flag which indicates whether they are suffering from an illness. A sick noble will lose some health each week, while a wounded noble who is otherwise free of illness will recover somewhat each week.

Health is shown in the turn report for each noble that the player controls.

This is a noble in perfect health:

```
Health:          100%
```

This is a rather sick noble:

```
Health:          38% (getting worse)
```

If the noble were cured of illness, this would instead show:

```
Health:          38% (getting better)
```

Medical technology is rather crude in the age of Olympia. Sanitation and hygiene are not the best. Even a minor wound runs a risk of developing into a serious, possibly life-threatening problem.

When a noble receives a new injury, their health is reduced by the amount of the injury, and a check is made to see whether they get sick. The chance that a character falls ill is (100 - health). Thus, the more seriously the noble is wounded, the greater the probability that infection will set in.

Health is updated at the end of each game week (on days 7, 14, 21 and 28). Sick nobles lose 3-15 health each week. Healthy but wounded nobles recover by a like amount. Each week sick nobles have a 5% chance of fighting off their illness.

Nobles located in an inn benefit from the rest and relaxation that the inn provides. Inns increase the chance of fighting off infection to 10% each week.

Illness can also be cured by spells, special skills, or magical potions. Sick nobles would be wise to seek out those practiced in healing, and recover in a nearby inn.

Notes:

1. Characters who recover from illness will not suddenly become sick again. The illness check is only made when a new wound is received.
2. Since the chance that a character becomes sick is based on the unit's health after deducting a hit, an already-wounded noble stands a greater chance of becoming ill from a wound than a healthy one.
3. Health is only tracked for nobles. Peasants, workers, sailors, soldiers, etc. are either dead or alive; they have no health rating.
4. A wound received in combat will be randomly chosen between 1 and 100. Wounds received in other activities are scaled to match the level of danger involved.

6.1 Death

When nobles die, they are buried in a nearby graveyard. A dead body exists for one game year, at which point it fully decomposes, and the dead noble's spirit passes on.

Example: A dead body rots after eight months have passed, so if a noble dies in turn 20, his body will decompose at the end of turn 28.

The bodies of nobles lost a sea will wash ashore somewhere, and be buried by the locals.

Bodies may be obtained from graveyards with the `exhume` command. Bodies decompose after one year regardless of whether they remain in the grave or are exhumed.

Priests may learn a skill to administer last rites, hastening the passing of the dead noble's spirit. Some exceptionally skilled priests possess the ability to resurrect dead characters.

Once a dead body decomposes, or last rites are administered, the former owner of the noble will get back 1 Noble Point. NPs are not returned for nobles who are resurrected or defiled by Necromancy spells.

7 Markets

Characters issue `buy` and `sell` commands to indicate their desire to trade goods. Every trade must be between a buyer and a seller. Whenever a compatible `buy` and `sell` request are found, the trade will be executed.

Trades are only matched in cities, where merchants gather at the local bazaar to swap goods.

(Note that characters are always free to use the `give` command to exchange items, regardless of where they are. But `buy` and `sell` orders are only matched in cities.)

A buyer indicates what he wants to buy, how much of it he wants, and the most he is willing to pay. A seller indicates what item is being sold, how much of it is to be sold, and the per-item price.

A trade will match if the seller's price does not exceed the maximum price specified by the buyer; if the buyer has enough gold to buy at least one of the item; if the seller possesses at least one of the item; and if the buyer and seller are both in the same city.

If the buyer is willing to pay more than the seller is asking, the trade takes place at the seller's price.

Example: A noble who wants to buy five iron [79] at no more than 10 gold each issues:

```
> buy 79 5 10
Try to buy five iron [79] for 10 gold each.
```

If someone who had iron issues a `sell` order which matches this buyer's request, the trade would be executed:

```
> sell 79 5 10
Sold five iron [79] for 50 gold.
```

The buyer would see:

```
Bought five iron [79] for 50 gold.
```

Either the `buy` or the `sell` order could have been issued first. If the order can't be matched with pending trades from other units in the city, it will become a standing order and remain in effect until executed or canceled.

```
> buy 79 0
Cleared pending buy for iron [79].
```

Note that the buyer and seller can't specify what character they will deal with. They will trade with any unit that has a matching `buy` or `sell` order.

7.1 Market report

The location report for each city includes a market report listing pending trades.

```
trade    who    price    qty   item
-----    ---    -----    ---   ----
  buy   2019      100      1   peasant [10]
  buy   4274       74      3   iron [79]
 sell   3682       12     11   sailor [19]
 sell   2019       50      1   elite guard [15]
```

A trade will not be listed unless it could be executed. For instance, a unit might issue an order to sell iron [79], even though the unit doesn't possess any. (Perhaps the character plans to get some iron later in the month, and wants the `buy` order to be in place when the iron arrives). This order will not be shown in the market report, because the seller doesn't have any iron to sell.

Similarly, the buyer must have enough gold to buy at least one of the item. No buy order would be listed for a penniless unit that wanted to buy five iron at 10 gold each. If the unit later obtained 10 gold, enough to buy one of the five desired units of iron, the order would be listed in the market report for one iron, not five.

Some traders may work through middlemen to hide their identity. In such cases, the *who* field of the market report will not show their unit number, and their identity will not be revealed when the trade is executed.

Even sneakier traders may have their pending trades omitted from the market report entirely.

7.2 City purchasing

Some cities will issue `buy` or `sell` orders on their own behalf for certain tradegoods. Port cities may buy a certain amount of fish each turn. Other cities may desire goods such as ale, pipeweed, rugs, spice, etc. These may be produced by player units through the use of skills, or bought from other locations and transported to the buying cities.

Enterprising characters may turn a profit by moving goods between cities, and shrewdly trading with them.

7.3 Resolution of trades

If a buyer has a choice between two or more sellers, the one offering the lowest price will be chosen. If several sellers offer at the same price, the one nearest the top of the location's character listing will win the trade. A seller with a choice among two or more buyers will pick the highest unit as it appears in the location's character listing.

Since characters are added to the end of the list when they enter a location, the units who have been in a place the longest tend to appear toward the top of the list. Characters who have been in a place the longest have an advantage when one of several possible trades may be matched.

However, consumption or production by the city itself will always have lowest priority. Cities defer to characters when multiple matching trades are possible.

The location market report is ordered according to how multiple trades will resolve.

```
trade    who    price    qty   item
-----    ---    -----    ---   ----
 sell   3682       10      1   iron [79]
 sell   2019       12      1   iron [79]
```

A buyer would get 3682's iron, since the price is lower.

```
trade    who    price    qty   item
-----    ---    -----    ---   ----
 sell   2019       10      1   iron [79]
 sell   3682       10      1   iron [79]
```

A buyer would get 2019's iron. 2019 must appear before 3682 in the location character list.

```
trade    who    price    qty   item
-----    ---    -----    ---   ----
  buy   4846       10      1   iron [79]
  buy   1783       10      1   iron [79]
```

4846 would win the buy from a lone seller. 1783 must come after 4846 in the location character list.

```
trade    who    price    qty   item
-----    ---    -----    ---   ----
  buy   4846       10      1   iron [79]
  buy   1783       11      1   iron [79]
```

sell 79 1 10 will match 4846. sell 79 1 11 will match 1783.

7.4 Partial trades

Partial trades are executed, if possible. If a buyer wants 100 iron, but only 25 iron are available, he will buy 25. The pending `buy` order will be reduced so that 75 more will be bought whenever possible.

7.5 More market examples

A buyer who desires three iron at no more than five gold each orders:

```
> buy 79 3 5
Try to buy three iron [79] for 5 gold each.
```

Suppose the only seller of iron in this city had ordered:

```
> sell 79 5 6
Try to sell five iron [79] for 6 gold each.
```

The trade will not take place, because the seller's price exceeds the buyer's. Later, the buyer travels to a different city, where another seller has previously issued:

```
> sell 79 10 4
Try to sell ten iron [79] for 4 gold each.
```

As soon as the buyer enters the city, the trade will be matched:

```
Bought three iron [79] for 12 gold.
```

Note that the trade takes place at the seller's price, and that the seller will still be offering seven iron at four gold each.

If the seller had only one unit of iron for sale, the trade would have executed, but the buyer would still be looking for two more iron to buy, at five gold each or better.

7.6 Trade goods

Most cities buy or sell one or more common trade goods, such as wool, beans or pipeweed. It should be possible to earn a decent living buying and selling these items locally.

Some trade goods, such as ivory and myrrh, are extremely rare, and may only be bought and sold in a few cities. These rare items may need to be transported across vast distances to find a buyer. Trade in these items is extremely profitable, and knowledge of such routes is jealously guarded by players.

While ordinary trade goods are produced and sold though city markets on a monthly basis, rare trade items are only produced once each game year.

```
Market report:

    City produces ivory on month 7.

trade    who    price    qty   item
-----    ---    -----    ---   ----
  buy   ev18        2    100   fish [87]
  buy   ev18        4     15   woven baskets [94]
 sell   ev18       19     60   ivory [105]
 sell   ev18       50     28   pipeweed [89]
```

The supply of ivory in this market is replenished once per game year, at the end of month 6. If purchased, no ivory will be available here until the first day of month 7.

8 Loyalty bonds

Nobles are bound to their lords by one of three kinds of loyalty:

contract The noble is being paid for his service

oath The noble has taken an oath to serve his master

fear The noble serves out of fear

The loyalty bond is rated. For example, a character's loyalty could be oath-1, contract-500, or fear-50.

A player's initial character has oath-2 loyalty.

Newly hired nobles have loyalty contract-500. Nobles are paid for their service with the `honor` command. A noble who issues '`honor 50`' will spend 50 gold to raise his own contract loyalty rating by 50 points.

A noble may take an oath of loyalty, pledging one or two Noble Points to secure it. This would yield loyalty oath-1 or oath-2. The `oath` order secures an oath of loyalty for a noble.

Nobles may be terrorized by their masters. The severity of their treatment accumulates in the loyalty rating: fear-10, for instance. Fear loyalty is maintained with the `terrorize` order.

Only one kind of loyalty may be active at a time.

Contract and fear loyalty decay over time. Contract loyalty loses the greater of 50 points or 10% of the current rating each month. Fear loyalty loses 1-2 rating points each month. Oath loyalty does not decay.

Units which fall to contract-0 or fear-0 have a 50% chance of deserting each month.

Nobles serving through contract or fear are susceptible to bribes, which may induce them to renounce loyalty to their lord, and pledge their service to the bribing faction. For details on bribing characters, see [bribe], page 104.

Oath-1 nobles ignore all bribes. There is a persuasion skill which may cause an oath-1 noble to defect, although its use is difficult and rarely succeeds. Oath-1 nobles may reveal their factional affiliation if tortured.

Oath-2 nobles will not renounce loyalty to their lord under any circumstances, nor can they be forced to reveal any information about themselves.

Units may be given to other players with the `swear` order. The loyalty of sworn units does not change when they are transferred. A player may only swear one unit per turn, however.

Summary:

contract	Amount of gold invested in noble
	Decays by max(50, 10% of current rating) each month
fear	Severity of `terrorize` used on noble
	Decays 1-2 points each month
oath	1 or 2 NPs may be invested in an oath bond
	Does not decay

Commands dealing with loyalty bonds:

```
bribe
honor
oath
swear
terrorize
```

9 Stacking

One unit may `stack` under another unit. Two or more units grouped in this way are referred to as a *stack*.

Stacks move together and fight together. Here is a stack of four units:

```
Law Netexus [2020], accompanied by:
  Feasel the Wicked [1109]
  Drakkar the Trader [1752]
  Alion Krysaka [2785]
```

Law Netexus is the stack leader, the top-most unit in the stack.

Only one level of stack depth is shown, so all that can be determined from the location report is that Feasel, Drakkar and Alion are stacked somewhere beneath Law Netexus. The exact arrangement of stacking bonds is not shown.

Feasel might be stacked under Law Netexus, with Drakkar and Alion under Feasel. Or Feasel, Drakkar, and Alion may all be stacked directly beneath Law Netexus.

Generally, such internal arrangements are only important when the stack breaks up. If Drakkar is stacked beneath Feasel, he will stick with Feasel if Feasel drops out of the stack. But if Drakkar were stacked beneath Law Netexus, he would not follow Feasel if Feasel unstacked.

If Law Netexus issues a `move` order, the entire stack will move. If, however, Feasel issues a `move` order, he will first drop out of the stack before moving.

Similarly, if Law Netexus engages combat with the `attack` order, all characters in the stack will fight together. If one member of the stack is attacked, the entire stack will respond in defense.

Multiple levels of internal stacking can be useful if one wants several stacks to join together for a while, but then split apart later into their old arrangements.

Ocean ships may not be stacked together. There is no way to cluster ships into a fleet.

10 Carrying capacity

Men and items are rated for how much they weigh, and how much they can carry, walking, riding or flying.

	weight	walking	riding	flying
man	100	100	-	-
riding horse	300	150	150	-
wild horse	300	self	self	-
warmount	300	150	150	-
knight	400	100	100	-
elite guard	400	100	100	-
ox	1000	1500	self	-
winged horse	300	150	150	150

'man' includes all of the varieties of men, including peasants, sailors, workers, etc. as well as nobles. A knight includes both the man and the horse, hence the 400 weight.

In order to ride, the total riding capacity must cover the weights of all the units that may not ride themselves.

Examples:

man + riding horse
ride capacity is 150 - 100 = 50
walk capacity is 250

man + riding horse + wild horse
ride capacity is 150 - 100 = 50
walk capacity is 250

The wild horse can walk or ride on its own, but will not carry anything.

man + riding horse + ox
ride capacity is 150 - 100 = 50
walk capacity is 1750

The ox may be driven alongside the horse, but will not carry anything when moving so quickly.

A stack will ride if there is enough riding capacity to carry all of the non-riders. Otherwise, the stack will walk.

In order to fly, the total flying capacity for the stack must cover the weights of all units that can not fly themselves.

Stacks which are overloaded beyond their walking capacity will travel slower than normal. A stack loaded at 150% of its walking capacity will take 50% longer to traverse a route. Stacks overloaded to over 200% of walking capacity may not travel at all.

Weights and capacities are always considered for the stack as a whole. One unit may have all the men, and another unit may have all the horses. If they are stacked together, the distribution of items across units is irrelevant.

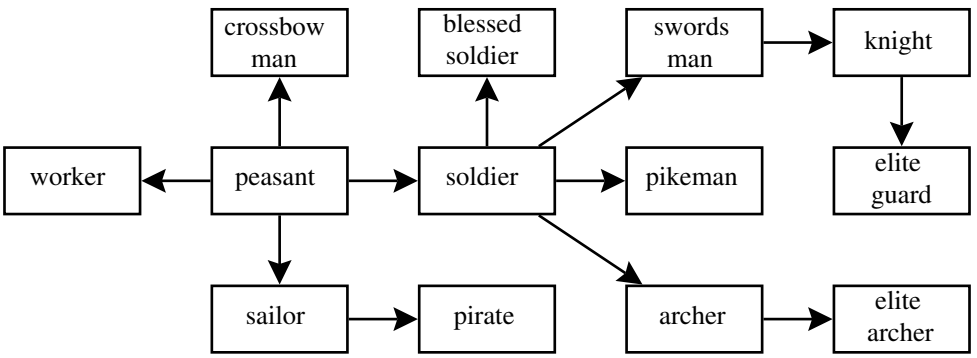
11 Training men

Nobles may be accompanied by different kinds of men. Men may be peasants obtained with the `recruit` order, or peasants who have been trained into other kinds of men such as sailors, soldiers or workers.

The different kinds of men:

peasant [10]
worker [11]
soldier [12]
archer [13]
knight [14]
elite guard [15]
pikeman [16]
blessed soldier [17]
sailor [19]
swordsman [20]
crossbowman [21]
elite archer [22]
pirate [24]

A peasant may be trained into a soldier, sailor, worker, or crossbowman. Soldiers may be given further training to become pikemen, swordsmen, or archers. More advanced fighters may be trained from swordsmen and archers.



Training a man may require that the noble have a certain skill, or possess some item. For instance, training soldiers into swordsmen requires a longsword [74] for each swordsman produced. The training character must also know Combat [121].

Some men may only be trained in certain kinds of locations. Elite guard and elite archers, for instance, may only be trained in castles. Blessed soldiers may only be trained in temples.

Training takes one day per man. Training five archers into five elite archers would take five days, for example. Training ten peasants into ten crossbowmen would take 10 days.

num	kind	skill	input man	input item	where
11	worker	none	peasant [10]		
12	soldier	121	peasant [10]		
13	archer	9579	soldier [12]	longbow [72]	
14	knight	9580	swordsman [20]	warmount [53]	
15	elite guard	9580	knight [14]	plate armor [73]	castle
16	pikeman	121	soldier [12]	pike [75]	
17	blessed soldier	150	soldier [12]		temple
19	sailor	9502	peasant [10]		
20	swordsman	9580	soldier [12]	longsword [74]	
21	crossbowman	121	peasant [10]	crossbow [85]	
22	elite archer	9579	archer [13]		castle
24	pirate	9580	sailor [19]	longsword [74]	ship

A character needs no skills to train a worker.

To train a sailor requires Sailing [9502], a subskill of Shipcraft [120].

Training archers and elite archers requires Archery [9579], a subskill of Combat [121].

Training swordsmen, knights and elite guard requires Swordplay [9580], a subskill of Combat [121].

For order for more information and examples, see [train], page 143.

11.1 Maintenance cost

Men such as soldiers, workers, archers, etc. must be paid in gold monthly or they will leave the service of their noble. Peasants do not willingly leave a noble's service, but will starve if they are not paid. This cost is charged to the noble holding them at the end of each month.

If the noble does not have enough gold to pay his men, he will ask other nobles in his stack (provided they belong to the same player) for gold. Thus, only one member of a stack need carry gold for maintenance costs for the entire stack. Nobles will not share gold with units from other players.

If the noble can only afford to pay some of his men, one-third of those not paid will leave service at the end of the month. The computer chooses which men remain and which leave or starve.

num	kind	cost
10	peasant	1
11	worker	2
19	sailor	2
21	crossbowman	2
12	soldier	2
13	archer	3
16	pikeman	3
17	blessed soldier	3
20	swordsman	3
24	pirate	3
14	knight	4
22	elite archer	4
15	elite guard	5

Note that nobles may `drop` men to release them from service deliberately.

11.2 Making weapons and armor

Weapons and armor are required for the training of some kinds of fighters. Archers require longbows, for instance, elite guard require plate armor, etc.

Weapons and armor are made with the `make` command (see [make], page 123). The Weaponsmithing [9581] subskill of Combat [121] is required to make weapons and armor.

num	item	material
72	longbow	yew [68]
73	plate armor	iron [79]
74	longsword	iron [79]
75	pike	wood [77]
85	crossbow	wood [77]

One unit of the input material may turned into one of the desired items each day. For example, `make` 72 2 would spend two days turning two yew [68] into two longbows [72].

12 Skills

Skills represent knowledge that Olympian characters may know. Shipbuilding, thievery, kidnaping, training soldiers, castle building, mixing potions, and forging magical artifacts are just a few of the possible actions which skills allow a character to perform.

Skills are divided into category skills and sub-skills within those categories. The skill categories are:

```
num  name                    time to learn
---  ----                    -------------
120  Shipcraft               one week
121  Combat                  one week
122  Stealth                 two weeks
123  Beastmastery            two weeks
124  Persuasion              two weeks
125  Construction            one week
126  Alchemy                 two weeks
128  Forestry                one week
129  Mining                  one week
```

There are also six schools of magic. For more details on learning and casting magical spells, see Chapter 13 [Magical Arts], page 59

The category skill must be learned before any of the sub-skills within the category may be known.

With each skill learned, the player will receive a lore sheet describing background information about the skill and how it may be used. Most skills are invoked with the '`use` *skill*' order. The skill lore sheets will give specific information about arguments to `use` and and requirements or limitations for using the skill.

The lore sheets for the skill categories list some of the skills available for study within the category.

For instance, a noble wishing to undertake the study of Shipcraft would first learn the category skill with the `study` command:

```
study 120
```

The Shipcraft lore sheet lists some of the sub-skills available for building and sailing ships. One of these is Sailing [9502], the skill required to control a ship on the ocean. The aspiring captain could then order:

```
study 9502
```

Once Sailing [9502] is known, the lore sheet describing its usage will appear in the player's turn report.

To begin study of a skill requires the following:

- A source of instruction
- Payment of a fee for necessary materials
- Payment of Noble Points to learn advanced skills

12.1 Sources of instruction

The source of instruction may be one of the following:

- The skill is commonly known, and may be studied anywhere once the category skill is known. When a category skill is learned, the player receives a lore sheet for the category in the turn report. The lore sheet lists the commonly known sub-skills which may be studied anywhere once the category skill is known.

  Some sub-skills within a category may not be listed in the category lore sheet. These are not directly studyable, even once the parent skill is known. They must either be learned from a scroll, a rare book, or through research (see [research], page 135).
- The character is in a city which teaches the skill.

  Many cities offer instruction in skills. Shipcraft [120], for example, is commonly taught in port cities.

  The turn report lists skills taught by the location:

  ```
  Skills taught here:
     Alchemy [126]
  ```
- The skill was discovered through `research`, and is listed as '`Partially known`' in the turn report.

  ```
  Partially known skills:
     Improve ship rigging [9999], 0/7
  ```

- The player has a book which teaches the skill. Items which teach skills are shown in the inventory listing:

```
Inventory:
             qty  name
             ---  ----
               1  Old book [6001]

Old book [6001] permits study of the following skills:
   Alchemy [126]
```

12.2 Fee to begin study

A fee of 100 gold is charged when `study` is first issued for a skill. The payment is used to acquire various materials needed for the study and eventual practice of the skill.

12.3 Advanced skills

Some advanced skills require Noble Points to begin study. Each player begins the game with a number of NPs, and receives an additional one each year on the anniversary of their entry into the game. Noble Points may be spent to acquire new nobles, or to learn advanced skills.

Most skills do not require Noble Points to learn. NPs are required for some heroic combat skills and for advanced magical spells.

12.4 Diminishing returns from study

Study is a mentally taxing activity. The more a character studies during a month, the less advancement each attempt will yield.

The first seven days of study each month are fully effective. The second week yields 4/7th the normal advancement, the third yields 2/7th, and the fourth, 1/7th.

days spent studying	yield
1-7	full
8-14	4/7th
15-21	2/7th
22+	1/7th

For example, a character could learn a skill requiring two weeks of study by either issuing four `study` order for one month, or two `study` orders for two separate months.

Another way to view diminishing study results is that a character who studies four weeks per month will progress twice as fast one studying one week per month.

This limit exists on the act of studying, and is not related to the study of a particular skill. For example, a character who studied Shipbuilding the first week, and Combat the second, would have 7/7th effectiveness for learning Shipbuilding, but 4/7th for learning Combat.

Intervening tasks during the month do not affect the tally of study time. One could study the first week, engage in other tasks the second and third, and study again on the fourth week. The study command issued on the fourth week would yield 4/7th.

Study results are kept track of daily. For instance, if a character studies a skill for three days, and then is interrupted, the three days of study are not lost. If the skill requires a week to learn, four more days of study are needed.

A `study` command which continues into the next month will revert to the fully effective rate of return. For instance, suppose a character spent all month studying skills. The last study command doesn't finish at the end of the turn:

```
unit 5499  # Osswid
   # > study 161 (executing for three more days)
```

Although this study command was yielding 1/7th at the end of this turn, at the beginning of the next turn it will be fully effective.

12.5 Experience

Once a skill is known, further `study` of that skill has no effect. Gains in proficiency come with experience using the skill.

Experience is counted for each turn that a skill use is successfully completed. If a skill is used more than once per turn, only the first success will count towards experience.

Projects which take multiple turns to complete, such as shipbuilding or castle construction, only count towards experience when the project is finished.

```
number          level
------          -----
 0-1            apprentice
 2-3            journeyman
 4-5            adept
 6-8            master
  9+            grand master
```

Experience will speed work with some skills. For example, a master shipbuilder will be able to construct a galley somewhat faster than an apprentice. Some skill uses benefit more from experience than others.

12.6 Skills not rated for experience

A few skills are not rated for experience. These may be skills which are not directly used, or ones for which experience has no meaning. If a skill is not rated for experience, the skill level will not be shown in the skill listing.

For instance:

```
Skills known:
   9502  Sailing, apprentice
   9999  Strength bonus
```

Since experience is not applicable to Strength bonus [9999], its level is not shown.

12.7 Teaching

Direct character-to-character teaching is not possible in Olympia. However, it is said that some magicians and alchemists possess the ability to record knowledge on scrolls, which other characters may study from.

12.8 Summary of studying

- The category skill must be learned before a sub-skill within the category may be studied.
- Beginning study of any skill costs 100 gold.
- Payment of Noble Points may be required to begin learning some advanced skills.
- A source of instruction must be available the first time `study` is issued:
  - The skill is taught by the location.
  - The skill is commonly known, so it may be studied anywhere once its category skill is known.
  - The skill was discovered through `research`, so it may now be studied.
  - The skill is taught by a book or a scroll.
- Diminishing returns from multiple studies within a month:
  - First week: 7/7
  - Second week: 4/7
  - Third week: 2/7
  - Fourth week: 1/7
- Once a skill is known, further `study` of that skill has no effect.

12.9 A rough example

This is a rough, heavily edited example of how some study commands might look in a turn report. All of the other details that a real turn report would have were omitted to focus on the study orders.

Turn one:

```
Skills taught here:
   Shipcraft [120]

 1: > study 120
 1: Paid 100 gold to begin study.
 1: Will study Shipcraft for seven days.
 7: Learned Shipcraft [120].

Lore for Shipcraft [120]
------------------------
All skills concerning ocean travel fall under this category.
Shipcraft encompasses the building and repair of ships,
training of sailors, and navigation at sea.

The following skills may be studied directly once Shipcraft
is known:

   num   skill                              time to learn
   ---   -----                              -------------
   9502  Sailing                            one week
   ...
```

Further skills may be found through research.

Turn two:

```
1: > study 9502
1: Paid 100 gold to begin study.
1: Will study Sailing for seven days.
7: Learned Sailing [9502].
```

(It doesn't matter where 9502 is studied, since 120 offers it directly.)

Turn three:

```
1: > study 126
1: Instruction in [126] is not available here.
```

(Need to find a location or book that offers instruction in Alchemy)

Turn four:

```
 1: enter xxxx
 1: Arrival at City of Alchemists [xxxx]

Skills taught here:
   Alchemy [126]

 1: > study 126
 1: Paid 100 gold to begin study.
 1: Will study Alchemy for seven days.

Partially known skills:
   126  Alchemy, 7/14
```

(Alchemy requires 14 days of study to learn, seven of which we have completed.)

Turn five:

```
 1: > study 126
 1: Continue studying Alchemy.
 7: Learned Alchemy [126].

Skills known:
   120  Shipcraft
        9502  Sailing, apprentice
   126  Alchemy
```

12.10 Research

Research attempts to discover sub-skills which are not commonly known or made available when the category skill is learned.

Research is mostly used to discover new magical spells, as few spells are granted when a magic school is learned. However, even common skills such as Shipcraft, Combat and Construction may have hidden sub-skills which can be found through research.

Research for all skill categories except Religion [150] must be performed in a tower, by the tower's owner (the first character inside the tower). Towers make good laboratories for scholarly investigations, and minimize distractions. Other occupants of the tower may not use `research`.

Research into Religion [150] must be performed in a temple, by the temple's owner.

When a category skill is learned, its lore sheet appears in the player's turn report, listing information about the skill as well as sub-skills which may be studied directly based on the parent skill.

An example:

```
Lore for Shipcraft [120]
------------------------
All skills concerning ocean travel fall under this category.
Shipcraft encompasses the building and repair of ships,
training of sailors, and navigation at sea.

The following skills may be studied directly once Shipcraft
is known:

   num   skill                              time to learn
   ---   -----                              -------------
   9502  Sailing                            one week
   ...

Further skills may be found through research.
```

The last line ('`Further skills` ...') indicates that there are sub-skills of Shipcraft which are not mentioned in the lore sheet. These hidden skills may represent rare or hard-to-learn knowledge, or perhaps technology which has not yet been discovered.

Study of these skills is not possible based simply on knowledge of the parent skill. Sailing [9502] can be learned by a character, no matter where he is, once Shipcraft is known. Hidden Shipcraft skills, however, must be learned in other ways, even if the character has learned their entity numbers from other players.

There are two possible ways such hidden skills may be learned:

1. Through a rare book or scroll which offers instruction in the skill.
2. Through research.

Research is the more difficult choice. However, to learn rare sub-skills there may be no alternative. Perhaps there are no players who already know the rare sub-skill, and so cannot record scrolls to instruct others. Or if there are, they choose to keep their knowledge secret.

Research incurs a fee of 25 gold to pay for miscellaneous materials and costs.

Each week of research yields a 25% chance that a new skill will be discovered.

If research is successful, the new sub-skill will be added to the character's partially known skill list. It then must be studied in order to become fully known and usable.

For example:

```
1: > research 120
1: Will research Shipcraft for seven days.
7: Research uncovers a new skill:  Improve ship rigging [9999].
7: To begin learning this skill, order 'study 9999'.

Partially known skills:
   9999  Improve ship rigging, 0/7
```

The '`0/7`' qualifying the new sub-skill in the partially known skill list indicates that seven full days of study are required to learn the skill, and none of them have yet been completed.

```
1: > study 9999
1: Paid 100 gold to begin study.
1: Will study Improve ship rigging for seven days.
7: Learned Improve ship rigging [9999].
```

The `research` order is no longer used on the new skill once it becomes partially known. Research may continue to be used on Shipcraft [120], however, to seek out more hidden sub-skills.

Category skills may not be learned through research.

13 Magic

> Magic is a dead cat in an oil-stained burlap bag.
> Magic is a smelly old man, despised but feared by his neighbors.
> Magic is what the king turns to, when his soldiers fail.
> — a long-dead wise man of Areth Pirn

Magic is the dark art by which events are influenced outside of the normal boundaries of cause-and-effect. Rather than the glamorous ideal of shining wizards casting powerful fireballs at wicked foes, the reality of magic instead tends to be base, tedious work which earns few friends.

Hated and feared, the magician pursues his craft out of the sight of men. Like wisps of smoke rising from an ember cast into dry straw, so the mage's spells slowly take hold, woven with secret knowledge and foul ingredients.

The casting of a magical spell is accomplished with three ingredients: Knowledge of the spell, possession of any items necessary to fuel the spell, and a sufficient level of magical aura to perform the ritual or ceremony.

13.1 Aura

Aura is a mystical force necessary to cast spells. Powerful mages will have a high aura rating, while apprentice sorcerers may only command a few points worth of aura.

Characters are rated for their current aura level and their maximum aura level. With each magical spell learned, a character will gain one point of maximum and current aura. Current aura is depleted by casting spells, and is naturally replenished at a rate of two points per turn. Current aura will increase until it reaches the maximum aura level. Other ways of gaining current aura may be found as the mage researches his craft.

Spells are rated on the amount of aura which the casting mage must possess. Minor spells may demand only one point of aura to cast. Powerful spells may require a current aura level of ten or higher.

13.2 Magician status

Characters receive a rating in the turn reports based on their magical abiliity:

maximum aura	label
6-10	conjurer
11-15	mage
16-20	wizard
21-30	sorcerer
30+	??

For example:

```
Osswid the Brave [5639], wizard, with three workers
```

The Basic magic [160] spell Appear common [9104] allows magicians to prevent this label from displaying.

13.3 Required Items

Many spells will require the magician to possess a rare or obscure item in order for the cast to succeed. Many of these items exist, of interest chiefly to sorcerers. Roots from plants found only in dense forests, bat's wings, and a dark blue powder which produces a brilliant cobalt flame when burned are only a few. Usually the required item is consumed by the attempt to cast the spell.

13.4 Study of Spells

Magic is divided into six schools of study:

num	name	time to learn	
160	Magic	two weeks	1 NP req'd
161	Weather magic	three weeks	1 NP req'd
162	Scrying	three weeks	1 NP req'd
163	Gatecraft	three weeks	1 NP req'd
164	Artifact construction	four weeks	1 NP req'd
165	Necromancy	four weeks	2 NP req'd

Magical spells are simply sub-skills of one of the magical skill categories.

An aspiring mage will issue the `study` order to learn the basics of a particular school of magic, known as the category skill. For example, one wishing to pursue knowledge of Magic [160] would order:

```
study 160
```

Once the category skill has been learned, the mage will receive a lore sheet listing some of the known spells of that school. The magician may then attempt to learn these spells through study.

A character must know the category skill for a school of magic before a spell in that school can be known.

Only some of the spells in the each school are commonly known. The more rare, obscure or powerful spells will need to be discovered via `research` or by finding magic scrolls describing them.

Knowledge of individual spells in a school of magic is not possible without having learned the category skill.

13.5 Schools of Magic

Magic [160]
: The most common and well-known of the magical schools, Magic nonetheless has many useful and powerful spells.

  Since many cities offer instruction in Magic, and several useful spells may be learned quickly, apprentice mages often begin their studies here.

Weather magic [161]
: The study of spells to control the elements. Advanced weather magicians are said to research forgotten elf-lore in the hopes of finding tools to battle evil.

  Weather magic is taught in Nimbus, Stratos and Aerovia, the three cities of the Cloudlands.

Scrying [162]
: Scrying is the study of magical far-seeing, the ability to view images or learn information at a distance.

  Scrying is taught in the Faery City.

Gatecraft [163]
: The study of ancient portals of teleportation which long ago connected distant regions of the land. Knowledge of this lost art tends to be difficult to acquire and jealously guarded when found.

Artifact construction [164]
: The realm of very advanced sorcerers, artifact construction concerns the making of physical items to focus, amplify or otherwise enhance a mage's power. Most spells in this school require the magician to possess high levels of aura.

Necromancy [165]
: The darkest of the Dark Arts, necromancy involves trafficking with undead or demon spirits to gain knowledge. Hated by civilized men, feared by other magicians, the necromancer seeks power and domination over the physical world.

  Necromancy is taught in the City of the Dead, in Hades.

Monsters may sometimes be found guarding ancient books which teach the rare magical skill categories (see Section 5.16 [Guard monsters], page 32).

13.6 Religion

Learning the skill category Religion [150] labels the character as a priest. Religion [150] requires 1 Noble Point and three weeks to learn. Religion [150] may be studied in any temple. Religion is not taught by cities. Skills in the Religion category are known as prayers.

Temples yield 100 gold in offerings per month to their owner, if the owner is a priest. Temples may be built anywhere, except inside another building.

Research into Religion [150] must be performed in a temple rather than a tower.

14 Battles

Combat in Olympia occurs when one stack attacks another stack.

A unit will be defended by any other characters it is stacked with. Thus, a unit which is part of a stack cannot be attacked alone. No matter which unit is specified in the `attack` order, the entire stack containing the unit will respond.

Only the leader of a stack may initiate combat. If another unit in a stack wants to issue `attack`, it must first drop out of the stack.

Combat involves nobles, who possess strong heroic fighting abilities, and fighters who may accompany them. Fighters include soldiers, pikemen, swordsmen, knights, elite guard, crossbowmen, archers, and elite archers.

14.1 Resolution of battle

Each kind of fighter has an attack and defense rating:

	attack	defense	missile
peasant	1	1	
worker	1	1	
sailor	1	1	
soldier	5	5	
pikeman	5	30	
swordsman	15	15	
pirate	5 (15)	5 (15)	
knight	45 (20)	45 (20)	
elite guard	90 (65)	90 (65)	
crossbowman	1	1	25
archer	5	5	50
elite archer	10	10	75

Numbers in parenthesis are for shipboard combat.

Nobles have strong innate heroic abilities, and so even an unarmed, untrained noble is a formidable opponent on the battlefield.

A noble is rated (attack=80, defense=80, missile=0).

A blessed soldier fights with the same attack and defense values as a regular soldier, but has a 50% chance of surviving a hit.

A pirate fights (15,15,0) on a ship, but only (5,5,0) on land.

Knights and elite guard receive a -25 penalty to both their attack and defense ratings when fighting on a ship or in a swamp province.

During combat, a random man is chosen to attack. If the attack succeeds, the target is removed from battle. This continues until the battle is over.

More specifically:

1. A man is picked at random to attack. This man may be from either side in the battle.

   For example, if there are 10 men on side A, and 20 men on side B, there is a 1/3 chance that one of A's men will be chosen, and a 2/3 chance that one of B's men will be chosen.

2. The attacker chooses an enemy target at random. The target may be any of the men in the enemy's front units. A ship or building housing the enemy also counts as a potential target.

   In other words, if there are five men in a tower, each man has 1/6 chance of being selected as a target, and the tower also has 1/6 chance of being paired with an attacker.

3. The chance that the attacker will score a hit against the target is:

   A = attacker's attack rating
   B = target's defense rating

   A / (A + B)

   For example:

   A(attack=90) vs B(defense=45) A has a 2/3 chance of killing B
   A(attack=90) vs B(defense=90) A has a 1/2 chance of killing B

4. If the attacker scores a hit against a noble, the noble will receive a random wound of 1-100 health points. Note that there is a 1% chance that a perfectly healthy noble will be instantly killed, and a greater chance that a previously wounded noble will die (see Chapter 6 [Health], page 33).

   Wounded nobles do not continue fighting, even if their wounds are minor.

   If a hit is scored against a fighter (soldiers, pikemen, archers, etc.), the fighter is killed. However, a blessed soldier has a 50% chance of surviving a hit.

   A man successfully attacking a building or a ship will cause one point of damage to the structure. A siege engine attacking a building will cause 5-10 points of damage.

5. Repeats until a side breaks. A side breaks when its total offensive plus defensive value falls by 50%.

   For instance, a noble plus two pikemen has an offensive value of (80+80) + (5+30) + (5+30) = 230 with a break point of 115.

A noble plus two knights has an offensive value of (80+80) + (45+45) + (45+45) = 340 with a break point of 170.

Therefore, a noble with two pikemen will continue to fight if the pikemen are killed. However, a noble with two knights will be declared the loser if both knights are killed.

The offensive value used is either the attack or the effective missile rating, whichever is higher.

Note that in step 4, if the target is not hit, the target does not immediately retaliate against the attacker. Only when the target is randomly selected to make an attack will it have a chance to score a hit.

The winner in combat will attempt to take prisoners. The chance that a given defeated unit will be taken prisoner is proportional to the sizes of the remaining forces.

(Taking prisoners in battle and claiming loot requires many soldiers to run after the fleeing enemy. Thus, the chance of success is based on numerical advantage rather than combat skill.)

```
1:1     25%
2:1     50%
3:1     75%
```

If the winning side outnumbers the defeated force by 2:1, there is a 50% chance that a given defeated unit will be captured. Defeated units which are not captured retreat from battle. If they are occupying a building or are located in a city, they may flee into the outlying province.

The victor always has at least a 25% chance of taking a unit prisoner, but no better than a 75% chance.

The victor will claim the defender's position in the location list if it is better, or will move into the defender's structure, ejecting the losing force. (The attacker may specify a flag to the `attack` order to inhibit this behavior.)

Prisoners are stripped of their belongings by the victor, including any men accompanying the prisoners, such as workers or peasants. The stack leader of the winning force receives all of the loot from the battle.

14.2 Front and rear

Units may issue the `behind` command to declare whether they will line the front or the rear in battle. Rear units do not become targets for the enemy until all of the units in the rows in front of them have been killed. Only missile fighters, such as archers, may attack from the rear.

The leader of each stack (the top-most unit in the stack) will be the last unit to receive hits, regardless of its `behind` status.

14.3 Missile Attacks

Units in the rear may attack with their missile rating. Units in front attack with either their missile rating, or their attack rating, whichever is higher.

Thus, a noble with rating (attack=80, defense=80, missile=40) will do an attack of 40 when in the rear, and an attack of 80 when in front.

If the location of the battle is windy, all missile ratings are halved. If it is raining, the missile rating for archers and elite archers is reduced to zero.

If a fighter's missile rating is zero, it can not attack from the rear. In this case, the fighter will be in the front and use its attack rating like a regular soldier.

14.4 Fortifications

Structures which may aid fighters in battle are rated for their defensive bonus.

```
Castle Imperius [gx56], castle, defense 25
```

During battle, the fortification rating is added to the defense number for the men who fit inside the structure.

structure	men protected
Castle	first 500
Tower	first 100
Galley or roundship	first 50
Other structures	first 50

Attacking fighters may randomly select the structure instead of an enemy fighter. The attack is resolved in the same way as for two fighters, using the attacker's attack rating and the structure's defense rating. If the attacker is successful the structure's defense rating will be lowered by one point.

Once the defense rating reaches zero, further hits will cause the building to become damaged. A fully damaged building (100% damage) will collapse, ejecting its occupants.

Siege engines always select the structure as a target.

engine	attack	defense	missile
catapult	25	200	25
battering ram	30	250	
siege tower	30	250	

Siege engines do 5-10 points of damage to the structure per hit.

Siege engines are not used in combat at sea.

14.5 Item bonuses

Nobles may possess items which grant attack, defense or missile bonuses in combat. Only one item may be wielded for each category. If the noble possesses multiple items with bonuses in the same area, the item with the largest bonus will be chosen.

For example, suppose Osswid had the following items:

Shield of Achilles [fx78]	+25 defense
Sword of Death [gl23]	+10 attack
Mithril Axe [wt29]	+15 attack
Magic javelin [ht02]	+5 missile

Osswid would wield the javelin and axe, and wear the shield. He would not use the Sword of Death.

Nobles automatically use items with combat bonuses in battle. No special orders are needed to wield them.

14.6 Prisoners

Characters may become prisoners by losing to an enemy in battle, or by being kidnaped (kidnaping is a subskill of Stealth).

Since prisoners are unable to report where they are and what they are seeing, they do not contribute to the turn report of their faction. The player's turn report will show that that a unit is being held prisoner, but little else.

Prisoners will not execute any orders while they are in captivity. Queued orders will remain pending, but none will be processed.

Prisoners when spotted appear as stacked units, marked with the '`prisoner`' string:

```
Seen here:
   Kosar the Indefectible [2022], with six peasants, one archer,
   two soldiers, accompanied by:
      Alion Krysaka [2785], prisoner
```

Unstacking a prisoner sets them free. Kosar could free Alion by ordering '`unstack 2785`'.

Prisoners may be transferred between units with the `give` command. Kosar could transfer Alion to Osswid [501] by ordering:

```
give 501 2785
```

14.7 Prisoner escapes

Prisoners are always on the lookout for ways to escape. Units holding prisoners can reduce their chances by remaining inside a structure, not transferring prisoners with `give`, and not traveling with prisoners.

Each week (four times each game turn), a prisoner being held by a unit which is outside of a building has a 2% chance of escaping. A prisoner being held by a unit which is inside a structure, such as a castle, tower, inn or ship, has a 1% chance of escaping.

Each time a prisoner is transferred with the `give` order, there is a 2% chance of a escape. Also, each time a unit holding a prisoner engages in travel which takes longer than one day, there is a 2% chance that the prisoner will be able to get free. Thus, short movement, such as entering or exiting a building, will not give the prisoners additional opportunities for escape, but traveling between provinces with prisoners will.

Prisoners inside building or sub-locations will flee out into the surrounding location upon gaining their freedom. Escaped prisoners on ships will leap over the side and swim to a nearby shore.

15 Permissions and declared attitudes

Commands dealing with permissions:

```
admit
hostile
neutral
defend
default
```

Attitudes can be declared by or for either specific units, or an entire faction. For instance, player [613] could declare a permission or attitude for player [555], or a specific attitude for individual units within player 555's faction.

Declaring a permission for a player works so long as the player's units are not concealing their faction identity with Conceal faction [9591], a subskill of Stealth [122].

15.1 Allowing entrance and stacking

By default, a unit may not stack with a character belonging to another faction. A unit is also denied entry to a building or ship controlled by another player.

Players may allow units from other factions to stack with them or enter buildings or ships they control with the `admit` order.

15.2 Combat attitudes

A unit may have one of four combat attitudes to another unit:

`hostile`	Attack on sight.
`defend`	Defend other unit if attacked.
`neutral`	Do nothing if other unit is attacked.

`default` Neutral to units in other factions; Defend units in the same faction unless either one is concealing its lord.

Every character, and player faction entity, keeps three lists of units or other factions which have attitudes declared towards them. A unit is either on the `hostile`, `defend`, or `neutral` list. If a unit does not appear on any of the three lists, it has attitude `default`.

Example:

```
player          778                     816
                  hostile 816

units           4205                    6499
                4600                    6530, concealing lord
                                        6599
```

Player 778 has declared player 816 hostile. One of 816's characters is concealing its lord.

If 4205 or 4600 run into unit 6499, they will attack it on sight. However, since 6530 is hiding its affiliation with 816, it will not be attacked on sight.

If 6499 is attacked and both 6530 and 6599 are present, 6599 will aid in the defense, but 6530 will not, because that might give away its affiliation.

If player 816 wanted 6530 to defend the faction's units anyway, either 816 or 6530 should issue the order '`defend 816`'. This would override the default attitude of units in the faction to one another.

Attitude toward units is considered before attitude toward the unit's faction. Thus, one may declare a faction hostile, but exclude certain units within the faction by specifically declaring them neutral.

A unit must be the top-most character in its stack to aid in defense. If a unit joins a combat because of `defend`, it will bring its entire stack along, even if the other members of the stack have not declared a `defend` attitude.

Defenders only help when units are attacked, not when they initiate attacks. For example, if A has declared '`defend` B', and B attacks C, A will not help B, even if B loses the battle.

Characters declared `defend` to units which are guarding a province against pillaging will aid the guards if they are attacked, either explicitly with `attack`, or implicitly via '`pillage 1`'.

Units which joined a combat because of a `defend` declaration are shown with the qualification '`ally`' in the combat report.

15.3 Buildings

The following structures may be built by characters with the Construction [125] skill:

kind	effort	material	skill	where
inn	300	75 wood	125	province or city
mine	500	25 wood	125 or 129	mountain or rocky hill
temple	1,000	50 stone	125	anywhere
tower	2,000	100 stone	125	anywhere
castle	10,000	500 stone	125	province or city
galley	250	50 wood	9503	ocean port
roundship	500	100 wood	9503	ocean port

The term '`anywhere`' means a province, city, or other sublocation. Buildings may not be built inside other buildings, with the exception of towers. Up to six towers may be built inside a castle.

Effort is in worker-days.

The builder must know the required skill, have at least three workers, and possess at least one-fifth of the necessary construction materials in his inventory.

To start building, unstack from beneath other characters and issue one of the following `build` orders:

`build inn "`*name of inn*`"`

`build mine "`*name of mine*`"`

`build temple "`*name of temple*`"`

`build tower "`*name of tower*`"`

`build castle "`*name of castle*`"`

For example, a character who wanted to build a tower would need to know Construction [125], have at least three workers [11], and possess at least 20 stone [78].

The construction materials will immediately be deducted from the builder's inventory and put to use. The builder and his workers will be placed inside the new structure.

The remaining construction materials are paid as work on the structure progresses. A second fifth is required as the building becomes 20% complete, the third fifth at 40%, etc. Construction halts if the builder runs out of materials.

The building is completed when the required number of worker-days is invested in construction.

To resume construction of a partially completed building, first enter the structure, then issue the appropriate build order (such as `build inn`, `build mine`, etc.)

Mines may only be built in mountain provinces or on rocky hills. Only one mine per location is allowed. Mines may also be built by characters with the Mining [129] skill.

Only one castle may be built in each province. The castle may be built either in the province or a city, if the province has one. Castles may not be built sub-locations other than cities.

15.4 The first tower you build is free

Since towers are essential to research, the first tower a player builds requires no workers, no stone, and takes only one day. The player should choose an appropriate site for the tower, and issue a regular `build tower` order.

15.5 Inns

Inns must be built in provinces. A good site for an inn is just outside a city, as the inn will benefit from the patronage of many travelers.

The restive environment inns provide help wounded nobles to heal. Nobles located in an inn have an 10% increased chance of fighting off infection each week.

Inns generate income each month from the visitors who stop for a meal and a pint of ale, or spend in the night in one of the inn's rooms. (Nobles who enter are not directly charged. Other patrons are anonymous; the only indication of their presence is the income the inn generates.)

Inns generate between 50 and 75 gold per month. Looting and pillaging scares away customers, and so lowers the inn's income. If more than one inn is built in a province, the profits are split between them.

Income is paid to the owner of the inn at the end of each month.

15.6 Mining

A mine is a deep shaft or tunnel which allows workers to extract valuable resources from the earth, such as iron and gold. At most one mine may be built in each mountain province or rocky hill.

A new mine has an initial depth of one. The mine shaft becomes deeper as characters use it to obtain natural resources. The shaft will become one level deeper for every three uses of a mining extraction skill.

As the depth of the shaft increases, the mix of resources obtainable changes. Iron is usually found nearest the surface. As one proceeds deeper, gold may be found in higher quantities. Other rare elements may be found by going deeper still.

The deeper a mine becomes, the more frequently cave-ins or other accidents will occur. With each accident, the mine's damage percentage will rise. If not attended with `repair`, the mine will eventually collapse. Once a mine collapses, it remains in the province for one game year (eight game months). Characters may not enter or use a collapsed mine. After the year has passed, the collapsed will vanish, and a new mine may be built in the location.

16 Ships

Players may build two kinds of ships in Olympia: galleys and roundships.

A galley, also known as a warship, is a slender rowed vessel. Galleys require 14 sailors as oarsmen for travel, and may carry up to 5,000 units of cargo. The oars permit the ship to be maneuvered regardless of wind conditions.

Galleys may be fitted with rams, which are used to crush the sides of enemy ships in battle.

The roundship, also known as the merchantman, is a deep, wide sailing ship, usually with one or two masts and steered with great, oar-like paddles. The large cargo space makes it well suited for trade and extended ocean travel.

With favorable winds, roundships will make better time than galleys, even when fully loaded. Roundships require a crew of eight sailors travel, and may carry up to 25,000 units of cargo.

The Shipcraft [120] skill category allows nobles to train sailors, and has sub-skills for building and sailing ships.

ship	capacity	sailors
galley	5,000	14
roundship	25,000	8

Ships may be damaged while sailing by storms, or by submerged rocks in coastal waters. Damaged ships may be repaired with the `repair` command. Repairing a damaged ship requires one unit of pitch [261].

The capacity of a damaged ship is a reduced in proportion to the amount of damage. A galley with 10% damage may only carry 4,500 units of cargo.

16.1 How to sail

The Sailing [9502] subskill of Shipcraft [120] is required to pilot a ship. Shipcraft may be learned in any port city. For information about piloting galleys and roundships, see [sail], page 136.

16.2 Making ships

Construction of ships requires a character to have the Shipbuilding [9503] skill, a subskill of Shipcraft [120].

ship	effort	material
galley	250 worker-days	50 wood
roundship	500 worker-days	100 wood

To begin construction of a ship, the shipbuilder should unstack from beneath other characters and issue one of the following `build` orders:

`build galley` *name of galley*

`build roundship` *name of roundship*

One-fifth of the lumber will immediately be deducted from the shipbuilder's inventory and put to use building the new ship. The shipbuilder and his workers will be placed inside the new ship.

At least three workers are needed to begin construction of a ship.

Until the ship is completed, it will be shown as '`in progress`':

```
Ships docked at port:
   HMS Pinafore [ih39], galley-in-progress, 28% completed, owner:
      Osswid the Constructor [5499], with five workers
```

The remaining construction materials are deducted from the builder's inventory as work on the ship progresses. A second fifth of the lumber is required when the ship becomes 20% complete, a third fifth when the ship becomes 40% complete, etc. Construction halts if the builder runs out of materials

For example, a noble who wanted to build a galley would need to know Shipbuilding [9503], have at least three workers, and start with at least 10 wood [77]. (40 more wood is required to bring the galley to completion).

As soon as the required number of worker-days has been invested in construction, the ship will be christened and declared seaworthy.

To resume construction of a partially completed ship, first enter the ship, then issue the either `build galley` or `build roundship`.

Ships may only be built in locations with a passable route to an ocean.

16.3 Operating a ferry

There are several commands useful for operating a commercial ferry:

`fee`	Sets the fee passengers will be charged
`board`	Passengers use this to pay their fee and board a ferry
`unload`	Unload passengers once the destination is reached
`ferry`	Signals passengers waiting in port that they may board

The captain of a ferry must issue '`fee` *gold*' command to set a fee which will be charged to passengers wishing to `board` his ship. The fee is expressed as how many gold pieces per 100 weight of the passenger's stack will be charged.

For instance, if the captain wanted to charge 1/2 gold per unit weight of the passengers stack, he would issue '`fee 50`'. (a 50 gold fee for every 100 weight).

The fee is a property of the captain, not the ship.

A character may clear the fee with the '`fee 0`' order. If no fee is set, then the ship is not considered to be operating as a ferry, and characters are may not use the `board` order to enter the ship.

Passengers issue '`board` *ship*' to board a ferry. The order will fail if the ship is not present, or if it is not operating as a ferry (the captain of the ship has no fee set). `board` will cause the character to pay the captain the required boarding fee, then move the character's stack onto the ship.

The captain shouldn't issue an `admit` order to let characters on board who will be paying to take the ferry. Otherwise they could board the ship with `move` instead of `board`, bypassing the ferry fee.

16.4 Ferry synchronization

Suppose now that the captain has a ship, has set a fee, announced his service in the *Olympia Times* and with `post`, and is now ready to ferry passengers. What should he tell them? How will the synchronization work?

Passengers should travel to the port the ferry will be arriving at and issue '`wait ferry` *ship*'.

When the ship arrives at port, the captain should order `unload` to eject his current load of passengers, then `ferry` to signal any passengers waiting in port that they may now board.

Note that passengers should not use `wait ship` to wait for the ferry. Otherwise, they will attempt to `board` as soon as the ship reaches the port. The captain's `unload` order may not have executed at this point. In this case, `board` orders might fail because there won't be any room to enter the ship until the existing load of passengers has disembarked.

Example:

```
Posted by Captain McCook [3402]:
"Captain McCook's ferry to Drassa departs each
 Sunsear on the 15th.  The fee is 1 gold/wt.
 Issue WAIT FERRY ht34 then BOARD ht34 for
 ferry service.   Arrr!"
```

Captain McCook's orders:

```
> fee 100                 # 1 gold/wt. is our fee
> sail ...                # arrival at port
> unload                  # unload current passengers
> wait day 15             # wait until stated time of departure
> ferry                   # sound our horn
> sail ...                # on to the next port
```

Passengers wishing to travel on McCook's ferry:

```
> wait ferry ht34             # [ht34] is McCook's ship
> board ht34                  # pay our gold and embark
> wait loc destination        # Captain McCook will unload us at the
                              # end of the journey, so wait until we
                              # find ourselves in the destination city.
```

16.5 Control of buildings and ships

The owner of a building or a ship is the first character shown inside. This will be the first character to have entered the building or ship, unless another character attacked the previous owner and taken position at the head of the list.

For example, assume the following characters are on a ship, the HMS Pinafore [4000]:

```
Candide the Captain [1269], with ten sailors
Osswid [5499], accompanied by:
   Feasel the Wicked [1109]
```

Since Candide is first in the ship's character list, he is the owner of the ship. Only Candide may issue sail orders, or change the ship's name. If Candide were to leave, Osswid would become the new captain of the ship.

The owner of a building or ship may determine who may enter with the `admit` order. The default is to refuse entry to units from other factions.

If the first character inside a building leaves, he is no longer the owner.

A building is a castle, tower, inn, temple or mine.

17 Province control and ownership

The section for empire-builders: How to own land and gain status.

17.1 Quick overview

Each province and city generates an amount of gold each month. This gold is known as its *tax base*. The size of the tax base is determined by the civilization level of the province. This gold may be pillaged, taxed by a castle, or taxed by a garrison. It does not accumulate if left uncollected at the end of a turn.

A castle automatically collects all of the gold in its province.

The '`garrison` *castle*' order installs a group of at least ten soldiers in a province to claim it and guard against pillaging. Garrisons must be bound to a castle.

A garrison pays maintenance for its members from the province tax base, then forwards 1/5 of the remaining gold to its castle.

The castle owner gains status from the number of provinces under control:

```
provinces       rank
---------       ----
   1-3          lord
   4-7          knight
   8-11         baron
  12-15         count
  16-19         earl
  20-22         marquess
   23+          duke
  region        king     (region must have at least 15 provinces)
```

A noble may `pledge` to another noble, granting status and control of owned provinces. The status of a noble who pledges is the smaller of the original status or one below the rank of the pledge target:

```
new status = min(original status, one below rank of pledge target)
```

Control of a province allows one to change its name or the name of any of its sublocations, take items from the garrison, and issue decrees to watch for certain units, or to attack specified units on sight.

The castle continues to receive the income from garrisoned provinces, even if the castle's owner is pledged to another noble.

Every noble in the pledge chain shares control of the garrisoned provinces. In other words, a castle owner may pledge to a noble, who in turn may pledge to a third noble, etc. Thus a province may have any number of rulers.

Visitors to a province are informed of the castle to which the garrison is bound, and the top-most ruler in the pledge chain (which may simply be the owner of the castle):

```
Province controlled by Amber Keep [cy09], castle, in Forest [cj12]
Ruled by Erekosse [5210], baron
...
Seen here:
  Garrison [780], garrison, on guard, with ten soldiers
```

17.2 Tax base

Each province and city generates a tax base each month. The amount of gold fed into the tax base is determined by the civilization level of the province:

civ level	tax gold
wilderness	50
civ-1	200
civ-2	250
civ-3	300
civ-4	350
civ-5	400
civ-6	450

The tax base support garrisons, can be collected by castles, or seized through pillaging.

Pillaging and opium consumption reduce the future tax base of a province (see Section 17.11 [Pillaging], page 85 and Section 17.12 [Opium], page 86).

A city's tax base is added to the province's tax base at the end of the turn. If the city is pillaged during the month, the amount transferred to the province will be diminished.

Gold left in the province at the end of the turn does not accumulate.

17.3 Castles

Castles are the foundation of land ownership. A castle provides its owner with taxes from the province it is located in, as well as from garrisons in other provinces which are bound to the castle.

The owner of a castle automatically receives the remaining tax base from the castle's province at the end of each month. If a garrison is stationed in the same province as a castle, the garrison will first pay maintenance from the province's tax base, then the castle will collect whatever is left.

Each province may contain only one castle. The castle must be built in the outer province or in a city, if the province has one. (Tax revenue for the castle is the same no matter where it is built.) Castles may not be built inside other sublocations.

A castle alone is not sufficient to rule a province. A garrison must be stationed outside the castle in the province to protect it.

17.4 Garrisons

Garrisons are groups of men who are stationed in provinces to protect them, and collect taxes in the name of a castle. Garrisons must be created with the `garrison` order, and must be bound to a castle located in the same region.

For example, suppose that the region Lesser Atnos had 20 provinces. One of these provinces contains Amber Keep [cy09]. A garrison bound to Amber Keep could be stationed in each of the 20 provinces (including the province containing the castle itself).

Continuing the example, garrison units not in the Lesser Atnos region could not be bound to Amber Keep. The castle a garrison is bound to must be in the same region.

Garrisons can be bound to any castle in the region. If Lesser Atnos had two castles, some of the garrisons could be bound to one, and the rest to the second castle.

17.5 More about garrisons

Garrisons are established with the '`garrison` *castle*' order. Ten soldiers are required to create a garrison. The `garrison` order must be issue at the outer level of a province; one can't establish a garrison while inside a city, building or other sublocation.

The garrison pays the maintenance cost of its men directly from the tax base of the province. One-fifth of the remaining tax base is forwarded to the castle the garrison is bound to.

For example, a garrison of ten soldiers would require 20 gold per month to support. This would leave 280 gold remaining in a typical province. 20% of this, or 56 gold, would be forwarded to the owner of the garrison's castle.

Example:

```
> garrison cy09
Installed Garrison [780], garrison, on guard, with ten soldiers
```

Visitors to this province would see:

```
Province controlled by Amber Keep [cy09], castle, in Forest [cj12]
Ruled by Erekosse [5210], baron
```

Note that Erekosse may be located inside the castle, or the castle's owner may have pledged service to him, in which case Erekosse could be anywhere.

17.6 Garrison reports

Garrisons do not provide full location reports to their owners. They do notice any resource depletion activity, such as timber cutting or mining, as well as any large or unusual parties which enter their province. This includes any stack of five units or more, any party of 20 or more men, and most monsters or wild beasts.

Garrisons do not monitor activity in hidden locations, even if the players who rule over the garrisons have discovered the hidden locations.

The '`decree watch` *who*' order may be given by a ruler to instruct all garrisons to watch for a particular unit. This is useful for locating individuals who would otherwise go unnoticed by the garrisons.

17.7 Referring to garrisons

Since a province may only have one garrison, garrisons may be referred to without knowing their entity number. The keyword `garrison` will match the province's garrison, if there is one.

Examples:

```
give garrison 12 5
attack garrison
```

17.8 Status

The number of provinces a noble controls determines his status or rank:

provinces	rank
1-3	lord
4-7	knight
8-11	baron
12-15	count
16-19	earl
20-22	marquess
23+	duke

In addition, if a character has control over every province in a region, and the region contains at least 15 provinces, then the character is given the rank of king.

Provinces may be directly owned, if the noble is the owner of a castle, or indirectly, through other pledged nobles.

17.9 Pledging land

A noble may `pledge` his lands to another noble. This grants the pledge target status by increasing the number provinces he may rule over.

For example, suppose there are two castle owners, Osswid and Feasel. Osswid has garrisoned six provinces, and Feasel has three. Osswid is therefore a baron, and Feasel is a lord.

If Feasel and Osswid both pledge to Candide, Candide would attain the rank of Count. Osswid and Feasel would remain at the same rank in this example.

Candide would receive garrison reports for all provinces which Osswid and Feasel control. He would have the same privileges in the controlled provinces: he could take items from the garrisons, alter the names of the provinces or their sublocations, and issue watch and hostile decrees.

However, the income generated by the provinces would continue to be forwarded to the castles. No extra income goes to the pledge target.

17.10 Status after pledging

The status of a noble A who is pledged to another noble B will be either A's original status, as determined by how many provinces he controls, or one rank below B, whichever is lower.

For example, a noble with 5 provinces who pledges to a king will remain a baron. However, if pledged to another baron, the noble's rank would fall to lord.

17.11 Pillaging

Gangs of ten or more men may use the `pillage` command to seize loot from a province or a city. Pillaging yields 1/3 of the tax base for a location, and destroys another third. For example, a location with a tax base of 300 gold would yield 100 gold to a pillager, and be left with 100 gold (100 was destroyed by the act of pillaging).

Pillaging has a harmful effect on the future tax revenue of the location. The more a location has been pillaged, the lower its tax base. Provinces and cities take as many months to recover as

they have been pillaged. For example, a city which was pillaged for five months in a row would take five months to return to its normal tax base.

Pillagers must first defeat any units guarding the province, including any province garrisons.

17.12 Opium

Opium is produced in swamp regions, and consumed by markets in desert, plain, forest, and mountain provinces. All markets (except those in swamp regions) have some level of opium demand. However, this demand will not be visible in the market report at low levels.

Satisfying opium demand in a market will cause the next month's demand to be higher. As peasants become addicted to opium, the increased demand will be shown in the location's market report. If no opium is sold to a market, the demand will fall.

Opium adversely affects the city's tax base. The more opium the market buys, the more tax revenues will be reduced.

Appendix A Summary of important tables

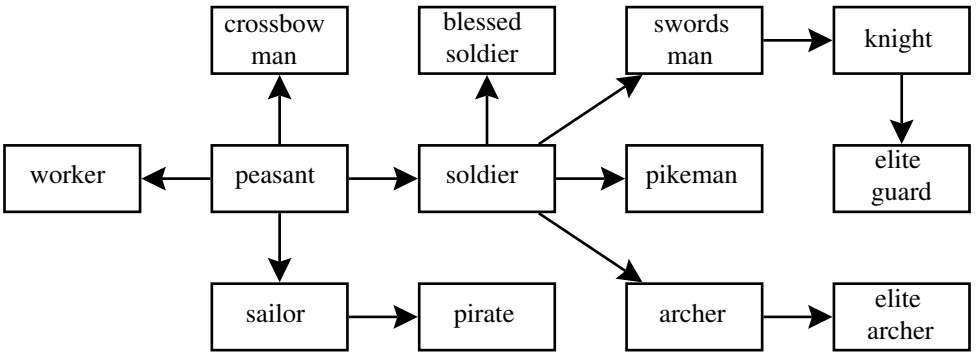
A.1 Combat ratings for various monsters

name	combat rating
elf	(50,50,100)
spirit	(50,50,0)
undead	(1,1,0)
savage	(1,1,0)
skeleton	(5,5,0)
barbarian	(2,1,0)
nazgul	(80,80,0)
ratspider	(5,5,0)
centaur	(30,30,0)
minotaur	(30,5,0)
giant spider	(150,100,0)
rat	(3,3,0)
lion	(100,100,0)
giant bird	(200,150,0)
giant lizard	(30,30,0)
bandit	(3,3,0)
chimera	(130,130,0)
harpie	(80,120,0)
dragon	(500,500,0)
orc	(20,15,0)
gorgon	(10,20,0)
wolf	(5,5,0)
cyclops	(25,75,0)
giant	(75,25,0)
faery	(1,1,0)

A.2 Some common items

item	name	item	name	item	name
1	gold	53	warmount	80	leather
10	peasant	60	battering ram	81	ratspider
11	worker	61	catapult	84	blank scroll
12	soldier	62	siege tower	85	crossbow
13	archer	68	yew	86	rug
14	knight	69	elfstone	87	fish
15	elite guard	72	longbow	88	spice
16	pikeman	73	plate armor	89	pipeweed
19	sailor	74	longsword	90	ale
20	swordsman	75	pike	91	wool
21	crossbowman	76	ox	93	opium
22	elite archer	77	wood	94	woven basket
51	wild horse	78	stone	95	clay pot
52	riding horse	79	iron	97	fish oil

A.3 Training tables



num	kind	skill	input man	input item	where
11	worker	none	peasant [10]		
12	soldier	121	peasant [10]		
13	archer	9579	soldier [12]	longbow [72]	
14	knight	9580	swordsman [20]	warmount [53]	
15	elite guard	9580	knight [14]	plate armor [73]	castle
16	pikeman	121	soldier [12]	pike [75]	
17	blessed soldier	150	soldier [12]		temple
19	sailor	9502	peasant [10]		
20	swordsman	9580	soldier [12]	longsword [74]	
21	crossbowman	121	peasant [10]	crossbow [85]	
22	elite archer	9579	archer [13]		castle
24	pirate	9580	sailor [19]	longsword [74]	ship

A.4 Combat ratings

	attack	defense	missile
peasant	1	1	
worker	1	1	
sailor	1	1	
soldier	5	5	
pikeman	5	30	
swordsman	15	15	
pirate	5 (15)	5 (15)	
knight	45 (20)	45 (20)	
elite guard	90 (70)	90 (70)	
crossbowman	1	1	25
archer	5	5	50
elite archer	10	10	75

A noble is rated (attack=80, defense=80, missile=0).

Blessed soldiers fight with the same attack and defense values as regular soldiers, but has a 50% chance of surviving a hit.

Pirates fight (15,15,0) on a ship, but only (5,5,0) on land.

Knights and elite guard receive a -25 penalty to both their attack and defense ratings when fighting on a ship or in a swamp province.

structure	men protected
Castle	first 500
Tower	first 100
Galley or roundship	first 50
Other structures	first 50

engine	attack	defense	missile
catapult	25	200	25
battering ram	30	250	
siege tower	30	250	

A.5 Civilization

feature	contribution
safe haven	2
castle	1.5
city	1
tower	1
temple	1
inn	0.5
mine	0.5

A.6 Maintenance costs

num	kind	cost
10	peasant	1
11	worker	2
19	sailor	2
21	crossbowman	2
12	soldier	2
13	archer	3
16	pikeman	3
17	blessed soldier	3
20	swordsman	3
24	pirate	3
14	knight	4
22	elite archer	4
15	elite guard	5

A.7 Carrying capacities

	weight	walking	riding	flying
man	100	100	-	-
riding horse	300	150	150	-
wild horse	300	self	self	-
warmount	300	150	150	-
knight	400	100	100	-
elite guard	400	100	100	-
ox	1000	1500	self	-
winged horse	300	150	150	150

Appendix B List of known skills and spells

Shipcraft [120]

All skills concerning ocean travel fall under this category. Shipcraft encompasses the building and repair of ships, training of sailors, and navigation at sea.

The following skills may be studied directly once Shipcraft is known:

num	skill	time to learn
9502	Sailing	one week
9503	Shipbuilding	one week
9554	Add ram to galley	two weeks
9582	Fishing	one week

Combat [121]

The student of combat learns the training of soldiers and how to lead them well in battle. With study and experience, the battle lord will find strategies to be used in difficult combat situations.

The following skills may be studied directly once Combat is known:

num	skill	time to learn
9501	Survive fatal wound	three weeks, 1 NP req'd
9505	Fight to the death	three weeks
9541	Construct catapult	one week
9570	Defense	one week
9579	Archery	two weeks
9580	Swordplay	one week
9581	Weaponsmithing	one week

Stealth [122]

Stealth includes the arts of theft, spying, assassination, kidnapping, and other covert activities. Most stealth skills will not make use of soldiers or other followers, instead relying on the sneakiness of a lone character.

The following skills may be studied directly once Stealth is known:

num	skill	time to learn
9509	Petty thievery	one week
9519	Determine inventory of character	one week
9520	Determine skills of character	one week
9521	Determine character's lord	one week
9591	Conceal faction	three weeks

Beastmastery [123]

Beastmastery spans the capture, training and breeding of all manner of animals, both wild and domestic. Common Beastmastery skills are used to capture and train horses for travel or combat. Higher level skills within this category concern the care and use of stranger beasts.

The following skills may be studied directly once Beastmastery is known:

num	skill	time to learn
9504	Bird spy	two weeks
9506	Capture beasts in battle	three weeks
9507	Use beasts in battle	three weeks
9517	Catch wild horses	one week
9529	Train wild horse to riding horse	one week
9530	Train wild horse to warmount	two weeks

Persuasion [124]

The skill of smooth-tongued oratists, Persuasion involves the manipulation of others through eloquent verbal dialog. From subverting a noble's loyalty to rousing peasant mobs with firey speeches, Persuasion can a powerful tool or weapon.

The following skills may be studied directly once Persuasion is known:

num	skill	time to learn
9515	Bribe noble	one week
9537	Raise peasant mob	one week
9538	Rally peasant mob	two weeks
9592	Train angry peasants	one week

Construction [125]

The domain of engineers, builders and architects, Construction concerns the erection of inns, castles, city walls, and other fortifications, as well as the training of workers.

The following skills may be studied directly once Construction is known:

num	skill	time to learn
9542	Construct siege tower	one week
9566	Stone quarrying	one week

Alchemy [126]

Called petty magician, druggist, and sometimes fool, the alchemist seeks to brew useful potions from elements and rare extracts. A method to transmute matter into gold is eagerly sought, but remains unknown.

The following skills may be studied directly once Alchemy is known:

num	skill	time to learn
9551	Brew healing potion	two weeks
9552	Brew potion of death	two weeks
9584	Collect rare elements	one week

Forestry [128]

Forestry is skill at managing forests and harvesting their products, such as timber, yew, and rare foliage.

The following skills may be studied directly once Forestry is known:

num	skill	time to learn
9540	Construct battering ram	one week
9568	Harvest lumber	one week
9569	Harvest yew	one week
9583	Collect rare foliage	one week
9588	Harvest opium	one week

Mining [129]

Mining is the skill of sinking deep shafts into mountainous provinces for the purpose of extracting valuable elements. Mining skills require a mine shaft to be constructed first with the `build mine` order. The likelihood of death because of cave-ins or poisonous gases seeping into the shafts tends to make mining dangerous work.

The following skills may be studied directly once Mining is known:

num	skill	time to learn
9563	Mine iron	one week
9564	Mine gold	one week

Religion [150]

The study of religion labels a character as a priest. Priests may learn to receive visions of far-away places, characters, or items. Some priests are said to possess miraculous powers, including the ability to bring dead nobles back to life. Religion may only be researched in temples.

The following skills may be studied directly once Religion is known:

num	skill	time to learn
9301	Receive vision	three weeks

Magic [160]

No apprentice magician can rightfully don the robes of his order without some knowledge of Magic [160]. Magic contains many spells useful on their own, as well as in conjunction with spells from the other schools of sorcery.

The following skills may be studied directly once Magic is known:

num	skill	time to learn
9101	Meditate	one week
9103	Perform common tasks for gold	one week
9104	Appear common	one week
9105	View current aura level of others	two weeks
9146	Heal	one week

Weather magic [161]

Weather spells concern the summoning and control of storms. Mild weather may have an effect on battle and ocean navigation. Fierce storms are capable of great damage.

The following skills may be studied directly once Weather magic is known:

num	skill	time to learn
9187	Summon rain	two weeks
9188	Summon wind	two weeks
9189	Summon fog	two weeks
9190	Direct storm	two weeks

Scrying [162]

Scrying concerns the gathering of information by means of magical spells. Simple spells may allow the mage to learn information about other characters nearby, or to view distant regions.

Advanced study delves into spells wrought long ago in by the men of Areth-Pirn as tools to rule their land, including the erection of magical barriers to travel, as well as additional spells of far-seeing.

The following skills may be studied directly once Scrying is known:

num	skill	time to learn
9141	Scry location	two weeks

Gatecraft [163]

Long ago the Ancients, a race of wise men who ruled the land with the aid of powerful magic, constructed a network of teleportation gateways to connect distant cities. As the reach of their magical portals grew, so grew the reach of their armies and knowledge. At their height the rule of the Ancients extended to every corner of the known world, and beyond.

With the passing of the Ancients, the system of gates fell into disuse and decay, and now retains only a hint of its former majesty. Gatecraft, once practiced by the most common among the Ancients, has since been forgotten by nearly all men.

The school of Gatecraft retains the only knowledge of the use of these mystical portals. Because knowledge of Gates can be a powerful tool or weapon, Gatecraft spells are jealously guarded even within the school. Some aspects of Gatecraft, such as the secret of their construction, have been completely lost.

The following skills may be studied directly once Gatecraft is known:

num	skill	time to learn
9115	Detect gates	one week
9116	Jump through gate	one week

Artifact construction [164]

The domain of high level wizards, artifact construction concerns the creation and manipulation of items which may focus and store aura, contain spells, or have other magical properties.

The following skills may be studied directly once Artifact construction is known:

num	skill	time to learn
9108	Reveal creator of artifact	one week
9109	Reveal where artifact was created	one week
9164	Destroy artifact	one week

Necromancy [165]

The way of the necromancer leads to fresh graves, the pits of Hades, and unspeakable horrors found only in dark corners of men's souls.

Power is gained through animation and enslavement of the dead, both followers as well as nobles. A necromancer looks upon a bloody battlefield strewn with the dead as a potential army waiting to be trained.

Advanced necromancers wield fierce powers, manipulating evil Runes to effect their will upon the world, and making pacts with demons to obtain power in return for souls.

With great power also comes great danger, and necromancy is the most dangerous of the magical arts. Only those willing to risk death at the hands of civilized men, or worse fates meted out by demons sent from the underworld, should even consider its study.

The following skills may be studied directly once Necromancy is known:

num	skill	time to learn
9179	Summon ghost warriors	two weeks
9180	Raise undead	two weeks
9185	Aura blast	two weeks

Appendix C Offenses against realism

- The training skills (such as Train peasant to worker) specify the time needed to train one man; time to train n men is linearly proportional. There are none of the economies of scale one would expect to find when teaching groups of men with a single instructor, and individual men can be trained in unreasonably short periods of time.
- Units in Olympia move by first waiting the duration of the journey in the starting location, then instantly jumping to the destination. One can see units long after they have departed. An interrupted movement order instantly brings units back to their starting location. Combat is possible with units after they have left.

  Note that most interaction with units, such as giving items and stacking, is explicitly prevented after they have begun movement. Combat is allowed up until the point a unit has actually moved out of a location, however, so that units can't become invulnerable by constantly moving.
- Reports are received for every location visited by a player's units up until the end of the month, even if the units have since left the viewed locations.

Appendix D Credits

Literally hundreds of playtesters offered suggestions on the design of Olympia, and spent countless hours testing the game and reporting bugs.

Special thanks to

Montgomery Box
Scott Cate
Steve Chapin
Dan Corrin
Carl Edman
Scott Hauck
David Kerven
Greg Lindahl
John Morrow
Harold Payson
John Sloan
Daniel Tormey
Scott Turner
Russell Wallace
and Chuk Craig, for suggesting the name *Olympia*

Olympia was inspired by Stephen Tihor's game T'Nyc.

Appendix E Listing of all commands

admit

`admit` *who-or-what* [‘`ALL`’] [*units*] • time: 0 days • priority: 0

Allow units to stack with the specified character, or to enter the given building or ship. By default, a unit may not stack with a character belonging to another faction, or enter a building controlled by another player.

The first argument to `admit` may either be a character, building or ship. If a character is given, `admit` will specify who may stack with the unit. If a building or ship is given, `admit` will determine who may enter the structure.

There are three cases for the rest of `admit`’s arguments: If no further arguments are given, `admit` will clear the permissions for the given entity, and deny admission or stacking to any foreign unit:

```
admit ce90                  # admit no one to ce90
```

If a list of units or players is given, those units will be permitted to enter or stack with the given entity:

```
admit ce90 778 2960 4240    # allow any member of faction 778,
                            # as well as 2960 and 4240, into ce90
```

Note that if a faction is specified, units within the faction must not be concealing their lord in order to be recognized.

Permission may be granted to every player’s units:

```
admit ce90 all                  # allow anyone into ce90
```

A list of units to exclude may follow ‘`ALL`’:

```
admit ce90 all 778 2960 4240    # allow anyone into ce90, except
                                # any member of faction 778, or
                                # units 2960 or 4240.
```

Summary examples:

```
admit ce90              # admit no one to ce90
admit ce90 all          # admit anyone to ce90
admit ce90 778          # admit any member of faction 778 to ce90
admit ce90 all 778      # admit anyone but a member of faction 778
admit ce90 2960 4240    # admit 2960 or 4240 to ce90
```

If there are too many units to list on a line with a single `admit` order, more `admit` orders may be used:

```
admit ce90 2596 3921 3934 3999 4012 4045 4046 4256 4300
```

has the same meaning as:

```
admit ce90 2596 3921 3934 3999 4012
admit ce90 4045 4046 4256 4300
```

Don't forget to give the entity again as the first argument on continuation `admit` orders! (ce90 in the above example)

Note, however, that the first `admit` order each turn begins a new admit list for the entity. In other words, units cannot be added to admit lists. Suppose, on the turn after the above admit orders were issued, the player ordered:

```
admit ce90 4345
```

Instead of adding 4345 to the admit list for ce90, the list would first be cleared, and 4345 would be the only permitted unit.

Only `admit` permissions toward the top most character of a stack are considered. Other units which do not have `admit` permission may accompany a character who does.

Admit lists are kept for the faction as a whole, not for particular units. The `admit` order should be executed by the player entity.

attack

`attack` *target* [*flag*] • time: 1 day • priority: 3

Engage two stacks in combat. Only the top most character in a stack may initiate combat.

target may be another unit, a ship, or a building, or a sub-location.

If flag is 1, the attacker will not attempt to move into the defender's position.

For example, one would use the order '`attack jg99`' to attack this castle:

```
Inner locations:
   Castle [jg99], castle, defense 20, owner:
      Law Netexus [2020], with three peasants
```

Do not use '`attack 2020`', as Law Netexus is not in the same location as characters outside the castle. If successful, the attackers will move into the castle.

A unit may not attack another member of its own faction. Two units in the same stack may not engage in combat.

Prisoners may not be attacked. To attack the stack holding a prisoner, issue an `attack` order for another member of the stack (such as the top unit in the stack).

For details on the resolution of battles, see Chapter 14 [Battles], page 63.

banner

`banner` [*unit*] "*message*" • time: 0 days • priority: 1

Provide a short message describing something about the unit. The message will appear after the unit in the turn report.

If a unit number is specified, the given unit's banner will be changed. This is useful for setting banners on entities which don't execute orders directly, such as ships and castles. Banners may not be set for units controlled by another player.

Example:

```
> banner "carrying a gold standard"
> banner 2019 "carrying a gold standard"

Seen here:
   Fighters of Pelenth [2019], "carrying a gold standard"
```

The message should be limited to 50 characters or it will be rejected.

behind

`behind` *number* • time: 0 days • priority: 1

Indicate how close to the front lines the unit should stand in combat. Number should be between 0 and 9. 0 is the front, 9 the rear.

Only missile fighters such as archers may attack from the rear. Swordsmen, knights, etc. may not. When all of the units in a row have been killed or taken prisoner, the next row will advance (skipping any empty rows).

board

`board` *ship* [*maximum fee*] • time: 0 days • priority: 2

Pay a fee to board a ship operating as a ferry. If a maximum is specified, no more than that amount of gold will be paid as a boarding fee.

The `board` order requires that the ship be operating as a ferry. In other words, the ship's captain must have set a boarding fee with the `fee` order. Boarding fees are charged per 100 weight of the passengers. For example, if the captain's fee is 200 per 100 weight, and a stack with a total weight of 1,500 wants to `board` the captain's ship, the stack leader must have 3,000 gold. The fee will be deducted from the boarding stack leader's inventory and given to the captain of the ship.

The `board` order will fail if the ship is already overloaded, or if the ship would be overloaded if the stack boarded the ship.

Passengers waiting to board a ferry should use `wait ferry`. Ferry captains should sound their horn before departure with the `ferry` order.

bribe

`bribe` *who amount* [*flag*] • time: 7 days • priority: 3

Offer a character gold in an attempt to win their service. The Persuasion subskill Bribe noble [9515] is required to use the `bribe` order.

The amount must be at least 250 gold or the receiving noble will not consider it a serious offer.

Nobles with oathbound loyalty ignore bribes. They may accept the gift, but they will not renounce loyalty to their lord.

Nobles with contract loyalty will consider the bribe if it meets or exceeds their current loyalty rating. For example, a noble of loyalty contract-750 would ignore any bribe of less than 750 gold.

Nobles with fear loyalty, or nobles not sworn to any lord, will seriously consider any bribe of at least 250 gold.

If the receiving noble considers the bribe, there is a 33% chance they will renounce loyalty to their current lord, and swear fealty to the bribing character. There is also a chance that they will report the bribe or pocket it for themselves. In some cases they may even renounce loyalty to their lord, but not swear fealty to the briber, instead choosing to go off on their own.

Characters who are successfully bribed will have loyalty contract-250 to the bribing character. If flag is 1, and the bribe attempt is successful, the bribed character will stack with the briber.

Bribes will be refused by characters who have already switched lords during the month because of bribery. For example, suppose Osswid successfully bribes Feasel, and Feasel swears loyalty to Osswid. Later that turn, if Procrustes attempts to bribe Feasel, Feasel will refuse to accept the bribe.

Examples:

```
bribe 5499 750      # Offer [5499] 750 gold to switch factions.
                    # If successful, [5499] will not stack with the
                    # briber.

bribe 3409 300 1    # Offer [3409] 300 gold to switch factions.
                    # If successful, [3409] will stack with the briber.
```

build

`build` *structure* "*Name*" [*max-days*] • time: varies • priority: 3

Build the indicated structure or ship. All construction tasks require at least three workers to begin. The character should acquire the required quantity of building material, unstack from beneath other characters, and issue one of the following build commands:

`build inn` "*name of inn*"
`build mine` "*name of mine*"
`build temple` "*name of temple*"
`build tower` "*name of tower*"
`build castle` "*name of castle*"
`build galley` "*name of galley*"
`build roundship` "*name of roundship*"

For example, to start building a castle, one might issue:

```
build castle "Castle Blood"
```

A new structure or ship will be created, and the builder will be placed inside. Construction proceeds until completed. Interrupted construction may be resumed by entering the uncompleted structure and reissuing the appropriate build command (`build roundship`, `build tower`, etc.)

kind	effort	material	skill	where
inn	300	75 wood	125	province or city
mine	500	25 wood	125 or 129	mountain or rocky hill
temple	1,000	50 stone	125	anywhere
tower	2,000	100 stone	125	anywhere
castle	10,000	500 stone	125	province or city
galley	250	50 wood	9503	ocean port
roundship	500	100 wood	9503	ocean port

Effort is in worker-days.

Construction [125] is required to erect buildings. Shipbuilding [9503], a subskill of Shipcraft [120], is required to build ships. Mines may be built either with Construction [125] or Mining [129].

The noble will cease work after the specified maximum number of days have been spent at construction, or the task is finished, whichever comes first. If a maximum number of days to spend working is not given, construction continues until the task is finished.

The first tower a player builds requires no workers, no stone, and only takes a day to build.

For more details, see Section 15.3 [Buildings], page 72 and Chapter 16 [Ships], page 75.

buy

`buy` *item qty price* [*have-left*] • time: 0 days • priority: 3

Buy *quantity* of the given item for no more than *price* each.

If *quantity* is zero, any pending `buy` order for the indicated item will be canceled.

Only one pending `buy` order may be active for each item at a time. If two `buy` orders are issued for the same item, and the first has not yet completed, the second `buy` order will replace the first.

Buys and sells are only matched in cities, where markets may be found to execute trades. However, `buy` and `sell` may be issued anywhere.

If the a match is found with a seller, the trade will be executed immediately. Otherwise, the trade will be saved until a match with a seller may be found.

Example:

```
> buy 79 3 5
Try to buy three iron [79] for 5 gold each.
```

If there is a seller of iron in the city, selling for 5 gold or less, the trade will be executed:

```
1: Bought three iron [79] for 15 gold.
```

Otherwise, the buy will be placed on the character's pending trades list:

```
Pending trades:

trade    price    qty   item
-----    -----    ---   ----
buy        5      3   iron [79]
```

At the end of the turn, the buy order will appear in the city's market report:

```
Market report:

trade    who    price    qty   item
-----    ---    -----    ---   ----
  buy   2019        5      3   iron [79]
```

A trade will match between a buyer and a seller if the buyer's price is greater than or equal to the seller's price. The actual price that the trade will be made at is the seller's price.

Have-left is an amount of gold that the character will not spend on this trade. For instance, if the unit issued `buy` 79 2 10 5, this would mean '`Buy two iron [79] for 10 gold each, but not if I'll have less than 5 gold left over.`' If the character had 15 gold, only one of the iron would be bought.

catch

`catch` [*number of horses*] [*days*] • time: as given • priority: 3

Attempt to catch any wild horses found roaming the plains or in a pasture. Each day the noble will have a 25% chance of catching a wild horse, if any can be found. Up to three horses may be caught per month in a location.

A unit must know the Catch wild horses [9517] subskill of Beastmastery [123] in order to catch horses.

Examples:

```
catch        # catch as many horses as possible
catch 1      # stop after catching 1 wild horse
catch 0 10   # try to catch horses for 10 days
catch 1 10   # stop after catching 1 wild horse, or 10 days,
             # whichever comes first
```

In all cases, horse-catching ceases when no more horses can be found roaming the plains. `catch` is an alias for '`collect 51`'.

claim

`claim` *item* [*number*] • time: 0 days • priority: 1

New players are given a few things to get them started. Rather than load them with possessions when they first set foot in the world, slowing their travel and make them tempting targets for bandits, they may obtain these items at any time with the `claim` command.

New players are given:

```
3,000 gold [1]
   25 wood [77]
```

Any unit in the player's faction may claim the items.

```
> claim 77
Claimed 25 wood.
> claim 1 1000
Claimed 1,000 gold.
```

collect

`collect` *item* [*number*] [*days*] • time: as given • priority: 3

Catch, harvest, collect or produce the given item from the unit's current location, if possible. Collection proceeds until the desired quantity of the item has been obtained, or the given number of days has passed, whichever comes first.

If neither number or days is given, `collect` will continue until the location's supply of the item is fully depleted.

Collection ceases when no more of the resource is available in the location. Thus, '`collect 87 10`' may obtain less than 10 fish [87], if the current ocean province's supply of fish runs out.

Examples:

```
collect 10          # Recruit as many peasants [10] as possible
collect 68 2        # Harvest two yew trees
collect 87 0 10     # Fish for 10 days
collect 87 25 10    # Fish until 25 fish [87] have been caught, or
                    # for 10 days, whichever comes first
```

Many collection commands have aliases so they can be more easily remembered. For example, the above orders could be written:

```
recruit
yew 2
fish 0 10
fish 25 10
```

Most collection tasks require the unit to possess a particular skill, and to be in a likely location to obtain the desired item.

num	item	who	where	qty	skill	chance	alias
10	peasant	noble	city, province[1]	10	none	100%	recruit
68	yew	worker	yew groves	5	9569	100%	yew
77	wood	worker	forest	30	9568	100%	wood
78	stone	worker	mountains	50	9566	100%	quarry
87	fish	sailor	ocean	50	9582	50%	fish
51	wild horse	noble	plains	3	9517	25%	catch

Other rare items which may be collected are listed in the lore sheets for Collect rare foliage [9583], a subskill of Forestry [128], and Collect rare elements [9584], a subskill of Alchemy [126].

Explanation of table columns:

num

item The resource to be collected. For example, Fish [87].

who Some tasks may only be performed by a noble, while others must be done by men, such as workers or sailors. For instance, recruiting requires the attention of a noble. Additional men are of no benefit.

Fishing, however, requires sailors. The noble himself does not participate. The more sailors available for fishing, the quicker the fish will be caught.

where

qty Resources are replenished at the end of each month. Every province (forest, mountain, swamp, desert or plains) will have ten recruitable peasants at the beginning of each turn. Each ocean province will have 50 fish that may be caught. Mountains yield up to 50 stone per turn.

skill Skills are required for most collection tasks (except recruiting):

activity	subskill	category
recruiting	none	none
cutting yew	Harvest yew [9569]	Forestry [128]
cutting timber	Harvest lumber [9568]	Forestry [128]
quarrying	Stone quarrying [9566]	Construction [125]
fishing	Fishing [9582]	Shipcraft [120]
horse catching	Catch wild horses [9517]	Beastmastery [123]

chance The rate of collection is certain for most tasks. Cutting timber yields one wood [77] per day per worker, as long as the forest has a supply of wood.

However, some tasks have an element of randomness. Horse-catching, for example, is a difficult, uncertain task. Each day there is only a 25% chance that a noble will catch a wild horse.

The chance is for the unit as a whole, not for individual men engaged in the task. Thus, a unit of sailors will collect fish about half of the days they spend fishing. The other half, no fish will be caught.

alias Common collection tasks have an alias so they are easier to remember.

task	alias
collect 10	recruit
collect 68	yew
collect 77	wood
collect 78	quarry
collect 87	fish
collect 51	catch

For example, the orders '`collect 51 0 10`' and '`catch 0 10`' are equivalent.

Note that if no arguments are supplied to the `collect` order (or one of its aliases), it may never terminate. For instance, suppose a unit with one sailor fishes alone in an ocean province:

```
fish    -or-    collect 87
```

Since no arguments were given for the `fish` order, fishing will continue as long as the ocean has a supply of fish. The sailor will produce, on average, a fish every two days, or 15 fish per month. Since this is less than the monthly supply of fish in an ocean province, the sailor will never exhaust the supply, and thus is likely to continue fishing forever, until interrupted with a `stop` order.

contact

`contact` *who* • time: 0 days • priority: 0

Used by characters hidden through stealth, magic, or because of fog to inform other units of their presence.

For example, if Osswid [5501] is employing stealth skills to hide, he will not be visible to other characters. Orders issued by other units referencing [5501] will fail.

However, Osswid may desire to allow certain units to interact with him while he is hidden. If this is the case, he should issue `contact` orders for those units to be informed of his presence.

The `contact` order may also be specify a player number. In this case, every unit in the specified player's faction will be aware of the hidden unit's presence.

Contact information is maintained until the hidden unit moves. Once the unit has moved, it must issue new `contact` orders if other units are to continue to be able to interact with it.

The `contact` order is silent; it does not provide a message to the contacted player or faction.

Units belonging to the same player are always aware of each other's presence. They do not need to use `contact`.

Hidden units are not shown in the location report, even if their presence has been reveal through `contact` or `seek`.

Related orders: [seek], page 138

decree

`decree` *decree* • time: 0 days • priority: 0

Issue a decree to all garrisons that the noble controls, either directly or through other characters who have pledged to the noble. There is no way to issue selective decrees to individual garrisons; all garrisons under control receive the same decree.

The decree may be one of the following:

watch who
: Instruct the garrisons to watch for the given unit. Each garrison may accept up to three `decree watch` order per month. Additional watch decrees are ignored.

hostile who
: Instruct the garrisons to attack the specified unit on sight. Each garrison will accept up to three `decree hostile` orders per month. Additional hostile decrees are ignored.

default

`default` *who* • time: 0 days • priority: 0

Clear any `hostile`, `neutral` or `defend` attitude for the unit or faction. May be issued by either a character or the player entity.

See Section 15.2 [Combat attitudes], page 70 for more information about declared attitudes.

Related orders: [defend], page 112 [neutral], page 127 [hostile], page 122

defend

`defend` *who* • time: 0 days • priority: 0

Declare a unit or player to have attitude `defend` towards another unit or player *who*. When a unit is attacked, other units which have `defend` attitude to the unit will join the combat and aid with defense.

When a combat occurs, the attitude of the top-most character in each stack in the location is checked. If any unit is declared `defend` to the victim of the attack, the unit's stack will join the combat on the victim's side.

See Section 15.2 [Combat attitudes], page 70 for more information about declared attitudes.

Related orders: [default], page 112 [neutral], page 127 [hostile], page 122

drop

`drop` *item qty* [*have-left*] • time: 0 days • priority: 1

Discard the specified quantity of item. If have-left is specified, at least this many will not be discarded. If quantity is zero, the entire inventory of the given item will be discarded.

For example, if Feasel has 100 gold [1]:

```
drop 1 50          #  Throw away 50 gold
drop 1 0           #  Throw away 100 gold  (0 means all)
drop 1 100         #  Throw away 100 gold
drop 1 100 25      #  Throw away 75 gold   (keep 25)
drop 1 0 25        #  Throw away 75 gold   (keep 25)
```

Common items including gold, lumber, swords, etc. are permanently lost when discarded. It is assumed that they are quickly scavenged by others.

Unique items, such as magical artifacts, scrolls, potions and rare books, do not become permanently lost. Such items will be found through exploration of the province in which they were lost. Unique items lost at sea will wash ashore on a nearby island or coast, where they may also be found through exploration.

The `drop` order is also be used to release men such as peasants, workers, soldiers, etc. from service:

```
drop 10 3              # Release 3 peasants from service
```

When used on a ship, `drop` will throw items overboard.

email

`email` *new email address*

Change the address to which turn results should be sent to the given address. Must appear after the `begin` order.

execute

`execute` [*prisoner*] • time: 0 days • priority: 1

Put the given prisoner to death. If no prisoner is specified, all of the noble's prisoners will be executed.

exhume

`exhume` [*body*] • time: 7 days • priority: 3

Dig up the body of a dead noble in a graveyard. If no argument is given, a random body will be recovered. Otherwise, only the specified dead noble will be exhumed.

When nobles die, they are buried in a nearby graveyard. A dead body exists for one game year, at which point it fully decomposes, whether it has been exhumed or remains in the grave.

Exhumed bodies may sometimes be brought back to life by priests, and are used by Necromancers for evil purposes.

explore

`explore` • time: 7 days • priority: 3

Explore the current location for hidden features or routes. If there is something to find, there is a random chance with each attempt that the explorer will find it.

```
1: > explore
1: Explore location for hidden features.
  ...
7: Exploration uncovers no new features.
```

Exploration in a province may also turn up rare items which have been lost.

If the exploring character is on a ship on the ocean, the surrounding ocean province will be searched for hidden islands or sea routes.

fee

`fee` [*gold per 100 wt*] • time: 0 days • priority: 1

Set a fee required to `board` a ship. The fee is an amount of gold which will be charged per 100 weight to units attempting to `board` a ship captained by the unit which issued `fee`.

If `fee` is zero or is not given, the boarding fee will be cleared. If no fee is set, units may not enter the ship with the `board` order.

Examples:

```
> fee 200    # charge passengers 2 gold for every point of weight
> fee 0      # we're not a ferry anymore; don't let strangers on
             # with the board order
```

ferry

`ferry` • time: 0 days • priority: 1

Sound a ship's horn, signaling units in port waiting with `wait ferry` that they may begin boarding the ferry. `ferry` may only be used by the captain of a ship.

Units waiting to board a ferry should use `wait ferry` instead of `wait ship` to allow existing passengers on the ferry time to unload. If those trying to `board` do so before existing passengers have unloaded, the `board` order may fail because of ship weight limits.

fish

`fish` [*number of fish*] [*days*] • time: as given • priority: 3

A unit must be on a ship on the ocean in order to fish. Each day, the unit has a 50% chance of encountering fish. If fish are found, each sailor will catch one fish that day, as long as fish stocks in the ocean province last. Ocean provinces generate 50 catchable fish per month.

A unit must know the Fishing [9582] subskill of Shipcraft [120] in order to fish, and must have at least one sailor.

Only sailors may be used for fishing. Pirates will not fish.

Examples:

```
fish         # obtain as many fish as possible
fish 25      # stop after getting 25 fish
fish 0 5     # fish for five days
fish 25 5    # stop after five days, or 25 fish, whichever
             # comes first
```

In all cases, fishing will cease as soon as the stocks in the ocean province have been exhausted. `fish` is an alias for '`collect 87`'.

flag

`flag` string • time: 0 days • priority: 1

Signal characters waiting with a `wait flag` order. The string is an arbitrary sequence of characters or numbers, but it should match the flag specified in the `wait flag` order.

For example, a character may wait for a particular signal from Osswid [5499]:

```
> wait flag yellow 5499
```

This will wait for Osswid [5499] to raise a flag saying '`yellow`'. Osswid would issue:

```
> flag yellow
```

Which would cause the wait order to complete.

The `wait flag` order defaults to watching for flags from units in the player's faction. It may also watch for flags from units in another faction, a flag from a specific unit, or a flag from any unit in the game:

```
wait flag yellow        # wait for 'flag yellow' from a unit
                        # in our faction

wait flag yellow 5499   # wait for 5499 to order 'flag yellow'

wait flag yellow 918    # wait for any unit belonging to player
                        # 918 to order 'flag yellow'.

wait flag yellow 0      # wait for any unit in the game to give
                        # the yellow flag
```

Characters do not need to be in the same location to flag.

Flags are remembered for the duration of the month. If a `wait flag` is issued for a flag which has already been raised, the wait will immediately finish (it will take zero time).

However, flags which have been raised are not remembered between turns, so a flag raised on the 30th of month 5 will not satisfy a `wait flag` on the 1st of month 6.

`fly`

`fly` *direction or destination* [...] • time: varies • priority: 2

Fly in the indicated direction, or to the specified location. Refer to the `move` order for details about movement.

Characters must have some means of flight to be able to use `fly`. For instance, a sufficient number of winged horses must be available to carry the entire stack and its belongings. If the weight of the stack exceeds the flight weight capacity, the `fly` order will fail.

Flying traverses any link in a maximum of three days.

rated time of route	time to fly across
0 days	0 days
1 day	1 day
3 days	3 days
4 days	3 days
8 days	3 days

Flying animals, such as winged horses, become tired when flying over the ocean, as they have no place to land and rest. Up to fifteen days may be spent flying over the ocean. At the end of the fifteenth day, the exhausted animals will be unable to continue, and the entire stack will plunge into the sea, killing all of the stack's characters.

Any arrival at a land or ship location will allow the flying creatures to rest and regain their strength.

Multiple destinations or destinations may be given as arguments to `fly`. If the first fails, the next will be tried, until a passable route is found, or `fly` runs out of arguments.

For example, '`fly n e w`' would first try to fly north. If travel north was not possible for any reason, east would next be tried, then west. In this way exploring units can attempt to deal with the case of running into a hole in the map or other impassable province.

forget

`forget` *skill* • time: 0 days • priority: 1

Forget all knowledge of a skill. If a category skill is specified, all subskills within the category will also be forgotten.

form

`form` *unit* "*name of new character*" • time: 7 days • priority: 3

Spend a Noble Point to obtain a new character for the player's faction. Nobles must be formed in a city.

The *unit* parameter should be selected from the line near the top of the turn report listing new character numbers:

```
The next five nobles formed will be:  5717 3215 4902 4489 5628
```

If 0 is given for *unit*, the first number in the list will be chosen. However, if more than one `form` is to be issued during the month, it can be uncertain which will execute first. By specifying the new unit number, the player will know the correct entity number when submitting orders for the new character.

Any character in the player's faction may issue `form`. The newly obtained noble will appear and stack beneath the forming character. The new noble is given the name specified as the second argument to `form`.

Orders for new units may be submitted in the turn they are formed. As soon as the new unit comes into existence, it will begin to execute its commands.

New units will occasionally bring with them possessions or knowledge of a skill, but most often they are not equipped and untrained. New nobles join with loyalty contract-500.

Example:

```
1: > form 3215 "Feasel the Wicked"
1: Attempt to form a new noble.
7: Feasel the Wicked [3215] stacks beneath us.
```

More than five units may not be formed in a turn.

format

`format` *num*

Some players find the location activity logs less useful than others, and prefer them to be banished to the end of the turn report. '`format 1`' will do this. '`format 0`' is the default. `format` should appear after the `begin` order.

0 Character reports listed first, followed by location reports including activity logs.

1 Location reports without activity logs, followed by character reports, followed by location activity logs.

garrison

`garrison` *castle* • time: 1 day • priority: 3

Claim a province in the name of a castle by leaving a garrison of soldiers to defend it.

The castle must be completed in order to bind a garrison to it.

There may not be any existing garrison in the location. Only provinces may be garrisoned (the character should be at the outermost level, not inside a sublocation or building).

The character must have ten soldiers [12] to create a garrison. The soldiers will be removed from the noble's inventory and placed in the new garrison.

See Chapter 17 [Provinces], page 80 for details about how garrisons work.

get

`get` *who item* [*qty*] [*have-left*] • time: 0 days • priority: 1

Get the given quantity of item from the specified target.

Characters may `get` items from other units in their faction, or from garrisons which they control. `get` may not be used to take items from another faction's units; the other faction's units must issue `give` orders.

Specifying zero for the quantity will cause the unit to attempt to take all of the target's inventory of the item. If have-left is given, at least that many will not be taken.

For example, if Feasel [5499] had 100 gold[1]:

```
get 5499 1 50          #  Get 50 gold from Feasel
get 5499 1 0           #  Get all 100 gold from Feasel
get 5499 1 100         #  Get all 100 gold from Feasel
get 5499 1 100 25      #  Get 75 gold from Feasel
get 5499 1 0 25        #  Get 75 gold from Feasel
```

Prisoners may also be transferred with the `get` order.

```
get 5499 4569          #  Transfer prisoner [4569] from [5499]
```

When taking men from a garrison, the garrison must be left with at least ten men, or the `get` will fail. For example, if a garrison had five peasants and six soldiers, either a peasant or a soldier could be taken. No men can be taken from a garrison with less than ten men.

give

`give` *to-who item* [*qty*] [*have-left*] • time: 0 days • priority: 1

Give quantity of an item in inventory to another character. Specifying zero for quantity will cause the unit to give all of its inventory of the item. An amount specified by have-left will be retained.

For example, if Feasel has 100 gold [1]:

```
give 5499 1 50          #  Give 50 gold to [5499]
give 5499 1 0           #  Give all 100 gold to [5499]
give 5499 1 100         #  Give all 100 gold to [5499]
give 5499 1 100 25      #  Give 75 gold to [5499]
give 5499 1 0 25        #  Give 75 gold to [5499]
```

Prisoners may also be transferred with the `give` order.

```
give 5499 4569          #  Give prisoner [4569] to [5499]
```

guard

`guard` *flag* • time: 0 days • priority: 1

Protect a city or province from looting and pillaging. '`guard 1`' will cause the unit to suppress `pillage` attempts by other characters. The guarding unit must be the top unit in its stack.

Any number of units in a location may be on guard. The message '`on guard`' will appear next to guard units in the location report.

A character attempting to pillage the city or province may specify a flag which causes them to attack the guards. In this case, all of the stacks on guard will respond to the attack. However, allies of the guard units will not respond. Only guard stacks will be involved in combat to suppress looting and pillaging.

Specifying a flag of 0 turns off guarding. Any movement by the guarding character will also clear the guard flag. For example, if a unit issued '`guard 1`' in a city, then moved out of the city, the `guard` flag would be cleared, and the unit would no longer be on guard.

The guarding units must be inside they location they are to guard. Units in a castle in a city will not protect the city. Units guarding a city will not protect the surrounding province.

honor

`honor` *amount* • time: 1 day • priority: 3

Pay a unit for its service. Units paid will take gold out of their own inventory and use it for their own purposes.

Nobles with oathbound loyalty will refuse to be paid, and reject the gold. Nobles with fear loyalty will be converted to contract loyalty.

The amount of money paid accumulates in the noble's contract loyalty rating. For instance, a fear-10 noble honored with 500 gold would have loyalty contract-500. A contract-250 noble honored with 500 gold would become contract-750.

hostile

`hostile` *who* • time: 0 days • priority: 0

Declare unit or player *who* hostile to the player or faction issuing the order. Units declared hostile will be attacked on sight.

If the player entity issues `hostile`, any unit in the player's faction will attack the hostile unit on sight. If an individual character issues `hostile`, the character will attack the hostile unit on sight.

Units will only initiate automatic attacks if they are in perfect health. Hostile units which are being held prisoner will not be attacked.

An entire faction may be declared hostile by specifying another player entity instead of an individual unit. Any unit not concealing its affiliation with the faction declared hostile would be attacked on sight.

Examples:

```
hostile 4240            # attack 4240 on sight
hostile 778             # attack any member of faction 778 on sight
```

See Section 15.2 [Combat attitudes], page 70 for more information about declared attitudes.

Related orders: [default], page 112 [defend], page 112 [neutral], page 127

lore

`lore` *sheet*

Resend a lore sheet which has appeared before in the player's turn report. Must appear after the `begin` order.

The `lore` command is processed by the order scanner, and does not appear in the turn report. Requested lore sheets appear at the end of the acknowledgement sent by the order scanner.

make

`make` *item* [*qty*] • time: as given • priority: 3

Make the indicated item. *qty* defaults to one. Each of the following items requires one day each to produce:

num	item	skill	material
72	longbow	9581	yew [68]
73	plate armor	9581	iron [79]
74	longsword	9581	iron [79]
75	pike	9581	wood [77]
84	blank scroll	126	lana bark [64]
85	crossbow	9581	wood [77]
94	woven basket	none	none
95	clay pot	none	none
99	hides	none	ox [76]

Knowledge of Weaponsmithing [9581], a subskill of Combat [121], is required to make weapons and armor. Alchemy [126] must be known to produce blank scrolls from lana bark. No skills or input materials are needed to make woven baskets or clay pots.

Examples:

To make 5 longbows from 5 yew (a kind of wood):

```
make 72 5               # takes 5 days
```

To make 1 plate armor from 1 iron:

```
make 73 1               # takes 1 day
```

To make 5 woven baskets:

```
make 94 5               # takes 5 days
```

Although making pots and baskets is considered menial labor, some desperate nobles have resorted to selling them in cities to earn gold. Each city in Olympia either buys woven baskets or clay pots.

message

`message` *# of lines of text to-who* • time: 1 day • priority: 3

Send a message to the target named in the second parameter. The target may be any valid entity, such as a character, a player, or a location (including provinces, cities, ships, buildings, or other sublocations).

The number of lines in the message should be given as the first argument to `message`. The text of the message should follow the `message` order.

For example:

```
message 2 2909
  Aye, ye not be safe in this world for long, Oleg.
  I be remembering ya, and yer past fould deeds.  Arrr!
move ev18
recruit
```

(The text of the message is indented here to make the orders easier to read. The order scanner will ignore leading spaces on each line.)

This would send the message to character [2909]. The recipient would see the message in his turn report:

```
Received a message from Captain McCook [2104]:
   "Aye, ye not be safe for long in this world, Oleg.
I be remembering ya, and yer past foul deeds.  Arrr!"
```

Take care to carefully count the lines of text in the message. If the wrong number of lines is given, your character may end up sending his orders to the recipient instead of executing them.

Each line of message text should be less than 60 characters long, or the message will be rejected.

move

`move` *direction or destination* [...] • time: varies • priority: 2

Travel in the indicated direction, or to the specified location. If a location is specified, there must be a direct route to it from the character's current location.

Compass points may be abbreviated: '`n`', '`s`', '`e`', '`w`'. Units may also `move out` of a sublocation to the surrounding province or location.

Visible routes leaving each province are listed in the turn report.

For example:

```
Routes leaving Plain [cw04]:
   North, to Ocean [cv04], South Sea, 1 day
   East, to Plain [cw05], 7 days
   South, to Plain [cx04], 7 days
   West, to Plain [cw03], 7 days
```

The following would be valid move commands from the example province:

```
move e  -or-  move cw05  -or-  move east
move s  -or-  move cx04  -or-  move south
move w  -or-  move cw03  -or-  move west
```

Note that `move` cannot be used to enter the ocean region [cv04]. The character would need to issue the `sail` order as captain of a ship to move north into the South Sea.

Route distances are rated in the number of days it would normally take a lightly loaded character, walking, to traverse them. Actual travel times may differ from the rated times.

Characters overloaded with belongings will make slower progress. Horses usually speed progress, but over especially rough or treacherous terrain, they may actually slow travel because they must be led and managed.

Inventory is considered for the stack as a whole. For instance, if one character possesses enough horses for every member of the stack, the stack may ride. Similarly, if one character is badly overloaded, but other characters have extra carrying capacity, they will assist their stackmate. Distribution of goods between units within a stack is irrelevant to movement.

The units stacked beneath a moving character will move with the stack leader. If the unit issuing `move` is stacked beneath another character, he will first automatically unstack before movement begins.

For example:

```
Seen here:
   Osswid the Constructor [5499], accompanied by:
      Feasel the Wicked [502]
```

If Osswid issues a `move` order, Feasel will go with Osswid. If Feasel issues a `move` order, however, Feasel would first unstack from Osswid before beginning travel.

Multiple destinations or destinations may be given as arguments to the `move` order. If the first fails, the next will be tried, until a passable route is found, or `move` runs out of arguments.

For example, '`move n e w`' would first try to move north. If travel north was not possible for any reason, east would next be tried, then west. In this way exploring units can attempt to deal with the case of running into an impassable province or a coast.

name

`name` [*unit*] "*new name for unit*" • time: 0 days • priority: 1

Change the name of a unit. If a unit number is specified, the given unit's name will be changed. This is useful for setting the name on entities which don't execute orders directly, such as ships and castles. Names may not be set for units controlled by another player.

Names of characters should be no longer than 35 characters. Names for other entities, such as towers, ships and items should be no longer than 25 characters. Longer names will be rejected.

Note that the name must be enclosed in quotes or it will not be recognized.

Examples:

```
> name "Feasel the Wicked"
> name cx53 "Castle Chardia"
```

neutral

`neutral` *who* • time: 0 days • priority: 0

Declare unit or player *who* to have attitude `neutral` to the player or faction issuing the order. Units declared neutral will not be aided if they are attacked.

Units in the same faction will defend each other if they are attacked, unless either unit is concealing its lord. This behavior may be suppressed with `neutral`. A unit may declare other units in its faction neutral. Or, the entire faction may declare itself neutral. In this case, members of the faction who were attacked would not aid each other, unless they were stacked together.

See Section 15.2 [Combat attitudes], page 70 for more information about declared attitudes.

Related orders: [default], page 112 [defend], page 112 [hostile], page 122

notab

`notab` *num*

Ordinarily, `tab` characters may appear in turn reports. Some players read their reports on computer equipment which cannot correctly display `tab` characters. If this is the case, issue the command:

```
notab 1
```

and tabs will be converted to spaces for your report.

oath

`oath` *level* • time: 1 day • priority: 3

Have a character take an oath of loyalty. Either 1 or 2 should be specified for *level*. One or two NPs will be deducted from the player's faction, and the specified character's loyalty will become oath-1 or oath-2.

Noble Points's spent on oathbound loyalty accumulate. Thus, oath-1 may be upgraded to oath-2 with the '`oath 1`' order.

password

`password` ["*password*"]

Set or clear a password for the player. If no argument is given, any existing password will be cleared. Otherwise, the specified password will be set.

Examples:

```
begin 663
password "xyzzy9"
end
```

This would set a password '`xyzzy9`' for the faction. Further orders will require the password to be specified after the player number on the `begin` line:

```
begin 663 "xyzzy9"
password                # clear the password
end
```

pay

`pay` *to-who* [*amount*] [*have-left*] • time: 0 days • priority: 1

Pay amount of gold to indicated entity, keeping at least have-left funds available. The `pay` command is shorthand for the equivalent `give` command to transfer gold from one unit to another.

The amount of gold to be paid will default to all of the unit's gold if not specified or if zero.

For example, if Feasel has 100 gold:

```
pay 5499 100            # pay 100 to [5499]
pay 5499 0              # pay 100 to [5499]
pay 5499 0 25           # pay 75 to [5499]
pay 5499 100 25         # pay 75 to [5499]
```

pillage

`pillage` [*flag*] • time: 7 days • priority: 3

Loot and pillage a province or city. If flag is not given, pillaging will not take place if the province is being protected by guards. If flag is 1, the looters will first attack the guards, then proceed with pillaging if the guards are defeated.

At least 10 men must be in the pillaging stack. They may be peasants, other nobles, soldiers, or other kinds of men. Only the top most character in a stack may initiate pillaging.

Pillaging will seize 1/3 of the available tax base, which otherwise would have gone to the location owner (if there is one). Another 1/3 of the potential taxes are destroyed.

Pillaging reduces the future tax base for a city or province. Locations take as many months to recover from pillaging as they have been pillaged. A city pillaged at least once per month for three months in a row will take three months to recover its full tax base.

Pillaging can be dangerous. There is a 33% chance that a mob of 12-36 angry peasants will form to resist further pillaging attempts. Angry peasants have a combat rating of (attack-2, defense-1, missile-0). If a mob forms, there is a 33% chance that the mob will immediately attack the pillager in retaliation.

players

`players`

Ask the order scanner to send a list of the players in the game.

pledge

`pledge` *who* • time: 0 days • priority: 1

Pledge all lands under control to another noble. The target noble will share control of all lands the pledging character owns, and the pledged lands will count towards the target's status.

Issue '`pledge 0`' to revoke control.

There is no way to pledge individual provinces. The target receives control and status for every province that the character owns, including provinces the character obtains after the `pledge` order is given.

See Chapter 17 [Provinces], page 80 for more details about pledging.

post

`post` # *of lines of following text* • time: 1 day • priority: 3

Post a public message in the location. All characters in the location will see the text of the message in the location description section of their turn reports. Signs remain visible for one turn.

The number of lines in the post should be given as an argument to `post`, followed by the text of the sign. For example, here is a post command followed by a move and a recruit:

```
post 3
  All ye beware!  These lands are the property of
  noble Osswid.  Those who trespass here without
  his permission will come to a GRAVE end!
move ev18
recruit
```

(The text of the post is indented here to make the orders easier to read. The order scanner will ignore leading spaces on each line.)

Take care to carefully count the lines of text in the post. If the wrong number of lines is given, your character may end up posting his orders on a sign instead of executing them.

Each line of posted text should be less than 60 characters long, or the post will be rejected.

promote

`promote` *who* • time: 0 days • priority: 1

Advance the given unit so that appears before the character issuing `promote` in the location list.

The character to be promoted should be stacked immediately beneath the promoting character, or appear later in the location list at the same level.

Examples:

Before:

```
Osswid the Constructor [5499], accompanied by:
   Feasel the Wicked [502]
```

Osswid issues '`promote 502`':

```
Feasel the Wicked [502]
Osswid the Constructor [5499]
```

Before:

```
Osswid the Constructor [5499], accompanied by:
   Feasel the Wicked [502]
Candide [1269]
```

Osswid issues '`promote 1269`':

```
Candide [1269]
Osswid the Constructor [5499], accompanied by:
   Feasel the Wicked [502]
```

quarry

`quarry` [*number of stones*] [*days*] • time: as given • priority: 3

Building stone is used in the construction of temples, castles and towers. Up to 50 stone may be removed from mountain provinces each month. Ten stone may be taken from desert provinces or rocky hills each month.

A unit must have the Stone quarrying [9566] subskill of Construction [125] in order to cut building stone. At least one worker is also required.

Each worker will cut and prepare one building stone [78] per day.

Examples:

```
quarry          # cut as much building stone as possible
quarry 25       # stop after cutting 25 stone
quarry 0 5      # quarry stone for five days
quarry 25 5     # stop after five days, or 25 stone, whichever
                # comes first
```

In all cases, quarrying will cease as soon as the supply of ready stone in the province has been exhausted.

quit

```
quit
```

Quit Olympia. This command must be issued by the player entity.

No turn report is mailed for the turn a `quit` order is processed.

raze

`raze` • time: varies • priority: 3

Tear down a building the character is inside. The command completes when the structure has been completely destroyed. The noble will safely leave buildings before they collapse.

Sturdier structures require more men to do a given amount of damage:

structure	man-days per point of damage
galley	1
roundship	1
temple	2
inn	2
tower	2
mine	2
castle	3

For example, two men can cause 1% of damage to a tower in a day.

Any men present with the noble will help with destruction. The noble is counted when the number of men are computed.

recruit

`recruit` [*days*] • time: as given • priority: 3

Recruit one peasant each day for the given number of days. If days is not specified, recruiting will obtain as many peasants as possible.

Peasants must be recruited in plains, mountain or forest provinces, or in cities. Recruiting can not be done in other sublocations (such as rocky hills, forest glades, etc.)

Up to 10 peasants may be recruited per location each month. One peasant is recruited each day the `recruit` command is used.

The `recruit` order will stop or fail when no further peasants are available to be recruited.

For example, one player could recruit for up to 10 days per month in a mountain province, obtaining one peasant on each of the 10 days.

If two players in a mountain province each issued `recruit` 10, both would recruit five peasants (each player would get a peasant a day until all 10 had been recruited).

Recruit is an alias for '`collect 10`'.

repair

`repair` [*days*] • time: as given • priority: 3

Repair the damaged ship or building that the character is inside. Requires at least one worker [11].

structure	worker-days per point of damage
galley	1
roundship	1
temple	2
inn	2
tower	2
mine	2
castle	3

Work proceeds until the structure has been fully repaired, or the given number of days has passed. Multiple units may work at repair simultaneously.

For example, a tower with 50% damage would require 100 worker-days to fully repair. A ship with 50% damage would require 50 worker-days. A castle, 150 worker-days.

Each use of `repair` on a ship requires one unit of pitch [261]. One unit of pitch is sufficient for a ship repair job, no matter how many days the repair takes. Pitch may be purchased in any port city.

research

`research` *skill* • time: 7 days • priority: 3

Attempt to learn hidden sub-skills of the specified category skill. Research into any skill except Religion [150] must be performed in a tower, by the tower's owner (the first character inside the tower). Research into Religion [150] must be performed in a temple.

Research requires a nominal fee of 25 gold to pay for materials. This fee is charge regardless of whether the research yields any results.

There is a 25% chance of success for each week of research.

If successful, the new subskill will be added to the character's partially known skill list. It then must be studied in order to become fully known and usable.

Example:

```
1: > research 120
1: Will research Shipcraft for seven days.
7: Research uncovers a new skill:  Improve ship rigging [9999].
7: To begin learning this skill, order 'study 9999'.

Partially known skills:
   9999  Improve ship rigging, 0/7
```

Research will fail immediately if there are no researchable skills to learn under the category. In this case the 25 gold fee is not charged.

See Chapter 12 [Skills], page 49 for more detailed information about `study` and `research`.

resend

`resend` [*turn*]

Ask the order scanner to resend a turn report. The turn argument defaults to the turn most recently run.

Because of their large size, turn reports other than the most recent may not be kept by the order scanner. If you request a report and the scanner responds that the report cannot be found, then the reports for that turn were deleted to reclaim disk space.

sail

`sail` *direction or destination* [...] • time: varies • priority: 4

Pilot a ship in the given direction, or to the specified location. If a location is specified, there must be a direct route to it from the current location of the ship.

Compass points may be abbreviated: '`n`', '`s`', '`e`', '`w`'.

The Sailing [9502] subskill of Shipcraft [120] is required to use the `sail` order.

The character issuing the sail order must be the ship's captain, i.e. the character must be the first unit listed in the '`Seen here`' report of the ship.

The captain must have a crew to assist in the handling of the ship. Galleys require fourteen pirates or sailors to row the ship. Roundships require at least eight pirates or sailors to man the ship's rigging.

ship	crew	capacity
galley	14	5,000
roundship	8	25,000

If the required number of crew are not available, travel is still possible, but will take an extra day for each hand the crew is short.

For example, a galley with only 10 men at the oars will take four extra days to travel across an ocean route. A roundship with no sailors at all would take eight extra days to traverse a route.

The `sail` order will fail if the ship is overloaded. Use `drop` to throw items overboard if necessary.

Multiple destinations or destinations may be given as arguments to `sail`. If the first fails, the next will be tried, until a passable route is found, or `sail` runs out of arguments.

For example, '`sail n e w`' would first try to sail north. If travel north was not possible for any reason, east would next be tried, then west. In this way exploring units can attempt to deal with the case of running into an impassable ocean route, or a coast.

Roundships sail from windy provinces in one day less than the rated time, provided the route would require two days or longer. For example, a roundship leaving a windy province across a route rated at three days would take only two to travel the link. But a roundship leaving port will always take one day, never zero days.

seek

`seek` *who* • time: 7 days • priority: 3

Seek for a hidden character. Characters may be hidden because of fog lying over a province, or through the application of stealth or magical skills.

Hidden characters can not be interacted with by a unit unless they `contact` the unit first, or the unit successfully finds them with `seek`. The `seek` order will return immediately if the character is visible, or has already been found with a previous `seek` order.

There is a 50% chance that a hidden unit will be found with `seek`, if it is actually present in the same location as the seeking character.

If a unit found with `seek` moves, the contact information is lost, and the unit again becomes inaccessible.

Hidden units are not shown in the location report, even if their presence has been reveal through `contact` or `seek`.

Related orders: [contact], page 111

sell

`sell` *item qty price* [*have-left*] • time: 0 days • priority: 1

Sell *quantity* of the given item for *max-price* each.

If *quantity* is zero, any pending `sell` order for the indicated item will be canceled.

Only one pending `sell` order may be active for each item at a time. If two `sell` orders are issued for the same item, and the first has not yet completed, the second `sell` order will replace the first.

Buys and sells are only matched in cities, where markets may be found to execute trades. However, `buy` and `sell` may be issued anywhere.

If the a match is found with a buyer, the trade will be executed immediately. Otherwise, the trade will be saved until a match with a buyer may be found.

Example:

```
> sell 79 3 5
Try to sell three iron [79] for 5 gold each.
```

If there is a buyer of iron in the city, buying for 5 gold or more, the trade will be executed:

```
1: Sold three iron [79] for 5 gold.
```

Otherwise, the sell will be placed on the character's pending trades list:

```
Pending trades:

trade    price    qty   item
-----    -----    ---   ----
 sell        5      3   iron [79]
```

At the end of the turn, the sell order will appear in the city's market report:

```
Market report:

trade    who    price    qty   item
-----    ---    -----    ---   ----
 sell   2019        5      3   iron [79]
```

A trade will match between a buyer and a seller if the buyer's price is greater than or equal to the seller's price. The actual price that the trade will be made at is the seller's price.

Have-left is an amount of the item that character will keep and not sell. For instance, if the unit issued '`sell 79 10 15 5`', this would mean "Sell ten iron for 15 gold each, but keep 5 iron for myself." If there character only have ten iron, this order would put five up them up for sale.

stack

`stack` *character* • time: 0 days • priority: 1

Stack beneath the specified character. If the character is already stacked beneath another unit, the character will first unstack before stacking with the new character.

Example: Two characters, Feasel and Osswid, are in the same location:

```
Seen here:
   Osswid [5499]
   Feasel the Wicked [502]
```

If Feasel issued the order '`stack 5499`', he would stack beneath Osswid:

```
Seen here:
   Osswid [5499], accompanied by:
      Feasel the Wicked [502]
```

stop

```
stop
```

Interrupt a command left running at the end of a turn. Must be the first order queued for a unit.

For example, suppose on day 28 a unit begins an explore command. Explore takes seven days, so as of the end of the turn, it will still have four days left before it completes. The order template for the unit will list the executing command:

```
unit 5499  # Osswid
   # > explore (executing for four more days)
```

The player may interrupt this command with the `stop` order, and begin unit 5499 executing a new set of orders (say, beginning with a `move`) on day 1 of the next turn:

```
unit 5499  # Osswid
   # > explore (executing for four more days)
   stop
   move
   ...
```

(Remember that '`#`' begins a comment in the orders, from the '`#`' to the end of the line).

The `stop` order will interrupt the executing `explore` before day 1 begins on the next turn. Some commands, such as `collect`, will give partial results; the unit will have whatever it has already collected. Other orders, such as `explore`, give no partial benefit, and the three days already invested in exploration will be wasted.

If a character executing a `move` order is interrupted with `stop`, the character will return to the original location, as if no travel had taken place.

`study`

`study` *skill* • time: 7 days • priority: 3

Study the given skill for one week.

If the skill has been studied before, learning will continue. Otherwise, the character must meet the requirements necessary to begin learning the skill.

To begin study of a skill requires the following:

- A source of instruction, one of the following:
  - The location teaches the skill
  - The character has a book or scroll which teaches the skill
  - The character discovered the skill through research
- Payment of 100 gold to begin study.
- Payment of Noble Points to learn advanced skills

Note that multiple `study` orders during a turn yield diminishing results:

- First week: 7/7
- Second week: 4/7
- Third week: 2/7
- Fourth week: 1/7

See Chapter 12 [Skills], page 49 for more detailed information about `study` and `research`.

swear

`swear` *who* • time: 0 days • priority: 1

Give a character to another player. A unit from the new player's faction must be present for the character to swear fealty to.

No change in the loyalty rating occurs as a result of this order.

A player may only issue `swear` once each turn.

take

`take` *who item* [*qty*] [*have-left*] • time: 0 days • priority: 1

Same as `get`. See [get], page 120 for a description.

terrorize

`terrorize` *who severity* • time: 7 days • priority: 3

Terror may be used to convert prisoners to a faction, or to reinforce fear loyalty among the faction's own units.

When used on one's own units, `terrorize` will establish or increase fear loyalty. Nobles of contract loyalty will be converted to fear. The severity amount will be added to the noble's fear rating. For instance, a fear-4 noble terrorized at severity 10 will become fear-14.

The severity indicates how much the noble will be abused to gain fear. The noble will suffer a *severity* hit to health as a result. Thus, there is a danger that the noble will fall ill with increasing amounts of terror. See Chapter 6 [Health], page 33 for details about illness.

Prisoners may also be terrorized. Prisoners of contract or fear loyalty lord have *severity* percent chance of swearing fealty to the terrorizing unit. Oathbound units can not be induced to convert through terror.

Thus, the more severe the terrorization, the greater the chance that a prisoner will swear to the terrorizer, but also the greater the victim's wounds. Prisoners have been known to die as a result of over-zealous conversion attempts.

Characters belonging to other player factions may not be terrorized unless they are being held prisoner.

Units of fear loyalty may not use the `terrorize` order.

train

`train` *kind* [*days*] • time: as given • priority: 3

Attempt to produce the given kind of men through training. One day is required to train each man.

See Section A.3 [Training], page 88 for more information.

num	kind	skill	input man	input item	where
11	worker	none	peasant [10]		
12	soldier	121	peasant [10]		
13	archer	9579	soldier [12]	longbow [72]	
14	knight	9580	swordsman [20]	warmount [53]	
15	elite guard	9580	knight [14]	plate armor [73]	castle
16	pikeman	121	soldier [12]	pike [75]	
17	blessed soldier	150	soldier [12]		temple
19	sailor	9502	peasant [10]		
20	swordsman	9580	soldier [12]	longsword [74]	
21	crossbowman	121	peasant [10]	crossbow [85]	
22	elite archer	9579	archer [13]		castle
24	pirate	9580	sailor [19]	longsword [74]	ship

Examples:

To train 5 peasants into 5 workers:

```
train 11 5              # takes 5 days
```

To train 5 peasants into 5 sailors (requires knowledge of Sailing [9502]):

```
train 19 5              # takes 5 days
```

To train 3 soldiers into 3 swordsmen (requires knowledge of Swordplay [9580], and a longsword [74] for each soldier to be trained):

```
train 20 3              # takes 3 days
```

ungarrison

`ungarrison` [*garrison*] • time: 1 day • priority: 3

Disband the garrison in the current province. The character removing the garrison must be a member of a faction which has rulership over the province.

If a specific garrison is given as an argument to `ungarrison`, only that garrison will be disbanded. This can be useful as a safety check, in case a failed `move` order might cause the wrong garrison unit to be disbanded.

The garrison's men will be transferred to the noble issuing `ungarrison`.

Examples:

```
ungarrison 5611         # remove Garrison [5611] from this province
ungarrison              # remove any garrison from this province
```

unload

`unload` • time: 0 days • priority: 3

Unload all passengers from a ship. May only be used by the captain of a ship. Every character not in the captain's stack will disembark from the ship.

Passengers may only be unloaded while the ship is docked in a port. They will not jump off a ship at sea.

unstack

`unstack` [*who*] • time: 0 days • priority: 1

When used without an argument, `unstack` removes the noble from being stacked beneath another noble:

Before:

```
Osswid the Constructor [5499], accompanied by:
   Feasel the Wicked [502]
   Captain McCook [2019], with five workers

502: > unstack
```

After:

```
Osswid the Constructor [5499]
Feasel the Wicked [502], accompanied by:
   Captain McCook [2019], with five workers
```

A stack leader may given a argument to `unstack`, specifying a unit to be dropped from the stack:

Before:

```
Osswid the Constructor [5499], accompanied by:
   Feasel the Wicked [502]
   Captain McCook [2019], with five workers

5499: > unstack 502
```

After:

```
Osswid the Constructor [5499]
Feasel the Wicked [502], accompanied by:
   Captain McCook [2019], with five workers
```

Note that in this example [2019] is stacked beneath [502], not [5499]. That is why Captain McCook remains with Feasel when Feasel unstacks. See Chapter 9 [Stacking], page 42 for more details on hierarchical stacks.

A special case of `unstack` is invoked if the noble specifies itself as the argument. In this case, the noble will drop out of the stack alone, leaving behind both those stack above as well as below:

Before:

```
Osswid the Constructor [5499], accompanied by:
   Feasel the Wicked [502], accompanied by:
      Captain McCook [2019], with five workers

502: > unstack 502
```

After:

```
Osswid the Constructor [5499], accompanied by:
   Captain McCook [2019], with five workers
Feasel the Wicked [502]
```

The effect on the stack in this case is the same as if Feasel had died or been captured in battle.

More examples:

Before:

```
Osswid the Constructor [5499], accompanied by:
   Feasel the Wicked [502]
```

After Feasel unstacks:

```
Osswid the Constructor [5499]
Feasel the Wicked [502]
```

If other characters were stacked beneath Feasel, they would remain with him.

Before:

```
Osswid the Constructor [5499], accompanied by:
   Feasel the Wicked [502], accompanied by:
      Candide [1269]
```

If Candide were stacked beneath Feasel, and Feasel unstacked, then:

```
Osswid the Constructor [5499]
Feasel the Wicked [502], accompanied by:
   Candide [1269]
```

On the other hand, if Candide were stacked beneath Osswid, then:

```
Osswid the Constructor [5499], accompanied by:
   Candide [1269]
Feasel the Wicked [502]
```

use

`use` *skill* [*arguments* ...] • time: varies • priority: 3

Make use of a skill or spell that the character knows. Sub-skills usually have a function which is invoked through `use`. These actions are described in the lore sheets revealed when sub-skills are learned.

Category skills do not have direct `use` functions. For example, one could use the Sailing [9502] skill to sail a ship:

```
use 9502 west
```

But no use function exists for the category skill, Shipcraft [120].

wait

`wait` *conditions* • time: varies • priority: 1

Wait for one of the the specified conditions to occur. The first condition which is satisfied will cause the wait command to terminate. If one of the specified wait conditions is true when the wait order is issued, no wait will take place, and the order will complete in zero time.

The following conditions may be waited on:

time n Wait for *n* days to elapse

day n Wait until the day of the month is *n*

unit n Wait until the character and the specified unit are in the same location

ship n Wait until the specified ship arrives in port.

loc n Wait until arrival at the given location.

gold n Wait until the character has at least *n* gold

item n q Wait until the character has at least quantity *q* of item *n*

stack n Wait until the character and unit *n* are members of the same stack. Position within the stack is not important; The condition is true whenever *n* and the character share a common stack leader.

top Wait until the unit becomes a stack leader. This could happen if the units's current stack leader unstacked or promoted the unit.

flag f [n] Wait until character *n*, or a character belonging to player *n*, signals with flag *f*. *n* defaults to the player issuing the flag order. If *n* is zero, wait until any character signals with flag *f*. See [flag], page 116 for more details

ferry n Wait for the ship *n* to sound its horn with the `ferry` order.

Units waiting to `board` a ferry should use `wait ferry` instead of `wait ship`. This will allow passengers on the ship time to disembark. When passenger unloaded is complete, the captain will sound the ship's horn, and those wishing to `board` may proceed.

Otherwise, those waiting to board may try to get on the ship before the current load of passengers has left, causing `board` orders to fail because of ship weight limits.

For example:

```
1: > wait time 1
1: Wait finished: one day has passed.
2: > wait time 10 day 5
5: Wait finished: today is day 5.
```

The first condition to become true will end the wait. This can be used to specify the maximum amount of time that the character will wait for something.

For instance, suppose we want to wait for [5499] to show up, but we don't want to wait past day 15.

```
> wait unit 5499 day 15
```

This order will finish on day 15 or as soon as 5499 shows up, which ever comes first. If 5499 is already here when the order is issued, no wait will take place. Likewise, if it is already past day 15, no wait will take place.

Keep in mind that the order priority mechanism eliminates the need to use waits in many cases. Consider synchronization in the following example:

Feasel [502] arrives at Osswid's [5499] location. Feasel gives Osswid some gold, and then leaves. Osswid departs after receiving the gold.

```
unit 502   # Feasel
  move ...
  pay 5499 100
  move ...

unit 5499  # Osswid
  wait unit 502
  move ...
```

The orders above will work; no additional waits are needed. After the `wait unit` becomes true, both 502's `pay` and 5499's `move` are ready to execute. Since `pay` has a lower priority than `move`, `pay` will happen first.

This technique also works with `stack` and `give`, since they also have a lower priority than `move`.

Table of Contents

Distribution 1

1 Introduction 2

1.1 Overview 2
1.2 Turn schedule 3
1.3 Olympia Times 3
1.4 Diplomatic email forwarding service 4
1.5 Compensation for errors 4
1.6 Cheating 4
1.7 Olympian Calendar 5
1.8 Game additions and rule changes 6
1.9 How do I join? 6
1.10 Email address, etc. 6

2 Important terms 8

3 Command summary 10

4 Submitting orders 12

4.1 Interrupting orders 14
4.2 Units not controlled by you (yet) 15
4.3 Use the order template 15
4.4 Be careful 16
4.5 Failed orders 17
4.6 Player Entity 17
4.7 Quitting 17
4.8 More examples 18
4.9 Order of characters in a location 19
4.10 Command priority 19
4.11 Command precedence within a location 21
4.12 At the same time 21

5 Olympian geography 23

5.1 Inner Locations 24
5.2 Inside a City 24
5.3 Who else is here? 25

5.4 More about geography . . . 25
5.5 Holes in the Map . . . 26
5.6 Hidden routes . . . 26
5.7 Ocean ports . . . 27
5.8 Port Cities . . . 28
5.9 An example city description . . . 28
5.10 Wilderness and civilization . . . 29
5.11 Dangerous places . . . 30
5.12 Wilderness . . . 30
5.13 Hades . . . 31
5.14 Faery . . . 31
5.15 The Cloudlands . . . 31
5.16 Guard monsters . . . 32

6 Health . . . 33

6.1 Death . . . 34

7 Markets . . . 35

7.1 Market report . . . 36
7.2 City purchasing . . . 37
7.3 Resolution of trades . . . 37
7.4 Partial trades . . . 38
7.5 More market examples . . . 38
7.6 Trade goods . . . 39

8 Loyalty bonds . . . 40

9 Stacking . . . 42

10 Carrying capacity . . . 43

11 Training men . . . 45

11.1 Maintenance cost . . . 46
11.2 Making weapons and armor . . . 47

12 Skills . . . 49

12.1 Sources of instruction . . . 50
12.2 Fee to begin study . . . 51
12.3 Advanced skills . . . 51
12.4 Diminishing returns from study . . . 51
12.5 Experience . . . 53

12.6 Skills not rated for experience . . . 53
12.7 Teaching . . . 54
12.8 Summary of studying . . . 54
12.9 A rough example . . . 54
12.10 Research . . . 56

13 Magic . . . 59

13.1 Aura . . . 59
13.2 Magician status . . . 60
13.3 Required Items . . . 60
13.4 Study of Spells . . . 60
13.5 Schools of Magic . . . 61
13.6 Religion . . . 62

14 Battles . . . 63

14.1 Resolution of battle . . . 63
14.2 Front and rear . . . 66
14.3 Missile Attacks . . . 66
14.4 Fortifications . . . 66
14.5 Item bonuses . . . 67
14.6 Prisoners . . . 68
14.7 Prisoner escapes . . . 68

15 Permissions and declared attitudes . . . 70

15.1 Allowing entrance and stacking . . . 70
15.2 Combat attitudes . . . 70
15.3 Buildings . . . 72
15.4 The first tower you build is free . . . 73
15.5 Inns . . . 73
15.6 Mining . . . 74

16 Ships . . . 75

16.1 How to sail . . . 75
16.2 Making ships . . . 76
16.3 Operating a ferry . . . 77
16.4 Ferry synchronization . . . 78
16.5 Control of buildings and ships . . . 79

17 Province control and ownership 80

17.1 Quick overview 80
17.2 Tax base 81
17.3 Castles 82
17.4 Garrisons 82
17.5 More about garrisons 83
17.6 Garrison reports 83
17.7 Referring to garrisons 84
17.8 Status 84
17.9 Pledging land 85
17.10 Status after pledging 85
17.11 Pillaging 85
17.12 Opium 86

Appendix A Summary of important tables 87

A.1 Combat ratings for various monsters 87
A.2 Some common items 87
A.3 Training tables 88
A.4 Combat ratings 89
A.5 Civilization 89
A.6 Maintenance costs 90
A.7 Carrying capacities 90

Appendix B List of known skills and spells 91

Shipcraft [120] 91
Combat [121] 91
Stealth [122] 92
Beastmastery [123] 92
Persuasion [124] 92
Construction [125] 93
Alchemy [126] 93
Forestry [128] 94
Mining [129] 94
Religion [150] 94
Magic [160] 95
Weather magic [161] 95
Scrying [162] 96
Gatecraft [163] 96
Artifact construction [164] 97
Necromancy [165] 97

Appendix C Offenses against realism 98

Appendix D Credits **99**

Appendix E Listing of all commands **100**

admit 100
attack 102
banner 102
behind 103
board 103
bribe 104
build 105
buy 106
catch 107
claim 108
collect 108
contact 111
decree 112
default 112
defend 112
drop 113
email 114
execute 114
exhume 114
explore 114
fee 115
ferry 115
fish 116
flag 116
fly 117
forget 118
form 118
format 119
garrison 120
get 120
give 121
guard 121
honor 122
hostile 122
lore 123
make 123
message 125
move 125
name 127

neutral . . . 127
notab . . . 128
oath . . . 128
password . . . 129
pay . . . 129
pillage . . . 130
players . . . 130
pledge . . . 130
post . . . 131
promote . . . 131
quarry . . . 132
quit . . . 133
raze . . . 133
recruit . . . 134
repair . . . 135
research . . . 135
resend . . . 136
sail . . . 136
seek . . . 138
sell . . . 138
stack . . . 139
stop . . . 140
study . . . 141
swear . . . 142
take . . . 142
terrorize . . . 142
train . . . 143
ungarrison . . . 144
unload . . . 144
unstack . . . 145
use . . . 147
wait . . . 147