# ScenarioMaker 1.5

the world builder
v 1.5 by Luca
Accomazzi

# Requirements

### Hardware and software requirements

ScenarioMaker requires a Macintosh with a 68020 or better processor, and a color screen capable of displaying 256 colors or more. A fast Macintosh (68040 or PowerPC) is suggested. A 13” (or bigger) screen is strongly suggested.
ScenarioMaker, just like the Dream application, requires System 7 version 7.0 or better. The ResEdit application, (version 2.1 or following), from Apple, is required to create some components of a scenario.

### Multiple monitor users

ScenarioMaker does support multiple monitors. Please remember a few technical considerations:
1. ScenarioMaker’s icons palette (i.e. windoid) always appears on the main screen (the one with the menu bar) and is sized so as to fit that screen.
2. If your main screen has less than 256 colors, please move the windoid to a screen that does.

### Multisync monitors and Radius Pivot users

If you use the Monitors control panel to change the resolution of your monitor, or (Pivot users only) you pivot your screen during usage of ScenarioMaker, the icons palette (i.e. windoid) will appear either too long or too short for your monitor. To correct, press the ‘R’ key on your keyboard. ScenarioMaker will immediately adjust its windoid.
For technical reasons, this command only works if you have an opened window (either an arena or a place under construction will do).

### Knowledge required

To create game scenarios with ScenarioMaker you need a working knowledge of ResEdit, the resource editor from Apple Computers. Addison-Wesley has a good book on the subject, which includes the software on a diskette.
Of course you’ll need both a copy of the Dream application and a knowledge of its mechanics (that is, you must have played at least a game adventure).
No programming is required to create a scenario.

### Legalese considerations

Apple, Macintosh, Radius, Pivot, Addison Wesley, and almost all uppercase terms used in this manual tend to be trademarks or registered trademarks of their respective owners. Those owners tend to be huge, wealthy companies whose staff include blood-thirsty lawyers by the dozen. The lawyers are usually kept inside small rooms, and fed only once a day with stale bread and

water; then they are thrown at those guys who use the copyrighted, trademarked and Jeeze-knows-what terms without including a note such as this, and strange symbols (like ©; ® and ™).
I do hope I satisfied the legal requirement of the above-mentioned firms. If I didn’t, a short, sharp whistle thrown in my direction would be more than enough to make me comply. Whimpering.

# What's in a Dream?

**Inside a Scenario**

A game scenario for the Dream game system is a Macintosh file which contains a description of the monsters, places, spells and other entities which the player will meet during game play. Every entity is contained inside a Macintosh resource. The resource is characterised by a type (four characters; e.g. 'Shop' for a shop, 'Mstr' for a monster) and a numeric identifier (e.g. the one 'Mstr' whose numeric ID is 1000 might be a kobold).
Take a look: drag a Dream scenario upon the ResEdit application icon. You might use "Spirit of Darkness", one sample adventure which is included with the Dream application itself.

You'll see, inside a window, the various resources which comprise a scenario. They include:

- 'Aren': an arena (that is, a room where fighting takes place);
- 'icl8', 'icl4' and 'ICN#': icons for monsters, objects ad game locations;
- 'Info': Information and instructions about the scenario, created by the game designer (you) for the Dream gaming system.
- 'MDeL': a monster decision list, that is, a list of orders for a monster to follow;
- 'Mstr': a monster type;
- 'Nctr': Encounter;
- 'Obj ': an item (either sold at a shop or found inside a monster's lair);
- 'PICT': a picture showing a place in full detail
- 'Plac': a place to visit (that is, a map as show in the Dream main window);
- 'Ridl': a question to be asked to the player, a list of possible answers and the consequences of each answer (for short "a riddle")
- 'Shop': List of items for sale inside a shop;
- 'SMkr': Information on the scenario created by the ScenarioMaker application for i own purposes.
- 'snd ': Digitised sound, used to make the monsters roar or speak;
- 'Spel': a spell (either a wizard's or a cleric's one);
- 'STR#': Collection of tips which the adventurers will be given inside inns.
- 'TEXT', 'styl': description of the places. To be shown in the description windoid and spoken aloud by Dream
- 'Trap': a mechanical or magical device;
- 'vers': Scenario version;
- 'Wndr': Wondering monsters list;

Not all scenarios will include every kind of resource listed before. A few very complex scenarios might include other kinds of resources to obtain a sophisticated effect known as "overriding", which we will discuss in a following chapter of this manual.

Most resources of the kinds seen before can be created by simply filling in a template (that is, a dialog box with fields). For example, let's try to see what makes a monster tick. Using ResEdit, open the file 'ScenarioMaker.rez', which you found in the same package which included this manual and the ScenarioMaker application. Now, double-click on the 'Mstr' icon inside the "Spirit of Darkness" window. You'll see a list of monsters. Choose one, and open it with another double-click. A window will open showing the description for this monster. You won't understand much, for

now: this manual will enable you to understand how the template you're looking at turns into a dangerous foe for player characters.

**Checking a scenario**

The best new feature for ScenarioMaker 1.5 is the availability of a "Check scenario" feature. Use it often, use it well: ScenarioMaker will take a long look at your work and tell you what's still missing to make a finished product. It's the next best thing to shipping a copy of the scenario to the Dream designers (yes, you can do that, also: we'll be glad to help, but ScenarioMaker can do this every time you ask and be finished in a few seconds)

The inspection is very through: there isn't much that can escape ScenarioMaker's attention.

**Warning**: Checking only makes sense when the scenario is open, but there's neither a place, nor an arena, shown in the main (construction) window. Close all windows before you check your scenario.

When ScenarioMaker is finished checking (it might take a few minutes: the more complex your scenario, the longer the search), it will report its findings in a dialog box.

The box has a list with the problems. If you click on an item, an explanation appears. Problems are divided in two categories: "warnings" and "errors". A warning might be ignored: it may be that you known perfectly well what you are doing; ScenarioMaker is only reporting that something strange, or potentially dangerous, is inside your scenario.

For example, ScenarioMaker might report:

*WARNING: You created a Place, but there is no reference to it in any other place, so the player characters won't ever be able to reach it*

There is normally no reason to have an unreachable place inside a scenario. Still, maybe you aren't finished creating your scenario, and are planning to add the references to that place later on.

So, in general, a warning means that you'd better double check your scenario and make sure that you are very sure of what you're doing.

Errors, on the other hand, are problems that will make your scenario break inside Sword Dream. For example:

*ERROR: There is no Place with an ID of 1000. Such a place is strictly required, since the game play is supposed to start from there*

In that case, ScenarioMaker is reporting a problem that you ought to fix as soon as possible. Your scenario won't work until you fix it.

**Required resources**

Every single game scenario must contain the following resources:

- 'Plac' ID 1000: the very first place where adventuring begins.
- 'PICT' ID 1128: "About this scenario" splash screen. It is suggested that the PICT i four-bit, grey scale image.

- 'Info' ID 1000: Information about the scenario from the scenario designer to the Dream game system. This is strictly necessary: the Dream application will assume standard values for the parameters and won't bomb even if there is no 'Info' resou inside the scenario, but the information inside the Info resource are needed to kee Dream and its Conflict Resolver working well together..
- 'ICN#': black and white icons for locations, monsters and objects.

**A few rules**

Under the MacOS™, resources must have ID between 128 and 32767. Dream furthe narrows this numeric range. Resources in a scenario must have ID between 1000 an 9999. All other valid IDs are reserved for usage by the Dream engine itself.
All resources should  be kept purgeable (this isn't strictly necessary — that is, the ga won't bomb — with current versions of the Dream application, but game designers should comply if they wish to stay compatible with versions beyond 1.5).

**The rest of the rules**

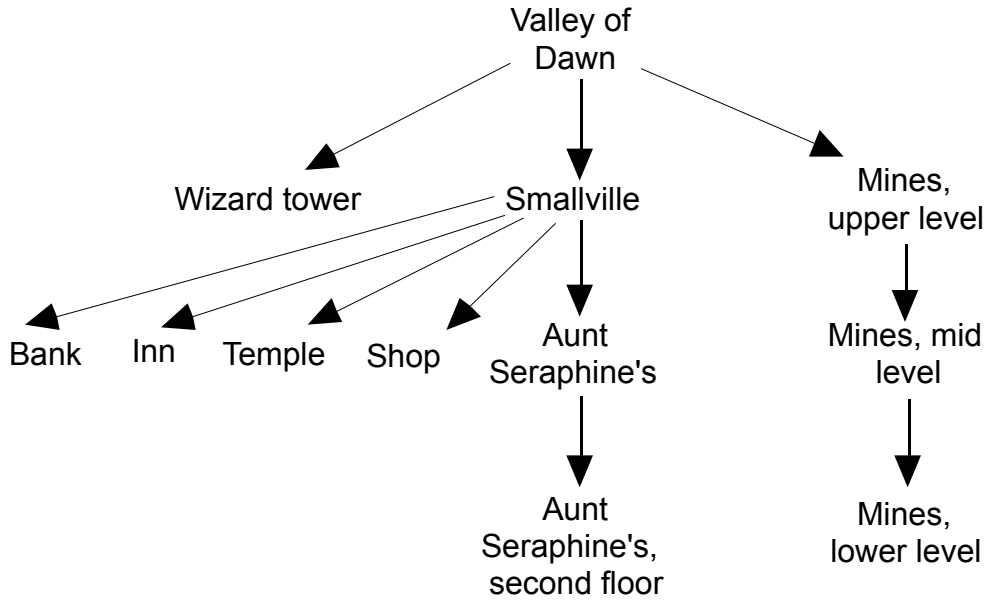
Keep in mind that the Dream application is distributed has been available for more than a year now in its first version, v 1.0.2. Dream 1.0 was included in at least four CDs (two shareware collections in the US, one in France, and one electronic magazine). Thus, it is very widespread.
If you wish to create a scenario compatible with the old version 1.0, avoid using those Dream features which are marked "version 1.1 (1.2, 1.3... 1.6) or following". If you do, Dream 1.0 will be able to play the scenario.
If you forfeit compatibility with the old version 1.0, you gain use to a slew of new feature and can use everything detailed in this manual (don't mind the references to versions between 1.1 and 1.4: those were never made available outside a small group of friend and beta testers in Italy).

There isn't much more to be said. To create your game scenario, you must simply create a new scenario file (the ScenarioMaker application will gladly do this for you, or you may use ResEdit), and fill it with resources.
In the following chapter we will see much of the internal structure of Dream and the concepts on which it is based. This knowledge will form the basis on which you'll build your scenarios.
Next, we'll follow a tutorial, so that you'll be able to build a simple scenario and get the taste of a session with ScenarioMaker.
A smallish scenario, "Akko's place"; is available on request to Designers: There you can create characters of every class, have them easily reach any experience level, and buy them very powerful magic items. Thus, you can create characters for game testing. To receive a copy, contact the Dream Team.
The rest of this manual is a reference guide; it's mostly devoted to the single resource types: a paragraph for each, describing what it is used for and how to create one. In its last part, this manual will give away a few tips and tricks, suggestions and other miscellaneous information.

# Inside Dream

**Map of the scenario**

One of the first things you should create when getting ready to write a scenario is the map of the places that comprise the scenario. I'll put here the map for "Spirit of Darkness", so that you may better understand what a scenario map is.




Each of the boxes in the graph above is known as a "place". The arrows connect places: if there is an arrow between two places, then the place above contains an entrance to the place below. Exit from the place below will always take the characters to the place above, from whence they came.

The depth of the map is not an issue. You can create a twenty level dungeon, if you like, or a skyscraper with sixty floors. You can also reuse some of the components. You might, for example, have two towns in the valley, and put the same shop in both. Players entering the shop from town 1 will always exit to town 1, and players entering from town 2 will find themselves back to town 2 when they exit the shop. To them, there will be two identical shops in the two towns.

The one rule that can't be broken when designing a scenario map is: a place is contained in a place; when exiting a place the characters will return to the place from where they came.

There are also a couple of conventions to be remembered: first, when the characters exit from the topmost place, they are exiting the scenario. Dream will ask the player if he/she wishes to continue playing another scenario or exit the game session.

Second, the topmost place must have a numeric identifier of 1000. Dream will put the player characters inside the place with ID 1000, whenever the player starts a new scenario.

## The characters

Whenever you have to refer to a player character class, make use of the following table:

| Class | Numeric identifier |
|---|---|
| Fighter | 0 |
| Ranger | 1 |
| Paladin | 2 |
| Rogue | 3 |
| Cleric | 4 |
| Wizard | 5 |
| Unclassed | 6 |

The “unclassed” specifier can be used with Extraordinary Characters (see below).

The available races for player characters use the following numeric IDs:

| Race | Numeric identifier |
|---|---|
| 0 | Human |
| 1 | Elf |
| 2 | Dwarf |
| 3 | Gnome |
| 4 | Non humanoid |

Code 4 is reserved for Extraordinary Characters (see below).

## Fighting

### Attack forms

When fighting takes place, a creature can attack another using one of a number of attack forms. Dream supports the following attacks (note that only a few of those can be enacted by a player character):

| **Code** | **Kind of attack** |
|---|---|
| 11 | Illness, disease |
| 12 | Curse |
| 13 | Special ‡ |
| 14 | Death Magic |
| 15 | Missile weapon (arrows, darts…) |
| 16 | Magic weapon, +5 to hit |
| 17 | Magic weapon, +4 to hit |
| 18 | Magic weapon, +3 to hit |
| 19 | Magic weapon, +2 to hit |
| 20 | Magic weapon, +1 to hit |
| 21 | Weapon (sword, mace, axe…) |
| 22 | Energy (magic missiles, energy burst…) |
| 23 | Experience level drain |
| 24 | Turn to stone † |
| 25 | Acid |
| 26 | Poison |
| 27 | Healing magic (cure graze and better) |
| 28 | Electricity (shocking grasp, Galvani’s spark…) |
| 29 | Frost (Ice storm…) |
| 30 | Fire (Fireball, dragon breath weapon…) |
| 31 | Mind attack, psionics |

† Turn to stone is only available from version 1.5.

‡ All attacks which can’t be assimilated to one of the above are said to be “special” attacks. For example, the Slow spell, which slows monsters, constitutes a special attack.

A creature can be protected, either by its monstrous nature or through magic, from any of the attack forms. Protection can be either total (for example, you shouldn’t be able to affect a skeleton with death magic, nor a giant worm with a mind attack) or

partial (for example, a character protected via Protection from fire spell can still be affected by fires, but takes only a half damage).

### Damage

Whenever it is necessary to inflict damage on someone, either character or monster, Dream must be told how heavy the attack form is. Typical damage ranges are:

1 to 4 hit points for a knife
1 to 8 points for a long sword
1 to 12 points for a halberd
5 to 30 points for a fireball created by a fifth level wizard
20-200 points for an Armageddon spell created by a twentieth-level cleric

As you can see, damage is given as a range of values: there's a minimum and a maximum. To assess a damage in the 1-6 range, Dream will generate a random number throwing a software six-sided die; you can instruct Dream to do so writing down inside the appropriate template that a weapon (or a spell) does damage with one (1) throw of a six (6) sided software die.

By the way, even if there are no physical seven-sided dice in existence, Dream might as easily create one on software for you.

To generate a damage ranging from 2 to 12, you can instruct Dream to throw two dice with six sides. Dream will then sum the results. You might know that, by throwing a couple of six-sided dice, results in the range 6 to 8 are the most likely: keep this in mind when designing damage ranges. For example, with the 5-30 range quoted before (that is, five six-sided dice) it is very likely that the actual damage will be between 14 and 21; other values are much harder to come by.

### Spells

Spells fall into one of two broad categories: a few are instantaneous (take place immediately and have permanent effect, like Cure Graze, Curse Undead or Blast Fireball), others have a variable duration and their effects vanish when the time is over (like Protection from Fire or Feign Death).

### Material components

As you know, most spells require a material component to be cast. The following codes are used to specify the material components.

| Code | Material |
|---|---|
| 0 | No material component required |
| 128 | Holy water |
| 129 | Amber |
| 130 | Bat guano |
| 132 | Glass lens |
| 133 | Magnet |
| 134 | Mushroom |
| 135 | Quicksilver |
| 136 | Silver mirror |
| 200 | Rations |
| 201 | Cure potion |
| Any other | The item whose resource id is equal to the code. Use only items defined in your scenarios or listed |

in the “Dream Database” chapter.

Note that any object can work as a spell component. This creates exciting possibilities: think of a scenario where the players know that only a very special, very secret spell can save the world. They have to locate the scroll where the spell

is written, then they must locate a wondrous item which is required to cast the spell (what about an edelweiss, lost somewhere among huge mountains?)

**Spell level and IDs**

Both wizard and cleric spells are characterised by a level. The higher the level, the more powerful the spell is. There is no relation between a spell resource ID and its level in Dream 1.5.

**Range**

An attack (by spell or otherwise) is also characterised by a range. If the attack can only be launched on somebody close to the attacker (like the thrust of a stiletto), then the range is said to be 1. This means that the square the defender is on can be at most one unit distance from the attacker's one.
All melee attacks have a range of one.
An action which is only attempted by a creature on itself, like the Feign Death spell, is said to have a range of zero.
Attacks from afar, like an arrow from a bow, a dart, or a Magic Missile spell, obviously have a range greater than one.

**Area of effect**

Another important characteristic of a spell is the area of effect. The following possibilities are supported by all versions of the Dream engine:

| Code | Area of effect | Example |
|---|---|---|
| 0 | The spellcaster | Light |
| 1 | Any one member of the spellcaster's group | Cure graze |
| 2 | Any one enemy | A sword stroke |
| 3 | All members of the spellcaster group | Cure all wounds |
| 4 | All enemies | Curse all undead |
| 5 | Circular area | Fireball |
| 6 | Square area | Ice storm |
| 7 | Straight path | Call Lightning |

**Saving throws**

A few spells always work their magic: Light, for example, invariably results in a magic form of illumination to appear above the caster's head. Some other spells might not work at times: such is the case for Slow. These spells are said to have a "saving throw" (the term is taken from traditional role playing games, where the working of the spell is adjudicated via the throw of a die. If the roll is good, the creature which is target to the spell was "saved" from the spell effects). The saving throws negates the effects of the spell (if the target is a monster, by the way, the higher the number of hit dice, the most likely it is the monster will be saved).
Yet other spells can be "partially saved" against. Take "Blast Fireball", for example: at best, the target might take only a half as much damage as prescribed (this excludes magical defences like "Protection from fire", of course). These spells are said to have a "save for one half" saving throw.

**Other spell attributes**

A spell may constitute a kind of attack (a fireball most certainly could be considered an attack by the unfortunate goblin caught in the middle). So, every spell is attributed a code from the table in Inside Dream/Fighting/Attack forms. It is known as the spell “kind code”.
Spells ought to have an associated icon. If you browse through the spells inside the Dream application, version 1.5, you’ll notice that there’s no icon for them. Icons will

be shown in future releases for the game, though, so you should create one if you wish to stay compatible.

**Spell code selectors**

All other things being equal, Dream needs a way to distinguish between variations of the same magic. For example, “Slow Poison”, “Cure Poison” and “Poison” spells all represent different forms of the same magic, and are distinguished via a special spell code.

Listed below are the supported codes. Unless otherwise noted, all versions of the dream application support all codes.

For all spells

-2: “Rune”-kind protection from some attack form. This spell will block the first enemy spell of the specified kind, then automatically expire. Code -2 can’t be used with petrify, poison an similar attack forms, whose effect is not physical damage. Code -2 is only supported by version 1.5 and greater.

-1: Protection from that attack form. Reduces all damage to one half, rounded down, for the duration of the spell.

0: Attack of the spell type, which does damage as specified. For example, if the spell kind code is 30 (fire) and the code selector is 0 (attack) then you are creating a fire-based attack spell, like Fireball.

For Illness, poison, turn to stone:

1: Slow down effects of the attack form

2: Cure effects of the attack form

3: Afflict target with the attack form

For special magic

1: Light

2: Identify

3: Slow

4: Haste

5: Feign death

7: Restore lost experience levels

8: Resurrect

9: Dispel magic

10: Fly

The following forms of special magic are only supported by Dream version 1.5 or greater:

11: Soul search (detect alignment)

12: Change the target’s armor class

13: Create a clone of the target

14: Vision (see the character’s surroundings)

15: Teleport. Use the spell “reserved” field to specify the destination. Use 0 for “back to the previous place”, 1 for the topmost place in the scenario, or any

place id.
16: Destroy cursed items
17: Invisibility. Use the spell “reserved” field to specify the spell scope. Use 0 to indicate that invisible enemies should become visible, 1 to make the spell target invisible.

All other codes are reserved. Don’t use them no matter what.

**Objects**

Objects can be single (like a sword) or multiple (like arrows, which are bought by the dozen). When an item is multiple, the number of singular entities which compose it are written inside the “Nr. charges” field of the object template.

**Weight and other things**

To define an object for use inside Dream, you must provide:

• An icon (which will be shown in the player information sheet and/or inside shops).

• A price. If the item is put in a shop, the shopkeeper will ask for the price you list. If the item is sold by the player, the character will get nine tenths (that is, 90%) of the listed price

• A weight. Remember that the character won’t be able to carry too much weight. You might have a bit of fun creating very useful items which are so weighty that carrying them around is a problem.

• A public name. This is the name by which the player will know the object if he finds it. It should be quite vague (like, “a glass sphere”).

• A private name. This is the name which is revealed through use of the Identify spell. It should be very specific (like, “Crystal ball of scrying”). Items bought in shops are known to the player by their private names: it is supposed that the seller told the characters all about the item.

• A list of body parts where the item can be stored. For example, a helm should be allowed in the sack, in hand and above the head. Be sure to allow for the sack at all times, or the characters will be unable to get it (all items end up in the sack when grabbed). As a general rule, allow everything to be kept in hand (otherwise it will be impossible to cast an Identify spell).

• A list of character classes which will be allowed to use (don) the item. Be creative, but remember the general rules in the following paragraph:

**Character classes and items**

1. Wizards shouldn’t be allowed to use armor. Their armor class should be kept very low, or they would quickly become the most powerful class. Similarly, allow them to use only the puniest weapons. All magic items should be open to them, with the only exception of magic weaponry and armor.

2. Clerics should be allowed to use almost all kind of armour and weapons, but the most powerful. They should be limited in the magic items they get access to.

3. Fighters should be able to use every kind of armor and weaponry, both magic and mundane. No magic item should be allowed to them, with the only exception of magic weaponry and armor.

4. Paladins and rangers should follow the same ground rules set for fighters, but rangers should be unable to use the best armor and able to use some magic items. Paladins should be unable to keep precious and luxury items, like jewellery, which wouldn’t befit their lifestyle.

5. Rogues should be able to use almost all weaponry, but be very, very limited as to armor (they need to be nimble and agile, and one simply can’t run

around inside a full plate mail). Rogues ought to be the only character class allowed to use throwing weapons.
6. Don’t inflate magic. A magic item should be a revered prize, not the rule. (there is only one magic item in “Spirit of Darkness”, after all). If players complain they don’t get enough magic items, that’ll whet their appetite for more adventure.
7. Don’t allow players to become invincible. You can easily create a magic ring which will give them all an armor class of -10. So what? What are we going to throw at them after that, dragons by the dozen? A character should find extremely hard to reach armor classes below 0. A fifth level paladin should be very happy to own a

single shield +1 and a long sword +1. Armor class -5 should be a revered dream until experience level 10 or so.

**Item categories**

Items can also be put in any of the following categories.

• Magic: a magic item is entitled to better protection against crushing blows.

• Weapon: a weapon is an item which can be wielded and used against monsters. See below for more information specific to weapons. If you state that an item is a weapon, be certain to provide the amount of damage the weapon shall inflict on enemies. If an item is both magic and weapon, insert a positive value in the “base damage” box. For a long sword to do damage in the 1-8 range, select base damage 0, number of dice 1, dice size 8. Then you might create a +1 magic long sword: type the save characteristics and put 1 in the base damage box. The magic sword will do damage in the 2-9 range.

• Armor: an armor is an object that will protect the owner from being hit. See below.

• Throwing weapon: a throwing weapon, like a bow, will propel ammunition, like an arrow. See below.

• Ammunition: an ammunition is used in conjunction with a throwing weapon. See below.

• Gives light: torches, lamps and lanterns are light-bringing devices, but you might also create a magic beaming item. To create a standard luminous object, put it into the “gives light” category, then type the number of minutes it will work before discharging in the “Number of charges” box. To create a magically resplendent item, put it into the “gives light” and “magic” categories, and write a zero in the “Number of charges” box.

• Rechargeable: a torch can’t be recharged, but a lantern might.

• Cursed: a cursed item can’t be thrown away or sold by the owner once wielded. In Dream 1.0 there’s no way a character can get rid of a cursed item, so use this feature with wisdom.

• Intelligent: not yet implemented

• Food: if it’s in the food category, then it is nourishing. Dream looks for items of this kind every 24 hours, and removes one charge.

• Scroll: a scroll is a magic parchment inscribed with all the explanations needed for a spellcaster to recreate a spell. If a spellcaster owns a scroll, he or she may copy it inside his or her spellbook (then the scroll vanishes). So, a scroll is a most precious item for the player: keep those very rare.

• Book: a book is an item with an associated TEXT piece. When the player “uses” a book, the Dream engine shows the text inside the text windoid.

• Map: a map is an item with an associated PICT piece. When the player “uses” a map, the Dream engine shows the picture inside the PICT windoid.

**Armor**

An armor is a special item which protects the wearer. You can create both

armor of the mundane type (say, a full plate mail) or magic (for example, a magic protection ring).
The most important attribute of an armor is the bonus it gives to the character's armor class. In "Spirit of Darkness", rogues can buy a leather armor, which is described as an armor with bonus 2. This means that wearing leather armor the AC of the character will drop by two factors (in other words, the character will be 10% harder to hit). So, a rogue with standard armor class (10) will be given armor class 8 when donning the leather armor.

## Weapons

Weapons, like armor, are a special kind of objects.
You, the scenario designer, can create any kind of weaponry you like: swords and maces, bows and lances, magic items. Under Dream, it would even be possible to create a gun machine (but please don't: it would spoil most of the fantasy).
In Dream, weapons can either work for melee combat or for firing missiles. As of version 1.0, you are limited in that you can't create a magic throwing weapon, like a magic bow. Still, you can create a magic ammunition, like a magic arrow.

Throwing weapons may be self-propelled (e.g. darts) or need a throwing instrument (e.g. arrows, which need a bow). We are going to call the throwing weapons that need a throwing instrument "ammunition" (genial, huh?).

## Monsters

Monsters are the creatures that characters meet and interact with. Most of those will be evil creatures, thirsty for the blood of the player characters. A few might be willing to help them.

A subset of the characteristics which describe player characters are used to describe the monster. Other characteristics are specific. The characteristics which are common between PCs and monsters are:
Intelligence,
Alignment,
Armor Class,
Level,
Damage,
Name.

## Monster alignment

Monsters, just like characters, are characterised by an alignment, which describes their ethos. Alignments are very important even now, and will become even more important with future releases of the Dream gaming system.
Good monsters might help the PCs (if they find a similarly aligned PC with a high charisma). Chaotic monsters won't collaborate between them; lawful monsters might regroup, try to flank their adversaries, and flee *en masse*.
When you wish to create a monster, you should use the following codes. The alignments are explained in the Sword Dream manual.

| Code | Alignment |
|---|---|
| 0 | Order and evil |
| 1 | Plain evil |
| 2 | Chaos and evil |
| 3 | Plain chaos |

| | |
|---|---|
| 4 | True neutral |
| 5 | Plain order |
| 6 | Chaos and good |
| 7 | Plain good |
| 8 | Order and good |

## Monster intelligence

Intelligent monsters will behave intelligently. Stupid monsters will perform poorly.

For example, all monsters tend to panic and flee when they are hurt and the fight is going badly. Still, non-intelligent monsters tend to act like the common roaches:

they flee very quickly for a while, then stand still, hoping that the enemy lost their track.
Monsters with an animal intelligence might flee right away from the character which punished them, and doing so will easily run into some other character.
Dumb monsters will avoid all characters, but won't plan their retreat; intelligent monsters will try to make the quickest possible exit.

Whenever you create a monster, you'll have to type an intelligence code. Choose it from the following table.

| Code | Intelligence level | Example |
|---|---|---|
| 0 | Non intelligent | Mushrooms |
| 1 | Animal intelligence | A bird |
| 2 | Semi intelligent | A very smart dog |
| 3 | Low intelligence | A goblin |
| 4 | Average intelligence | A man |
| 5 | High intelligence | A unicorn |
| 6 | Genius | A high level wizard; a demon lord |
| 7 | God-like | An avatar for a god |

In Dream 1.5, monster behaviour is implemented up to intelligence level 4 (average). Don't expect monsters to act particularly smart in any case. You are free to use codes above 4: in future releases of the Dream engine, monsters will be smarter if you employ those codes.

Remember that intelligent monsters will be harder to kill; so, all other things being equal, they should be rewarded with more experience points.

**Monster behaviour**

Normally, you'll want your monsters to behave aggressively. Monsters usually start attacking the player party, and might panic later.
Under version 1.5 and following of the Dream engine, the Scenario Designer is free to specify a different behaviour for monsters. Consult the following table:

| Code | Meaning |
|---|---|
| P | (Panic) The monster will flee the nearest character. Stupid monsters will flee the last character which hit them, disregarding all other considerations |
| F | (Fight)Standard behaviour. The monster tries to hit a character. If all characters are beyond range, the monster walks toward the nearest character. If two or more characters are equally distant, the monster chooses one at random. |
| G | (Group) The monster tries to get near to another monster |
| S | (Stand) The monster stays where you placed it. It will try to hit a character as soon as one is in range. Standing monsters can panic as usual. This behaviour is particularly useful for bodyguards to a spellcaster or for fighter-types defending a row of archers. |
| H | (Helpful) The monster is friendly. It will get near the group, select a similarly-aligned character, touch him or her. If you have enabled the "can cure" monster ability, it will cast a cure spell. |
| R | (surRound) The monster will coordinate with the other monsters and try to corner the player characters, surrounding them.<br>This behaviour is not yet available. It will be implemented under version 2.0 |

of the Dream engine.

## Controlling a monster

You'll normally be happy to leave the monster to the Dream system. Under version 1.5 and following of the Dream application the designer can also take control of the monster, specifying each and every action to be performed by it. If you choose to attach a Monster Decision List (MDeL for short) to a monster, you'll be allowed to choose what a monster will do. Your list of actions will be executed, one per melee round, by the monster.
Via a MDeL you can create spell-casting monsters, talking monsters and other sophisticated adversaries for the player.
Each move if the list contains a verb, a direct object and an indirect object. For example, you can order the monster to cast a "cure graze" spell upon itself. In this case, "cast" is the verb; "cure graze" is the direct object, "upon yourself" is the indirect object.
The direct and indirect object are specified using the numeric identifiers from the following **numeric object table**:

| Code | Meaning |
|---|---|
| 10 | The nearest fighter |
| 11 | The nearest ranger |
| 12 | The nearest paladin |
| 13 | The nearest rogue |
| 14 | The nearest cleric |
| 15 | The nearest wizard |
| 0 | The nearest Evil and Lawful character |
| 1 | The nearest Evil and Neutral character |
| 2 | The nearest Evil and Chaotic character |
| 3 | The nearest Neutral and Chaotic character |
| 4 | The nearest Totally Neutral character |
| 5 | The nearest Neutral and Lawful character |
| 6 | The nearest Good and Chaotic character |
| 7 | The nearest Good and Neutral character |
| 8 | The nearest Good and Lawful character |
| 20..27 | The character whose icon is at the topmost, second, third... bottom-most place in the Roster |
| 30 | The character with the least HPs |
| 31 | The nearest character |
| 50..99 | The first, second, third… nth monster alive |
| 100 | Myself |

Warning: if there's nobody fitting the description (e.g., you are asking a monster to walk towards the nearest fighter, but there's no fighter in the player group), the Dream engine will substitute code 31 (in the example above, the monster will march towards the nearest enemy character.
The verbs
The following table lists the verbs you can use in a Monster Decision List:

| Verb | DIRECT object | INDIRECT objects |
|---|---|---|
| 8: move North | None | None |
| 2: move South | None | None |
| 4: move West | None | None |
| 6: move East | None | None |

| | | |
|---|---|---|
| 5: stand still | None | None |
| 7: move North-West | None | None |
| 9: move North-East | None | None |
| 1: move South-West | None | None |
| 3: move South-East | None | None |
| M: move towards… (or try to hit, if in range) | Code from the numeric object table | None |
| K: Cast a spell | Spell code (see the next chapter, the Dream Database, for a list of available spells) | Code from the numeric object table |
| I: Let the Dream engine choose an appropriate move | None | None |
| S: Speak | Id of a TEXT resource to show | Id of a ‘snd ‘ resource to play (or zero if none) |
| F: Flee from | Code from the numeric object table (must be below 50) | |

**Wandering monsters**

Inside monster-infested places, the characters might run into wandering monsters. You can specify which places have wandering monsters, and which kinds of monsters can be met there. The characters won't be allowed to rest or wait in such places.

In Dream 1.0 there's a fixed chance of meeting wandering monsters (on the average, one encounter for every 36 moves). If your scenario runs under Dream 1.1 or greater, the Info resource created by the scenario Designer tells to the Dream engine how often an encounter will occur.

For each monster definition, you'll have to specify the maximum number of monsters in the wandering group. For example, if you say that goblins form groups of up to six monsters, the character will face groups of either one, two… or six monsters. Try to be logical, and remember: monsters which believe in order will tend to form groups. Chaotic monsters will be more likely to stay alone and aloof. Wild beasts might hunt alone, in couples, or in a pack.

# The Dream Database

Resources of common use (all standard spells, the items which are usually found inside shops, some common icons and more) are kept inside the Sword Dream DataBase file (Dream DB for short). The Dream DB is a part of the Dream package from version 1.1 only: version 1.0 kept those resources inside the Dream application: thus under 1.0 it isn't possible to safely transfer non-standard items and spells from scenario to scenario (when a scenario is discarded, the scenario file is closed and all resources inside it are forgotten).
Now, a scenario Designer like you can make use of the Dream DB, and even opt to include his or her ideas inside the Dream DB.
An example or two should help make the concept crystal clear. You wish to include a shop in your scenario: why bother selecting the available items one by one? There's a standard shop inside the Dream DB: just use it. Need a key to that safe? There's an icon ready for use in the Dream DB.

**Icons**

The following icons are available inside the Dream DB. They are guaranteed to be available in each and every future versions, and will be graphically enhanced if needed. Use them as needed (but remember that the if you do your scenario will only work under version 1.5 or following of the Dream application).

| Numeric ID | Icon |
|---|---|
| 201 | Magnet (metal bar) |
| 202 | Vial (mercury) |
| 203 | Question mark (use it during development for items whose icon isn't ready yet) |
| 206 | Food |
| 207 | Glass lens |
| 208 | Fungi |
| 209 | Mirror |
| 210 | Iron rations |
| 211 | Long sword |
| 300 | Potion |
| 301 | Amber |
| 302 | Bat guano |
| 303 | Lamp |
| 304 | Shield |

| | |
|---|---|
| 305 | Short sword |
| 306 | Mace |
| 307 | Long bow |
| 308 | Quiver |
| 309 | Darts (shuriken) |
| 310 | Book |
| 311 | Sling |
| 312 | Staff |
| 313 | Gem |
| 314 | Stones |
| 315 | Iron spheres |
| 316 | Dagger |
| 317 | Magic staff |
| 318 | Magic wand |
| 319 | Leather armor |
| 320 | Plate armor |
| 321 | Splinted armor |
| 322 | Chain armor |
| 323 | Scale armor |
| 324 | Ring armor |
| 325 | Key |
| 326 | Short bow |
| 327 | Two handed sword |
| 328 | Rope |
| 329 | Scroll |
| 330 | Axe |

## Items

| Numeric ID | Item |
|---|---|
| 128 | Holy water |
| 129 | Amber |
| 130 | Bat guano |
| 131 | Manna |
| 132 | Glass lens |
| 133 | Magnet |
| 134 | Fungi |
| 135 | Mercury |
| 136 | Silver mirror |
| 138 | Iron rations |
| 200 | Rations |
| 201 | Potion |
| 400 | Mace |
| 401 | Leather armor |
| 402 | Ring mail |
| 407 | Plate armor |
| 406 | Splint mail |
| 405 | Chain mail |
| 404 | Scale mail |
| 403 | Shield |
| 408 | Bow long |
| 409 | Long bow arrows |
| 410 | Short sword |
| 411 | Dagger |
| 412 | Long sword |
| 413 | Sling |
| 414 | Staff |
| 415 | Stone bullets |
| 416 | Iron bullets |
| 417 | Darts |
| 418 | Bow, short |
| 419 | Shortbow arrows |
| 420 | Axe |
| 500 | Lamp |
| 501 | Rope |

## Spells

## Cleric spells

| Numeric ID | Spell |
|---|---|
| 300 | Curse undead |
| 301 | Cure graze |
| 302 | Slow poison |
| 303 | Cleric light |
| 304 | Soul search |
| 305 | Bless |
| 306 | Damn |
| 310 | Cure illness |
| 311 | Protection from fire |
| 312 | Protection from cold |
| 313 | Feign death |
| 314 | Illness |
| 315 | Multiply food |
| 316 | Protection from electricity |
| 317 | Detect invisibility |
| 318 | Undead protections |
| 320 | Cure poison |
| 321 | Curse all undead |

| | |
|---|---|
| 322 | Poison |
| 323 | Cure wound |
| 324 | Protection from arrows |
| 325 | Prayer |
| 326 | Invisibility |
| 330 | Restore |
| 331 | Prayer |
| 332 | Vision |
| 333 | Saint Sebastian’s Bless |
| 334 | Saint Fakir’s Flemma |
| 340 | Cure gash |
| 341 | Remove curse |
| 342 | Excommunication |
| 350 | Raise dead |
| 351 | Group invisibility |
| 360 | Cure all wounds |
| 361 | Armageddon |

## Wizard spells

| Numeric ID | Spell |
|---|---|
| 200 | Magic Missiles |
| 201 | Shocking grasp |
| 202 | Wizard light |
| 203 | Wizard sword |
| 204 | Mind shield |
| 205 | Freeze |
| 206 | Silver rune |
| 207 | White rune |
| 208 | Blue rune |
| 210 | Identify |
| 211 | Slow |
| 212 | Haste |
| 213 | Mind blast |
| 214 | Galvani’s spark |
| 215 | Fly |
| 216 | Ground |
| 217 | Black rune |
| 218 | Red rune |
| 220 | Wizard sword +1 |
| 221 | Identify all |
| 222 | Blast Fireball |
| 223 | Call Lightning |
| 224 | Dispel magic |
| 225 | Acid Arrow |
| 226 | Hailstorm |
| 230 | Slow all |
| 231 | Mind storm |
| 232 | Fly all |
| 233 | Ground all |
| 234 | Teleport |
| 240 | Flesh to stone |
| 241 | Stone to flesh |
| 242 | Faraway lightning |
| 243 | Teleport Home |
| 250 | Wizard’s Wrath |
| 251 | Crushing word |
| 252 | Simulacrum |
| 260 | Killing word |

**The conflict resolver**

In order to better understand the requirements of a scenario, we must discuss the inner workings of the Dream application at large.

Let's suppose that a player faces and tries, one after the other, two different scenarios. During game play of scenario A, our player finds a magic sword. During game play of scenario B, the player is rewarded with a gem. Unluckily, both items (sword and gem) were given item ID 1000 by their respective Designers (who don't know each other and, thus, can't talk about the problem). If such an unfortunate combination of events happens under Sword Dream 1.0, as soon as the player quits the first scenario and enters the second, the sword is effectively forgotten and substituted with the gem. That is, the sword turns into a gem.

This is, to say the least, Not A Good Thing. Sword Dream 1.5 can prevent it happening... but it needs your help.

**What's the Conflict Resolver?**

Under version 1.5 and following, you can add an AdDB resource to your scenario. There you should list all of the items and spells that a player might wish to take from your scenario into the following. Sword Dream will do the hard work from there.

Back to our example. The Designer of the first scenario indicates (in her AdDB resource) that the sword might be kept by the player. So Sword Dream copies the definition for the sword in the Dream Database as soon as scenario A is opened: this way the sword won't be forgotten when the scenario is finished and closed.

When the player switches to scenario B, Sword Dream sees the gem with the same ID as the sword, realises the danger, and acts.

There's a piece of very sophisticated code, inside Sword Dream, called the Conflict Resolver, which gets executed in these cases. In short, the resolver sees that the two items share a single numeric ID, and changes one of the two to a spare ID. For example, the Resolver gives an ID of 20000 to the sword and brings every reference to it in memory up to date. The character holding the sword finds himself holding item ID 20000, and the sword definitely doesn't become a gem.

**Life with the Resolver**

As I said before, Sword Dream needs your help to make the Conflict Resolver work. Whenever you create a resource which could be use in other scenarios, you must tell it to the Sword Dream application (by adding the resource signature to the AdDB list). Whenever Dream will meet your scenario in some computer, it will copy your resources inside that computer's Dream DB.

You might wonder how you can recognize if a resource that's part of your scenario needs to be included in the AdDB list. That's easy. Let's suppose that you create a custom spell for use with an ingenious trap. You know that the trap is part of your scenario, and your only. So there's no need to put the spell

in the AdDB.
If, on the other hand, you create a “Turn every enemy monster into a frog” magic wand, the “turn target into a frog” spell must be put in the AdDB (as must the wand ‘Obj ’ definition), because the wand might be kept by a character when he wanders into the next scenario.

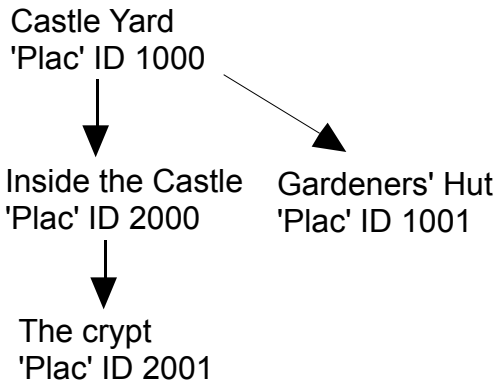
**Limitations of the current Resolver**

The Resolver was created to work out the conflicts between items and spells only (including their icons). That's what a character is likely to keep when he or she switches from scenario to scenario. The Resolver won't help you if you wish to put other things inside the Dream DB (e.g. a monster).
Also remember that the version 1.5 resolver requires that references to icons are listed in the AdDB resource from the least-colored to the most-colored version. For example, if your armor ('Obj ' ID 1500) is represented by icon ID 4000, and you created all color version of the armor (B/W, 256-color and 16-color), then you must insert in the AdDB the following data: 'Obj ' 1500, 'ICN#' 4000, 'icl4' 4000, 'icl8' 4000. You must include the icons, and they must be listed in the order shown. You might also use the sequence ICN#, icl4, icl8, Obj, but no other permutation of references is acceptable.

# Making a scenario: a tutorial guide

In this chapter we shall be creating a very simple scenario, to show the steps involved in creating a Dream adventure. Here's the core idea: a castle is haunted by a vampire. The vampire should prove impossible to defeat by the PCs, unless they discover a special magic item buried inside the vampire crypt. That item can cast a destroying spell at the monster.

**The castle of terror: a map**




**Start**

Using ScenarioMaker we create a new scenario. Type "Castle of Terror" as the name for the new file.

Use the "Grab icons from…" command in the File menu, and select Spirit of Darkness. This will borrow the icons used to depict scenario locations from inside Spirit of Darkness. Of course, if you can draw you can also create brand new icons, using ResEdit, or use the more fashionable 3D icons from Back to Dawn Valley...

Quit ScenarioMaker, for now, and enter ResEdit. Open the following three files: "Scenario Resources.rez", "Spirit of Darkness", "Castle of Terror". The first contains a few definitions we shall be using. The second will come handy for a few ready components, the third is the scenario we are creating.

We are ready to start the creative part!

**Monsters**

Let's say we wish to include a few undead inside the castle. They are supposed to work as servants to the vampire. For a start we can borrow the skeletons used inside "Spirit of Darkness". To do so we'll copy the Mstr resource, ID 1004, which describes the skeletons, and paste it inside "Castle of Terror". We shouldn't forget the appropriate icon: let's also copy the ICN# resource, ID 2001 and the corresponding colour icon: icl8 ID 2001. (To

discover the ID of the icon you can open the Mstr resource and see).
Now, let's create the vampire itself. The first thing we need is an icon. You can create one yourself, or copy the ICN#, ID 2012, from "Spirit of darkness". It isn't used in that scenario, and I put it there so that you'd have something to work with.
Now double check that you have the "Castle of Darkness" window as foremost in ResEdit, then choose "Create new resource" from the Resource menu. A list of alternatives appears. Choose or type Mstr. A template appears.
We will fill in the blanks, following the guidelines detailed in chapter "Resources" of this manual. For our vampire we choose:

Number appearing: 1
Alignment: 2 (that is, Chaos and Evil)
8-sided hit dice: 8 (it should be very nasty)
THAC0: 13 (that's 21-8, where 8 is the level)
Attacks per two rounds: 2 (standard)
Dice of damage: 1
Damage die size: 1 (the real damage is magical)
AC: 5 (a supernatural defence, as it doesn't wear armor)
XP: 1000
Can see invisible: 1, that is true
Can be invisible: 0, that is false
Can fly: 1
Regenerates: 0
Is undead: 1
Intelligence: 5 (high)
Morale: 140 (won't flee)
Name: Count Vlad
Icon: 2012
Special attacks: leave all of those to 0 (that is false, that is "no"), but "Drains levels" and "Fights"
The attack selector code isn't needed for this monster, so leave it to zero. You also must leave to zero all "reserved" and "spare" labelled attributes if you wish to stay compatible with Dream version 1.1 and beyond.
Now for the special defences: Vlad is an undead, so he gets the standard special features of the undead monsters: Mind attack immune, Poison immune, Can't be drained, Immune to death magic, Immune to illness. Being a very high level undead, Vlad is also Immune to cold and Can't be hit (by non-magical weapons).
We're finished. Before closing the window, choose "Get resource Info" from the Resource menu and type a valid ID for the vampire. 1000 is fine. Please also click on the "Purgeable" checkbutton.
You could also create some zombies, if you wish.

Now we can specify what kind of monsters are found wandering inside the castle. Create a new resource, type Wndr and ID 2000 (the same as the castle). Click on the row of asterisks and type Command-K to make room for a monster ID. Type 1004 (it's the ID for the skeletons, remember?)
If you created the zombie type monster click again on the asterisks, press Command-K again and type the zombies ID.
Save the Wndr resource, checking that you gave it a correct ID and attributes (purgeable).

**Spells**
We decided a while ago that the super-secret vampire-destroying magic item

will kill Vlad via a spell. So, let's create this very special spell.
Double check that you have the "Castle of Darkness" window as foremost in ResEdit, then choose "Create new resource" from the Resource menu. A list of alternatives appears. Choose or type Spel. A template appears.
Let's fill in the blanks, reminding that this spell shouldn't be available to characters: it's only meant to be cast via an item.
Level: zero. This is fundamental to let Dream know that the spell is bound to an object.

Kind code: 12; it's a curse for an undead monster.
Reserved: zero. You must leave to zero all "reserved" and "spare" labelled attributes if you wish to stay compatible with Dream version 2.0 and beyond.
Wizard or cleric: this doesn't matter.
ST for none and ST for half: both zero, as we wish to make sure that the vampire gets killed.
Dmg. is per level: 0, False. By the way, keep in mind that an item is considered 12-level when the level is relevant.
Can cast in fight: 1, true
Can cast in peace: 0, false
Area code: here, both a 2 (one enemy) and a 4 (every enemy) are OK. We might wish to choose 4, as this area code will avoid the need to click on the monster.
Distance: We don't wish this to be relevant, so we'll type 10. That is, the monster will be hit no matter the range.
Area size: this doesn't matter.
Damage die size: and Damage die qty: It's vital that this spell kills the vampire, so some huge amount of damage is called for. 20 die 20 should prove more than enough.
Duration base and duration per level: both zero, as this spell is instantaneous.
Mat. required: zero, as it doesn't make sense that a magic item needs some material component.
Code selector: 0
Name: this doesn't matter. The name won't appear anywhere for an item-based spell. You could type Vampire-killer for your own convenience.
Icon: this doesn't matter. Use zero.

We're finished. Before closing the window, choose "Get resource Info" from the Resource menu and type a valid ID for the spell. 1000 is fine. Please also click on the "Purgeable" checkbutton.

**Objects**

We are now to create the magic vampire-killer object. First we need an icon. We can easily create one using ResEdit: a cross will do. Let's give the icon an ID of 2000 (the numbers between 1000 and 1999 are reserved for location icons, remember?)

Next, we'll make up the item itself. Double check that you have the "Castle of Darkness" window as foremost in ResEdit, then choose "Create new resource" from the Resource menu. A list of alternatives appears. Choose or type Obj. (If you type it, please notice that there is a blank after the j. All resource signatures are four letters). A template appears.

Icon: 2000

Magic: 1, true. It casts spells, so it is magic.
All of the following attributes should be left at zero.
Bonus/code: Here we should write the ID for the spell (read the paragraph “items” inside the “Resources” chapter). So, let’s type 1000.
Number of charges: 1. We need the object to disappear as soon as the vampire is killed. It wouldn’t be appropriate that the PCs took the magic cross with them onto another scenario. Moreover, as the related spell is defined inside this scenario, the item wouldn’t work in another scenario where the spell isn’t defined.

Weight: Anything appropriate. 10 would be OK.
Price: The characters aren't supposed to sell the item, so anything will do.
Wear on...: In hand, of course, is appropriate. In sack should also be allowed (1), as with every item.
Which character classes will be allowed to use this item? Clerics, of course: so set Clerics can use to 1. Paladins would be probably appropriate, too, so set Paladins can use to 1.
Public name: A wooden cross
Private name: Count Vlad's Death Cross
Weapon XXX: it isn't a weapon, so write zero everywhere.

We're finished. Before closing the window, choose "Get resource Info" from the Resource menu and type a valid ID for the spell. 1000 is fine. Please also click on the "Purgeable" checkbutton.

**Scenario info**
We must give some information to the Dream application about the inner workings of our scenario.

Double check that you have the "Castle of Darkness" window as foremost in ResEdit, then choose "Create new resource" from the Resource menu. A list of alternatives appears. Choose or type 'Info'. A template appears.

Minimum version of the Dream application: 100, as we won't use the snazzy new features of version 1.5.
Full scenario name: The Castle of Terror
Screen depth: 1, as we have quite a few icons in black and white only.
Language code: 0, American English
Scenario version: 100
Author name: Write your name
Minimum number of characters: 1, as we won't allow creation of characters inside the castle.
Max number of characters: 8.
Suggested min level: Your own evaluation. I'd say 2.
Suggested max level: I'd say 4. Higher level characters could kill the vampire via spells.
New characters start at level: doesn't matter. Characters won't be created here.
Group starts at: let's say 5, 5. It depends on the map for place ID 1000, of course.
Scenario signature: Write here a four letter code which should uniquely identify your scenario. This is needed for compatibility. To guarantee maximum compatibility, contact the author to register this code.
% for wandering monster: the suggested value is 16%, or one in six.

We’re finished. Before closing the window, choose “Get resource Info” from the Resource menu and type a valid ID for the spell. 1000 is fine. Please also click on the “Purgeable” checkbutton.

**...and the rest**

We’re almost done. Now we have to create an encounter with the vampire, where our heroes get the chance to kill the bad guy. Or get killed, if they didn’t find the magic cross.

Here's a trick: the Nctr (encounter) format was changed for Dream 1.5. So we need to use the stratagem detailed in the description for the Encounter Resource to stay compatible with version 1.0:  change the name for the Nctr template in "Scenario resources.rez" to Nctr2, and then the name for Nctr1 to "Nctr". That's it, we're done.
Double check that you have the "Castle of Darkness" window as foremost in ResEdit, then choose "Create new resource" from the Resource menu. A list of alternatives appears. Choose or type Nctr. A template appears.
Must be done: Type 0
Item needed: type zero, as the encounter will take place even if the PCs didn't find the vampire. Note that there's another possibility here: you could create an encounter with a very weak vampire if the PCs found the cross, and strong vampire  if they didn't. In this variant you don't even need a spell bound to the cross!
Arena ID: Type a valid ID and jot it down, as you'll have to create an arena for the meeting with the vampire. Maybe 3000.
Now click on the asterisk row and type Command-K. You'll create the boxes where you'll type the vampire specs.
Monster ID: 1000, the ID we chose for Vlad.
Nr. in group: 1, of course.

Before closing the window, choose "Get resource Info" from the Resource menu and type a valid ID for the spell. 1000 is fine. Please also click on the "Purgeable" checkbutton.

Now we have to create the text, and maybe the pictures, for the scenario. That's very easy and we won't bother the reader with details.
We're done with ResEdit. That application can be quitted to return to ScenarioMaker.

**ScenarioMaker's final touch**
Full details on the ScenarioMaker application are given in the next chapter, but let's sketch its usage anyhow.
Using the File menu, open the "Castle of Terror" scenario. Choose "New place" from the Place menu. Then fill in the dialog box, keeping one eye on the scenario map which was drawn during the very first phases of scenario development.

Place name: Castle Yard
Place ID: 1000
Place kind: Standard
Choose freely a height and width. 8 and 8 are fine, though.
Leave the checkboxes unchecked and, if you created text or pictures, type their IDs in the boxes you see. Then press OK.

To draw the castle, as seen from outside, scroll in the leftmost window to the far right, until you see the castle icons (there are two halves). Click on the first icon, then click in the white window (somewhere in the topmost part, centered), to place it. Click on the other half, and then click in the white (construction) window near the first icon, to create the illusion of a full castle. Then click on a small house, and place it aside, somewhere, alone. It's the Gardener's Hut which we had planned.
Now for the green yard. Click on the all-green icon, and press F on the keyboard to let ScenarioMaker fill the construction window.

You should also insert some data in the locations to let ScenarioMaker know how the first place is bound to the others. The next chapter tells you how to do that.
For now, we are finished: close the construction window and answer “Yes” to ScenarioMaker’s suggestion that you should save it.

If you’ll also create the hut, the crypt, the castle interiors (just like we did the yard) and then the arenas for fighting, you’ll have completed your first scenario. Wasn’t that easy?
Oh, and don’t forget to use ScenarioMaker’s “Check scenario” feature before you distribute your creation. I bet you’ll forget something if you don’t. I know: I always do.

# ScenarioMaker features and functions

ScenarioMaker is an application created to make easy the building of place and arena resources.
An arena is the stage for a battle. In other words, it is what appears inside the battle window, when fighting begins.
A place is a part of the world where the characters move. The Valley of Dawn, an inn, a shop, one level of a dungeon, are examples of places.

### Using ScenarioMaker

First, you must ask ScenarioMaker to open an existing scenario file, or create a new one. Then you are given the possibility of creating new places and new resources, or to edit old ones.

### Grabbing icons

If you create a new scenario from scratch, you might wish to reuse icons you created for another scenario. To do so, simply choose “Grab icons from...” from ScenarioMaker’s File menu. Then select an existing scenario. ScenarioMaker will copy all location icons from the old scenario to the new one.
A few warnings apply:
1. Don’t select the newly created (and empty) scenario as the scenario to grab icons from. ScenarioMaker would be confused if the scenario to load from and the scenario where to copy were the same.
2. Icon grabbing works even for scenarios which already contain icons. ScenarioMaker will automatically renumber the new icons as it adds them.
3. Remember that icons are computer graphic pieces and, as such, are the property of their creator. You can’t simply go around and grab icons from other software.
You are granted the right to reuse the icons you’ll find inside “Spirit of Darkness”, if you wish. Such icons are the exclusive property of us, the Dream authors; you, as the legitimate owner of a copy of the Dream and ScenarioMaker applications, are given non-exclusive rights to use them in your own scenarios. If you do, please remember to list us as contributors to your scenario (both inside the splash screen (PICT ID 1128) and the written documentation, if any.

### Icons for mapping

To create new places and arenas you must have already prepared the icons you’ll use in laying out the maps. (See also the “Icons” paragraph in the following chapter).

The icons windoid, to your left, shows all of the icons available. If some icons don’t appear, or look funny, double check that you remembered to create an ICN# resource, with a valid mask filled black, and that the icons ID are numbered from 1000 upward, with no holes.
In the construction window, to your right, you can lay out the map of the arena or place. The process is simple: first click on the icon you wish to use inside the windoid, then click in the window to lay out the icon.

**Symbols in ScenarioMaker**

A few icons will, of course, be used to represent obstacles, like walls, water or mountain ranges. You can teach ScenarioMaker to distinguish between “impassable” icons and “accessible” icons. To do so, simply double-click the

“impassable” icons on the windoid. ScenarioMaker will mark as impassable every location where the icon is used.
To check that you identified a the “impassable” icons as such, select “Structure” from the appropriate menu, or press Command-T on the keyboard. In the windoid, all impassable icons will be marked with a red circle. In the construction window, impassable places will be marked dark grey, while passable places will be shown in light grey. Use this feature to check that you didn’t mistakenly divide a single place in two separate parts.
Other symbols used in the Structure view include:
• a hourglass-shaped yellow icon indicates locations which have an associated piece of text,;
• a red bullet indicates locations where an encounter is bound to take place;
• a red T shows traps;
• four question marks show the locations where riddles are placed.

**The fill mode**

Suppose that you wish to fill a place with a background: maybe you just created a building in the center of the window, and now simply wish to extend a grass lawn everywhere else.
To do so, first click on the icon with which you plan to fill in the holes. Then press ‘F’ on the keyboard. ScenarioMaker will fill every white space on the window with the icon you selected.

## Creating an arena

### Arena attributes

You should set attributes valid for the whole arena when you create the new arena; still, you are allowed to change your mind later. Choose “Area info…” from the Area menu to change the attributes you stipulated when you created the arena.
Arena name
This is the name that will appear as title of the window where fighting takes place. Keep it short, so that it will fit. This name will also be used inside ResEdit to identify the ‘Aren’ resource.
Arena ID
This is the resource ID of the ‘Aren’ resource where ScenarioMaker will save the arena map. remember that only IDs between 1000 and 9999 are considered valid.
**Warning**: if you change the ID of an arena already in existence, and set it to the ID of a different, already-existing arena, the latter will be destroyed and overwritten with the former.
Arena X and ArenaY
These are the horizontal and vertical sizes of the area, expressed in tiles. Since Dream is supposed to be playable even on LCs and Color Classic Macintoshes (with 12 inch screens), ScenarioMaker will restrain you to a maximum size which will be acceptable on such small screens.

The minimum size of an arena is six units by six units.

**Arena location attributes**

You can directly set the attributes for every location if you double-click on the location in the main window. You'll see a dialog box appear.

The meaning of the fields in the dialog box are as follows:

Is holy

Holy places (which are supposed to be extremely rare) damage evil supernatural beings by their very presence. If a demon, devil or other such creature were to step onto such a location, it would be damaged for each and every passing round.
Is unholy
Unholy places (which are supposed to be extremely rare) damage good supernatural beings, including paladins.
Is neither
Most places are neither holy nor unholy.
Slows
Locations marked “slow” will make every creature passing them slow down to one move for every two rounds. It is intended to be used for realising traps, like quicksand, narrow bridges, glued floors and such.
Anti-magic
Anti-magic locations (which are supposed to be the rarest of all places) make automatically fail every spell cast from there, or on there.
Is impassable
Impassable locations won’t let any creature walk on their surface. Note that flying creatures (monsters and player characters so empowered via a Fly spell) will be able to transit above impassable locations, thus making the attribute irrelevant.

**Creating a place**

**Place attributes**

You should set attributes valid for the whole place when you create the new place; still, you are allowed to change your mind later. Choose “Place info…” from the Place menu to change the attributes you stipulated when you created the place.
Place name
This is the name that will appear in the transcript windoid when the group enters this place. You are restricted to 30 characters maximum length. This name will also be used inside ResEdit to identify the ‘Plac resource.
Place ID
This is the resource ID of the ‘Plac resource where ScenarioMaker will put the place map. remember that only IDs between 1000 and 9999 are considered valid.
You should use IDs between 1000 and 1999 for those places where the player can go even if all of his/her characters are dead (or even non-existent). Use IDs of 2000 and above for the places where adventuring takes place: Dream will make sure that the player only enters those when his/her group is ready for adventuring.
**Warning**: if you change the ID of a place already in existence, and set it to the ID of a different, already-existing place, the latter will be destroyed and overwritten with the former.

X size and Y size
Spatial dimensions for the place. If the place is going to be a shop you should type 1 for both. Try to calculate in advance the dimensions for the place: you can resize it later, but if you do, garbage values will appear in the locations you added, and you'll have to put reasonable values in there by hand. It's bothersome.
Text on enter
Here you may type the resource ID of a piece of text (see Scenario resources/Text). If you do, that text will be shown whenever the players enter the place.
If Dream is running on a Macintosh where the MacInTalk Pro system extension is installed, the text will also be read aloud by the Mac. MacInTalk Pro can be run on any Macintosh capable to run Dream (LC or better), but is usually preinstalled only on AV and Power Macintoshes; it takes up to 4 MB of RAM to run — so, using

Dream with speech synthesis enabled requires a machine with at least 8 MB of RAM.
Pieces of text whose numeric ID ends with a “1” (e.g. 1001, 1011, 1021.. 9991) shall be spoken by a male voice. Pieces of text whose numeric ID ends with a “2” (e.g. 1002, 1012, 1022.. 9992) shall be spoken by a female voice.
I suggest using speech synthesis to deepen the effect that a scenario has on players: have characters met in the game speak to them, and avoid using the MacInTalk Pro voices simply read aloud the places descriptions (like I did in “Spirit of Darkness”!)
Text on exit
Here you may type the resource ID of a piece of text (see Scenario resources/Text). If you do, that text will be shown/spoken whenever the players enter the place.
PICT on enter
Here you may type the resource ID of a picture(see Scenario resources/Pictures). If you do, that picture will be shown whenever the players enter the place.
For map use PICT
If this button is checked, Dream will use a picture to represent the place on screen (the icon map won’t appear in this case): just like you see in the opening sequence for Back to Dawn Valley. Type in the box the resource ID for the picture (PICT resource). Please remember that this feature is only supported in versions 1.5 and above. If you insert a PICT map and run the scenario under Dream 1.0 you’ll see the icon map instead, and the picture map will only be revealed for a short while when the player exits the place.
If you opt to use a picture for map you MUST check “will be fully seen on entry”. You can see how your place will look like selecting “Picture” in the Place menu.
Bank, Brothel... Standard
Choose any one of those.
A bank is a place where characters can deposit money, withdraw it, and ask a new fighter to join the group.
A brothel is a place where characters can rest for one night, buy standard rations, and ask a new rogue to join.
An inn is a place where characters can talk with the innkeeper, buy standard and iron rations, and ask a ranger to join.
A mage tower is a place where the player can buy components for his or her spellcasters, and have a wizard join the group.
A shop is a place where the characters can buy and sell items. A shop must have an associated ‘Shop’ resource, detailing what items can be found there (See Scenario resources/Shops).
A shrine is a place where the player can have a paladin join the group. PLEASE keep shrines extremely rare. Finding a shrine and convincing a

paladin to join the group should be a difficult adventure unto itself.
A temple is a place where the group can be cured of its physical troubles (for a fee), and look for a cleric wishing to join.
In a special place you can place Extraordinary Characters for inclusion in the group. See the description for the Extraordinary Characters in the next chapter.
A standard place is neither of the above: just some place where the characters go adventuring, like a dungeon, a city or the wilderness.
**Warning**: Try to be reasonable. It doesn't make any sense to find a bank in the middle of a desert. If you wish to make something unavailable to the characters (e.g. they can buy every component but amber), use overriding (detailed in Tips and tricks/Overriding).

Must be saved when group exits
If this box is checked, Dream will save all changes to the place when the group exits the place. Thus, encounters will take place only once. If the box is unchecked, encounters programmed will take place every time the player character enter a location where an encounter is scheduled to take place.
Light is needed here
If the box is checked, Dream will require player characters to use a source of light, either magic (e.g. Light spell) or normal (e.g. a torch).
There are wandering monsters
If the box is checked, Dream will have random monster encounters happen to the PCs. Player characters won't be allowed to rest, relearn spells, or wait in a place where monsters roam.
**Warning**: if you check this box, you must create a Wandering monster resource where you specify what kind of monsters can be met here, and an arena for meetings. (See Scenario resources/Wandering monsters).
Will be fully seen on entry
If this button is checked, the whole place will be shown to the player as soon the group enters (there's no need to explore to see what's below the black spaces). You should check this for places where there are no wandering monsters.
This option is unsupported for Dream 1.0: under that version the player will always have to explore to see the map.
Outdoors, indoors, indoors underground
This will enable Dwarves, Elves and Gnomes to use their racial powers. Moreover, flying characters can fly over obstacles when outdoors, but aren't allowed to do so indoors.

**Place location attributes**

You can directly set the attributes for every location if you double-click on the location in the main window. You'll see a dialog box appear.
The meaning of the fields in the dialog box are as follows:
Is impassable
Impassable locations won't let any creature walk on their surface. Note that flying player characters so empowered via a Fly spell will be able to transit above impassable locations if your place is outdoors.
Text on enter
Here you may type the resource ID of a piece of text (see Scenario resources/Text). If you do, that text will be shown whenever the players enter the place.
If Dream is running on a Macintosh where the MacInTalk Pro system extension is installed, the text will also be read aloud by the Mac.
If this location is available only to player characters who own a special item, then this text will be shown only if they don't own the item. You can use this feature to create a door which is locked if they don't own the key, or similar.

Item needed to enter
Write here the resource ID of any item the player characters are required to possess to pass through this location.
Has trap/riddle/encounter/nothing
You can opt to have an encounter happen when the group steps onto a location, or a riddle asked, or a trap spring. See the paragraph relating to traps, encounters and riddles in the next chapter to see what these can do for you. Remember that only encounters are supported under version 1.0 of the Dream application
Player must search to encounter

If this box is checked and the “encounter” radio box is pressed, then the scheduled encounter will take place only if the player explicitly chooses “Search” from the Group menu. This can be used to hide treasures (like I did in “Spirit of Darkness”), simulate hidden monsters or buried treasures, and other.
Takes group…
Some locations are “doors” between places. (See Inside Dream/Map of the scenario). If you choose “Takes groups out of this place”, entering that location will have the characters return to the place from which they came. If you choose “Takes group into place ID...”, then entering the location will transport the player characters to some other place. You should normally avoid use of the “takes player out of here” option: the player knows that he or she has to step onto the white border surrounding the place to step outside, and transporting him or her out in this non-standard way can be confusing.

# Scenario resources

In this chapter we will detail each and every characteristic of the resources which comprise a game scenario. If the description of a feature is marked with a delta sign (like this: ∆) then that feature is only available under some version of the Dream application. For example, if you see:
*% for wandering monsters (∆ 1.5)*
...then you can set a percentage only under version 1.5 or following. The Compatibility notes for each paragraph will detail you more specific version characteristics for each of the released versions of the dream application.
A few of the features are marked “incompatible with previous versions”. If you choose to make use of one or more of these features, then you must remember to set a minimum version number inside your Info resource. For example, riddles were introduced only in version 1.5 of the Dream application. If you put a riddle in your scenario, then the scenario can only be played under version 1.5 or following. Dream 1.0 wouldn’t understand the instructions for a riddle, and would choke on them. If you write that the minimum required version for the dream application is “150” (that is, 1.5.0) in your Info resource, then you don’t risk anything. If a player with Dream 1.0 tries to open your scenario, Dream sees that a newer version is required, displays a dialog box saying that the player should find a more recent version and doesn’t load the scenario.

Everywhere inside ResEdit, “0” means “No” and “1” means “Yes”.

**Arenas**

**Resource signature**

‘Aren’: a special place where fighting takes place.

**What's it for?**
The arena is the place whose map appears inside the fight window wherever the player meets some monsters. There ought to be an arena for every place where wandering monsters may be found. Other arenas can be created for special encounters.

**Other needed resources**
Before you can create an arena, you should prepare the icons used to represent arena locations. Such icons must have consecutive numbers starting from 1000

**How to create one**
Use the ScenarioMaker application, as detailed in ScenarioMaker features and functions.

**Compatibility**
The "slows" and "anti-magic" attributes of an arena location is supported only under version 1.5 or following. Dream 1.0 allows these boxes to be checked, but won't support them.

**Tips and tricks**
You can easily create multiple similar arenas adapting the trick detailed under Places (see)

**Barkeeper tips**

**Resource signature**
'STR#': a collection of strings.

**What's it for?**
Inside STR#s you keep game tips which a barkeeper will give to its clients during game play.

**Other needed resources**
None

**How to create one**
Simply use ResEdit. When the group enters an inn, they can chat with the bartender. For each Inn you put in the scenario you must create a different STR# resource, which must contain the strings which the bartender will speak. For Dream 1.0, create three strings inside the STR#. The bartender will immediately give away the first. To hear the second, the player shall have to buy something, and to hear the third he will have to buy quite a bit of merchandise.

**Tips and tricks**
Keep the most valuable tip for the third, and most hard to obtain, string!

## Designer orders (Δ 1.5)

**Resource signature**
'AdDB': a list of resources to be ADded to the Data Base.

**What's it for?**
The best feature of the Dream gaming system is arguably its modularity. Under Dream, a player can use many scenarios from different Designers and still integrate them into a unique gaming experience.
Some efforts on the part of the Designers are required to make this possible. You must remember that the resources which constitute a scenario become unavailable to the gaming system when the scenario adventure is finished and the player switches to another adventure. So, Bad Things may happen if a player buys an axe ('Obj ' ID 5000) for his character, Grrr the Fighter, and then switches to another scenario where the item ID 5000 is a frozen haddock.
The Sword Dream application does most of the work to keep the axe from transforming into a haddock. Even version 1.0 offers some protection from this. Still, a fully functional solution to the problem is found only under version 1.5 and following, with the Conflict Resolver (see its description in the Dream Database chapter).
You simply have to list all of the objects and spells which you make available in your scenario, and order them transported inside the Dream Database, together with their icons. This way, objects and spells stay available forever.

**Other needed resources**
The AdDB resource should be the very last resource you put into your scenario. When the design is finished you can look into your work and identify the entities which might be kept (for items) or learned (for spells) by the player characters.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the ‘Scenario Resources.rez’ file. Fill in every field of the template, listing the resource signature and ID for every resource which you wish copied inside the Dream Database.

**Compatibility notes**

The AdDB resource is used only by version 1.5 and following of the Dream application. You might state that your scenario is compatible with Dream version 1.0, if you wish, as long as only items will be transferred from your scenario to the next: under Dream 1.0 the item definition will be kept, but the icon will be lost. You'll probably want to limit usage of your scenario to Sword Dream 1.5 users.

**Tips and tricks**

An ingenious scenario Designer might use the AdDB resource to keep his or her scenarios small. If you create a series of, let's say, three interlocking scenarios, you might include the relevant monsters, places, pictures and sounds only in the first scenario, and order them copied to the Dream Database: thus they would be available for use even from the second and third scenario which don't contain them.

## Encounters

**Resource signature**

'Nctr': description for a planned encounter.

**What's it for?**

Encounter is a very general term, which might cover a fight with a group of monsters, the discovering of a hidden treasure, or the location and reading of an ancient rune. Or all of the above.

Encounters always take place at some fixed location. They conform to one of three behaviours:

1. Encounters that always take place, whenever the player group passes through the encounter location; e.g., a guard post at the gates of a huge city.
2. Encounters that take place only once. E.g., finding and fighting a dragon and its hoard. (once they kill the dragon and take its treasure, if any, no more encounters happen inside the dragon's cove).
3. Encounters that take place only if the player characters actively search for something (the player must choose Search from the Group menu for the encounter to happen).

Encounters can also be bound to possession of some item (that is, the encounter only happens if one of the player characters owns some particular item).

**Other needed resources**

Before you create an encounter, you ought to create separate resources describing all monster types and objects for the encounter. You also should write the piece of text describing the encounter, and create an arena where the encounter happens, using ScenarioMaker.

If one or more of the monsters found during the encounter must be precisely controlled, create its Monster Decision List (MDeL) before the encounter.

**Special note**

The definition for an encounter as found in Sword Dream version 1.0 was

found to be somewhat limited (for example, it didn't allow a Designer to precisely place the monsters inside the battle arena). Sword Dream 1.5 introduces a new, enhanced encounter type (the EE, or Extended Encounter, for short). We encourage you to use this newer and more flexible form. Should you need to create a scenario compatible with version 1.0 of the Dream application, though, do the following:

1. Open the Scenario Resources.rez file
2. Open the TMPL icon
3. Find and select the "Nctr" item

4. Press Command-I. A window appears: Change the name of the item to “Nctr2”
5. Find and select the “Nctr1” item
6. Press Command-I. A window appears: Change the name of the item to “Nctr”

You can now create encounter using the old version 1.0 format. To return to the newer format apply the steps above in reverse.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the ‘Scenario Resources.rez’ file. Fill in every field of the template, as detailed here:

Uses new specs (must be 1) (Δ 1.5)

Click on “1”. This informs the Dream engine that this encounter follows the Extended Encounter framework.

Done (must be zero)

Click on “0”. The Dream engine will set this to 1 when the encounter happens, to memorize that it already happened.

Item needed

If you wish to create an encounter that only happens if one of the player characters owns some item, then type here that item’s ID. Otherwise, type 0 (zero).

Arena ID

Type here the ID of the arena where the player characters will fight the monsters in the encounter (if any). If there are no monsters here, type 0 (zero).

Monster count

This field will initially show a -1 value. For each monster type you wish to put in the encounter, click once on the row of asterisks, then press Command-K. For example, if you plan to include five skeletons, you’ll add a single group of monsters, and to do so you’ll only press once command-K. For five kobolds, three goblins and one orc you have to add three monster groups; to do so press Command-K thrice. Then fill in the fields that appear below, once per monster group.

Monster ID

Type the numeric ID for the kind of monster. (The “number appearing” field inside the monster description is not considered for an encounter. It only has significance for wandering monsters).

PosX and PosY (Δ 1.5)

Coordinates for positioning the monster inside the battle arena. The upper-leftmost position is 1,1, and the numbers grow when moving to the right and bottom.

Tactics (Δ 1.5)

Type a letter in this box to describe the monster’s behaviour (see Inside a Dream/Monsters/Monster behaviour).

Nr. in group
Type the number of monsters of that kind you wish to place in this encounter. E.g.: for three kobolds, type 3.
Golden Eagles
Type here the number of golden Eagles that the characters will receive at the end of the encounter. The amount is evenly divided between group members.
Treasure count
This field will initially show a -1 value. For each item you wish to put in the encounter, click once on the row of asterisks, then press Command-K. Then fill in the field that appears below
Item ID

Type the numeric ID for each of the items that the player characters will found at the end of the encounter.
Please note that Sword Dream versions 1.0 through 1.6 only support a maximum of 49 items for a treasure. If you exceed this limitation the application might crash.
Encounter text
This text will appear at the beginning of the encounter.

**Notes**

To create a type-1 encounter (as detailed above, in the “what’s it for” paragraph), mind that the “Must be saved when players exit” button is unchecked in the place where the encounter takes place.
To create a type-2 encounter, the “Must be saved when players exit” button should be checked.
To create a type-3 encounter, both the “Must be saved when players exit” and the “Only when player searches” button should be checked.

**Tips and tricks**

You can create an encounter that takes place at either one of two locations by creating a type-2 encounter, and then referring to it at both locations. So, you could have guards at both entries of a tower, but when the guards are defeated, both guard posts are left unattended.

## Extraordinary Characters (Δ 1.5)

**Resource signature**

‘Char’: a non-standard character that the group meets.

**What’s it for?**

An Extraordinary Character (XC for short) is a monster or non-standard humanoid being which the group meets in peace. The XC is allowed to join the group, and is always found in a Special Place (see Special places in this chapter).

**Other needed resources**

You must ready two different icons for each Extraordinary Character. The first will be used to show the EC’s face in the Roster, while the second will depict the creature full-figure and will be used during fights.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the ‘Scenario Resources.rez’ file. Fill in every field of the template. Most of the fields have the same name and use as in a monster, or are self-explanatory. The others are detailed here:
Is poisoned
If you wish to create a poisoned Extraordinary Character, type here the number of hit points the XC will lose every minute because of the poison. Otherwise write 0.
Class

Write a class code, from Inside a dream/Characters.
Race
Write a race code, from Inside a dream/Characters.
XC destination
If you wish to have the XC leave the group as soon as a certain place is reached, type here the place's numeric ID. If you wish to leave the XC indefinitely with the group type 10000.
XC text on exit
When the XC reaches its destination and disappears this text will be shown.

XC encounter on exit
When the XC reaches its destination and disappears this encounter will be executed.

**Notes**

Under Dream 1.5 the XC won't take part in fights. It has no need of equipment (in fact, its equipment is invisible to the player) and needs no food. An XC can cast spells, if you give it some, but won't do so during fights.
Part of these limitations will be lifted under subsequent versions of the Dream gaming system.

**Compatibility**

Extraordinary Characters can only be used under version 1.5 and following of the Dream software.

## Icons

**Resource signatures**

- 'icl8': icons for game play in 256 colors
- 'icl4': icons for game play in 16 colors
- 'ICN#': icons for game play in black and white

**What's it for?**

Most everything in Dream is represented through an icon: the characters in the player rosters are icons; the objects inside a character's sheet are shown as icons; and the maps are tiled icons.

**Other needed resources**

No other resources are pre-required.

**How to create one**

You create icons inside ResEdit. You also might use any graphic application you like, and then paste your work using ResEdit.
When you are satisfied with your work, give it a meaningful name (command-I in ResEdit, then type a name). Jot down the resource ID you gave to this icon somewhere, as you'll be required to type it to reference your artwork.

**Notes**

The color version of the icon is optional: Dream can work without them . Black and white icons, on the other hand, are mandatory. So, for every icon you create, there MUST exist a black-and-white version, in the form of an ICN# resource. Furthermore, all ICN#s MUST have a valid mask, and the mask MUST be a solid black (the icons for the monsters are the only valid exception to this rule: you can have white holes in the mask to show the background through).
Of course, all versions of the same icon must have the same ID. That is, if you create a black and white icon, ICN# ID 2500, for a red dragon, the color dragon should be situated inside icl4 and icl8 resources with ID 2500.
The Dream application code shall accept icons with any id (in the range 1000-9999). Still, if you wish to employ the ScenarioMaker application to create the

maps, you must submit to a quirk which ScenarioMaker enforces: all icons used to “paint” the maps must have consecutive IDs, starting with ID 1000. Suggested ID values for the icons are 1000-1999 for locations, 2000-2999 for monsters and 3000-3999 for objects.
Icon usage should be planned with care. If you create places and locations very similar to those found inside “Spirit of Darkness”, you should use the very same icons, or icons very similar to those. If you do, users will feel at home when they

enter your scenario, because they will recognise familiar shops, temples, mage towers and the like.
On the other hand, don't reuse icons for totally (or even partially) different places: this will confound users. So, for a temple of evil, come up with a totally different icon.

**Tips and tricks**

Many designers feel frustrated with the size of the icons. Still, you may create complex artwork by tiling different icons. Look at the bushes in "Spirit of Darkness" to see a simple example: many different icons were created, each with a part of the bush. Then, tiling the icons in the place map, the big, complex bushes were created. You can do the same with mountain ranges, floors, palaces...

## Items

**Resource signature:**

'Obj ': an item to be bought, found, used, dropped or sold.

**What's it for?**

Objects can be equipment (rations, armor, torches, weaponry and so on); treasure (like a gem); magic enhancements (like a ring of protection), and more.

**Other needed resources**

Before you create an object you should draw the icon which will represent that item. When you create the icon write down its numeric identifier: you'll need it when you create the item.
If the item is a scroll (see: Inside Dream/Objects/Item categories/Scrolls) for a new spell then you ought to create the spell before you create the scroll.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the 'Scenario Resources.rez' file. Then create the resource. Fill in every field of the template, as detailed here:
Icon: write here the ID of the icon you created to picture the item
Magic, weapon… scroll: See Inside Dream/Objects/Item categories.
Bonus/code: For armor, write here how much the armor will increase the character's AC. For scrolls, write here the ID of the spell. For generic magic items, write here the ID of the spell that the item will cast when used.
Number of charges: For light sources, the number of minutes the item will burn for. For food, this is the number of days the food will sustain the character for. For magic items, this is the number of uses before the item is discharged.
Weight: Weight in pounds of the item.
Price: Value in GE of the item. Zero is perfectly OK. The maximum allowed is 32.000.
Wear on finger, shoulder… sack: See Inside Dream/Objects/Weight and other things

Wizard… Fighter can use: See Inside Dream/Objects/Character classes and items.
Public name: See Inside Dream/Objects/Weight and other things
Private name: See Inside Dream/Objects/Weight and other things
Weapon base damage: The magic bonus for magic weapons. Zero for all other items. See Inside Dream/Objects/Item categories.
Weapon num dice and Weapon dice size: Together these numbers help fix the damage a weapon will  inflict. See Inside Dream/Objects/Item categories and Inside Dream/Fighting/Damage.

**Self-propelled throwing weapons**

To create a self-propelled weapon (like a dart to be thrown by hand) for use in a game scenario, do the following.

1. Create the weapon object as detailed in the previous paragraphs.
2. Set the “Throwing weapon” identification bit
3. Write “1” into the “Number of charges” box.

This weapons can be of the magic kind. You can set a Bonus in the appropriate box.

Keep in mind that only Rogues should be allowed to use throwing weapons. So, double-check that the Rogues allow bit is set, and that other bits are reset.

**Throwing instruments**

To create a throwing instrument (like a bow) for use in a game scenario, do the following.

1. Create the weapon object as detailed in the previous paragraphs.
2. Set the “Throwing weapon” identification bit
3. Write “0” into the “Number of charges” box.

Keep in mind that only Rogues should be allowed to use throwing weapons. So, double-check that the Rogues allow bit is set, and that other bits are reset.

Dream doesn’t allow creating magic throwing instruments. That is, no magic bows in here. Why? Because there must be an algorithm by which the game system can associate throwing instrument and appropriate ammunition. It would be ridiculous to throw a stone with a bow or an arrow with a sling. So, ammunitions “point” to the right kind of throwing instrument. But they couldn’t do so if there existed multiple throwing instrument per ammunition (like a bow, a bow + 1, a bow + 2…)

**Ammunition**

To create ammunition for a throwing instrument (like arrows for a bow), do the following.

1. Create the ammunition object as detailed in the previous paragraphs.
2. Set the “Ammunition” identification bit
3. Write the appropriate number into the “Number of charges” box. For example, if arrows are sold by the dozen, write “12” in the “Num charges” box.
4. Write the ID of the correct throwing instrument in the “Code” box.

Keep in mind that only Rogues should be allowed to use throwing weapons. So, double-check that the Rogues allow bit is set, and that other bits are reset.

**Monsters**

**Resource signature:**

‘Mstr’: a monster kind (race).

**What’s it for?**

This resource describes the characteristics, special attack and defence forms, and other attributes of a monster race.

**Other needed resources**

Before you create a monster you should draw the icon which will represent it. When you create the icon write down its numeric identifier: you'll need it when you create the monster.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the 'Scenario Resources.rez' file. Then create the resource. Fill in every field of the template, as detailed here:

Number appearing: Maximum number of monsters in a group. See Inside Dream/Monsters/Wandering monsters
Alignment: The alignment of the monster. See Inside Dream/Monsters/Monster alignment
8 sided hit dice: This is the “level” of the monster. To calculate the amount of hit points the monster has, Dream will throw this many times an eight-sided die.
A kobold, goblin or other similar low-level monster will have one hit die. An orc would have 2, and ogre 3. A wild panther might have 5. A demon would have 8 or 9 hit dice. A huge, ancient dragon might reach 16 hit dice.
THAC0: This number represents how good the monster is when fighting. Consider an imaginary foe (a PC) for the monster whose armor class is zero. If you wish that the monster hit the character one time out of 20, write 20. For two times out of 20, write 19, and so on.
Very low level monsters should have a THAC0 of 20 or 19. A wild panther would have a THAC0 of 15. A huge, ancient dragon might have a THAC0 of 2.
Attacks per two rounds: The number of attacks/mover the monster will do every two minutes. It’s normally 2. A very skilled human fighter might have 3. A panther might have 4 (two claws). A six-handed demon might have 12.
Dice of damage and Damage die size: Together these numbers help fix the damage a monster will inflict. See Inside Dream/Fighting/Damage.
AC: The armor class of the monster. A slow monster would have 10. A slow monster with leathery skin would have 8. A quick and/or small monster would have 7. A crocodile or other beast with a thick, scaled hide would have 5. An armored human foe would have 2. A dragon (which is a highly magic creature, whose thick muscles would block enemy thrusts and whose hide is covered with metallic scales) might have an AC between 2 (young, small animal) and -5 (ancient dragon). A supernatural foe of very high power, extremely agile, maybe smaller than normal man, might even reach AC -10.
XP: The number of experience points that the character group will be awarded for killing one monster of this kind. This amount will be evenly divided between all characters.
A few examples: 10 XP for a weakly kobold, but at least 20 if the kobold has a bow. 30 XP for a thickly-cuirassed giant ant or an orc; 100 for an ogre. 350 for a panther. 500 for an infant dragon, and up to 15000 for a huge, ancient, fire-spewing one. 1000 for a dinosaur; 5000 for a tyrannosaurus rex.
Can see invisible: Not yet implemented
Can be invisible: Not yet implemented.
Can fly: If “1”, the monster can pass above obstacles inside the arenas.
Regenerates: If “1”, the monster recuperates one lost hit point per minute.
Intelligence: See the table in Inside Dream/Monsters/Monster intelligence
Morale: The chance for fleeing, per minute, expressed as a percentage. This amount is lowered by 10 for every friend of the monster that is killed during the

fight, and by 3 for every hit point that the monster loses. Shouldn’t be lower than 100, except for special cases. 120 should be standard for organised monsters, and 140 should be normal for well-organised, tribal, order-aligned monsters. For berserker-kind monsters that will fight to the death, write -1.
Name: The name that will appear in the transcript windoid when the monster moves.
Icon: write here the ID of the icon you created to picture the monster
SPECIAL ATTACKS (Can mind attack... Brings illness): See Inside Dream/Fighting/Attack forms. The following are implemented for Dream 1.0:

Heal
Poison
Drain level
Fights... Fights as +5
Shoots missiles
Death ray
Curse
Illness
A normal monster should have all of there set to 0, with the only exception of "Fights". Archers should have all set to zero but "Shoots missile".
Spare: Reserved for the future. Write 0.
Attack selector code: This is a number which further specifies the attack abilities of a monster. For Dream 1.0, this is only used to specify the maximum throwing range of an archer.
SPECIAL DEFENSES: (Mind attack immune... Immune to illness). These specify the immunities the monster has to various attack types. All are implemented.
Undead monsters should be immune to Mind attack, poison, drain, death magic, illness. Most monsters (all but supernatural monsters) should be immune to curses. Use logic here: a fire-spewing dragon should be immune to fire. Some poisoning predators are immune to poisons (e.g. giant spiders), but others aren't (e.g. scorpions).
Spare: Reserved for the future. Write 0.
Special defences selector: This is a number which further specifies the special defences of a monster. For Dream 1.0, this is reserved.

**Notes**

To create wandering monsters see Wandering monsters list..

**Tips and tricks**

You can create a monster that tries to flee immediately by setting its morale to zero. Under Dream 1.5 you can do the same using the monster behaviour specification in the Encounter resource. The first method requires a monster concocted *ad hoc*, but works under every version; the second method lets you reuse an already defined monster, and save disk space, but can only used within an encounter.

## Monster decisions (∆ 1.5)

**Resource signature**

'MDeL': a list of moves for a monster to execute

**What's it for?**

A Designer can opt to control with maximum exactness every monster move. To do so, you must create a list of orders to the monster. The monster will execute your orders, one per melee round, until it is killed or it flees the arena. If the list is executed to the end, then the Dream application takes control of the monster decisions.

**Other needed resources**

You should create the Mstr resource with the monster definition before you start working on the Monster Decision List.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the ‘Scenario Resources.rez’ file. Then create the resource.

For each move you with to insert, click once on the row of asterisks and press Command-K. The fill in the boxes which appear.

Spare

This space is reserved for future enhancements. Type a one (1) in it.

Action

Type here a character representing the action the monster is asked to perform. See Inside a dream/Monsters/Monster behaviour.

Direct/Indirect

Type here the numeric code for the direct and indirect object of the move. See the table in Inside a dream/Monsters/Monster behaviour.

**Compatibility**

MDeLs are supported by version 1.1 and greater of the Dream application. If you choose to insert one or more you should make certain that your Info resource asks for version 1.1 of the Dream engine. If you don't, a player using Dream 1.0 will be able to open the scenario, but the software will then declare the scenario damaged or faulty when the related monster appears, and quit on the user.

## Pictures

**Resource signature**

PICT

**What's it for?**

You can have Dream display, inside the picture windoid, a color picture of the place the characters have just entered or exited. This can add drama to a scenario, and give the player a bird's eye view at a place he/she just entered.

**Other needed resources**

No other resources are pre-required.

**How to create one**

Use your favourite graphic package to create the picture, or simply use a digitised photo.

When you have the picture ready, copy it, switch to ResEdit, and paste it. Give it a name and a valid ID with ResEdit's Info menu item. Jot down the resource ID you gave to this icon somewhere, as you'll be required to type it to reference your artwork.

**Notes**

Bit depth is not an issue: Dream will try to display the picture at the best possible on the particular Macintosh on which it will run.

Size is not an issue, also. Don't exaggerate or your scenario will become too big to be manageable.

**Important**: remember that you MUST create a picture with ID 1128 with the "about this scenario" information. All game scenarios ought to have one. This picture should be a 16-greys PICT, no wider than 300 pixels. You can also create a piece of text (see TEXT), ID 1128, to go with that picture.

**Places**

**Resource signature**

‘Plac’: the scene where player characters act.

**What’s it for?**

Every place resource keeps information about a single game site. A place may be a castle or a shop, a tent in the desert or a shrine lost in the mountains.

**Other needed resources**

The places should be the very last entities you create inside a scenario, as they reference nearly everything else: icons, encounters, text, pictures, arenas, wandering monsters, and even other places.

**How to create one**

Refer to Inside Dream/Map of the scenario. First plan a map, as detailed there. Jot it down a piece of paper. Then assign a number to every single place: start with 1000 for the topmost place and work downwards from there. Then use the ScenarioMaker application to draw the map of every single place and establish the relations between places (see ScenarioMaker features and functions)

**Notes**

Double-check the resource IDs of the entities you refer to inside places. If you decide that a piece of text, ID 1500, should appear when the players enter a place, and then forget to create it, then Dream will recognise the error and interrupt game play with an error message saying that the scenario file is corrupt.

**Tips and tricks**

If you decide to create twin places (like a temple of good and a temple of evil very similar), you might create the first, then save it, then reload it, change its numeric ID and name using the “Place Info” item inside ScenarioMaker’s Place menu, do the necessary changes and save it again. This will save time.

If you need to bar any one location in a place from the player characters you can make that location impassable. If you do, the player has no explanation why the location is inaccessible. For those occasions when you wish a message displayed whenever the player tries to walk somewhere (e.g. “You can’t enter this door, you don’t own the key”), do the following: open the Location Info dialog box by double-clicking the location in ScenarioMaker. Type 10000 in the “item needed to enter” box. Since items have IDs between 1000 and 9999, there’s no chance that a character will ever own an item with ID 10000. Then create a TEXT piece with the message, and put its id in the “Text displayed on entry” box, and you’re done.

Note that ScenarioMaker knows of this trick, and won’t question usage of item id 10000 during a scenario check.

## Riddles (∆ 1.5)

**Resource signature**

‘Ridl’: a question or a riddle which the player has to give an answer to.

**What’s it for?**

The Sword Dream application, from version 1.5 onward, lets you ask a question to the player. You can have a dialog box appear, and present a question of your own choice: this can be a riddle, the request for a password

or anything else.
The player can then type an answer. The Dream engine will evaluate the answer and execute one of many different Encounters which you have devised in reply.
Under version 1.5 you can create as many as three different outcomes: the first and the second will depend on the player typing a specific answer, while the third will be used for all other answers.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the ‘Scenario Resources.rez’ file. Then create the resource. Fill in every field of the template.

**Compatibility**

Riddles are supported by version 1.3 and greater of the Dream application. If you choose to insert one or more you should make certain that your Info resource asks for version 1.3 of the Dream engine. If you don’t, a player using Dream 1.0 will be able to open the scenario, but the software will then declare the scenario damaged or faulty and quit on the user.

**Notes**

When the Sword Dream evaluates the player’s answer, it strips uppercase letters and diacriticals. Thus, “Hello”, “hello”, “HELLO” and “hellò” are considered equal.

**Tips and tricks**

If you have no need for a second alternative (that is, your question has only one acceptable answer, and everything else is just plain wrong) don’t write anything in the “Answer 2” field of the template (leave it blank).

## Scenario information

**Resource signature**

A single resource of type ‘Info’, ID 1000.

**What’s it for?**

Inside the Information resource you write information abut your scenario. This information will be used by the Dream application to fully support your scenario.

**Other needed resources**

No other resources are pre-required.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the ‘Scenario Resources.rez’ file. Then create the resource. Fill in every field of the template, as detailed here:

Minimum version of the Dream application

This is a number which represents the minimum version number of the Dream application which your scenario can be played with. To be compatible with Dream v1.0ß1 and following, write 90. To be compatible with v1.0 final, write 100. To be compatible with v1.0.2 and following, write 102. To be compatible with v1.5 and following, write 150.

Full scenario name

This is the name that will appear in the main Dream window and the transcripts. Remember that a user can easily change the name of the scenario file, but he or she won’t be able to change this, the real name. Keep it short, no more than 30 characters, or it won’t fit in the space.

Screen depth suggested

This is a number representing the Monitor setup that the Dream application, on your behalf, will suggest to the player. The number should be either 1, 4 or 8 (for black and white, 16-colors, and 256 colors respectively). The values 2, 16 and 32 are supported but don't make any sense with the current version of the Dream application. Other values are invalid and shouldn't be used.
If you created only 'ICN#' resources, use 1. If you also created a full-color version of the artwork inside 'icl8' resources, then type 8. If you went full circle and created also the 16-color version of the icons, inside 'icl4' resources, then type 4.

Language code

This code is needed to let Dream know what human language you employed in writing the scenario. This code is then passed to PlainTalk, so that your text is pronounced using the correct phonemes. The numeric code should be the same as in the ‘vers’ resource (see). The most common values are as follows:

| | |
|---|---|
| 0 | American English |
| 1 | French |
| 2 | British English |
| 3 | German |
| 4 | Italian |

Scenario version

This is the version number for your scenario. It should be version 1.0 (typed as 100) for the first release, then be incremented for subsequent releases. Use the same conventions detailed for the Dream application version, above.

Author name

Your name goes here. Dream will use it in various dialog boxes. Use the <given name, middle initial, family name> form.

Minimum number of characters

The minimum number of characters a player must have readied before entering your scenario. If you allow creating characters inside the scenario, the number may be 0.

Maximum number of characters

The maximum number of characters which the player should keep for game play in the scenario. This is usually found after game testing. Keep in mind that the Dream application won’t enforce the limit, but simply suggest to the player that he or she doesn’t use/create too may characters.

Suggested minimum level

Type here the minimum experience level the characters should possess. If your scenario is brimming with demon princes it doesn’t make sense to allow first level characters in.

Suggested maximum level

Type here the maximum experience level the characters should possess. If your scenario has a couple of kobolds with wood sticks as the greatest danger it doesn’t make sense to allow twentieth level characters in.

New characters start at level (∆ 1.5)

The experience level of characters met and joined in the group during this scenario. Should normally be 1, but can be higher if the scenario is particularly difficult.

**Warning**: Dream v1.0 will generate first level characters no matter what you write here.

Scenario signature

You must type here a four letter “signature” for your scenario. This will be used by the Conflict Resolver to uniquely identify your work. A few examples: the signature for Spirit of Darkness is SpDk. The signature for Back to Dawn Valley is BkDV. You can use any mix of uppercase and lowercase letters, digits

and punctuation symbols. Remember to devise a different signature for each one of your scenarios: to be perfectly certain that your scenario will be compatible with everybody else's, register the signature with the Dream author.

% for Wandering monsters (Δ 1.5)

This is the percentage for appearance by wandering monsters. It is checked once every six moves by the player when his or her characters are walking through any monster-infested place.

Under Dream 1.0, this Designer-defined number is not used, and a fixed percentage is used instead: Dream 1.0 always uses 16, that is 1 in 6. Thus, under version 1.0 encounters happen, on the average, every 36 user moves.

**Notes**

The information resource is **required**.

**Tips and tricks**

If you wish to prominently display your name, address and other information with your scenario, use the PICT and TEXT resources, ID 1128. Those will be shown whenever the user chooses “About this Scenario” from the Apple menu.

## ScenarioMaker information

**Resource signature**

‘SMkr’: Information created and maintained by the ScenarioMaker application.

**What’s it for?**

ScenarioMaker keeps track of which icons are used to represent impassable locations. When you exit ScenarioMaker such information is saved inside the scenario for future use.

**Other needed resources**

The SMkr information is related to icons.

**How to create one**

You don’t. ScenarioMaker does this for you.

**Tips and tricks**

You’ll probably want to leave this resource alone. Still, when you are completely finished with a scenario, you might want to remove the SMkr resource from it (using ResEdit). This will make the scenario file smaller by a half kB.

## Shop

**Resource signature**

‘Shop’: the list of items for sale at some place.

**What’s it for?**

A shop is a place where the characters can buy equipment, sell treasure, re-equip themselves.

**Other needed resources**

Before you can successfully create a shop you must create all of the items sold inside the shop (see Objects) and a Place for the shop to be in (see Places).

**How to create one**

First create a place, using ScenarioMaker. Any size will do, as the player won’t see inside the shop. 1x1 is fine, so you won’t waste disk and memory space. In the “Place info” dialog box of ScenarioMaker, select the “shop” radio button. Write a unique number, of your choice, between 1000 and 9999, in the “Time for a step” box. Jot it down: that number will identify your shop.

Then open the scenario using ResEdit and the “ScenarioMaker.rez” file. Create a ‘Shop’ resource, with the same ID that you wrote before in the “Time for a step” box. Put the IDs of the items for sale inside the ‘Shop’ resource. The order you use in specifying the items is the same order the items will appear inside the shop.

For example, if “At the Old Dwarf’s” ('Plac' ID 1066) people can find maces ('Obj ' ID 1176) and swords ('Obj ' ID 1255), then create a 'Shop' resource, ID 1066 (the same as the 'Plac') and put inside it the numbers 1176 and 1255.

**Notes**

You are limited to a maximum of 50 items in a shop. If you put more items than fit the available space, the scrollbar at the right will activate.

**The standard shop (∆ 1.5)**

Under version 1.3 and greater of the Dream application, there’s a standard shop defined for your use inside the Dream Database. If you wish to offer a shop which lists standard items, like the shops which are found in Spirit of Darkness or Back to dawn valley, simply create a “shop” kind of place in ScenarioMaker and write 10000 in the “Time for a step” box. That’s it.

**Sounds**

**Resource signature**

‘snd ‘: Digitised sound.

**What’s it for?**

If you wish to create a genuine roaring monster put a ‘snd ‘ with the same ID as the monster inside the game scenario.

**Other resource needed**

‘Mstr’ resource with the same numeric ID as the sound.

**How to create one**

Sounds can be created with the microphone that most likely came with your Mac, and the Sound control panel, then pasted inside the scenario using ResEdit.

**Special places (∆ 1.5)**

**Resource signature**

‘Menu’: menu items for non-standard places (but see below)

**What’s it for?**

Under version 1.0 of the Dream application, a scenario Designer could create places of many different kinds. First came temples, inns, shops and other standardized places; then there were generic, non-descript places useful for exploration and adventuring.

Under version 1.5 and above of the dream engine, a Scenario Designer can devise places with very special characteristics (like a prison, or a tomb). Inside a special place you can have the player characters meet other characters (the so-called Extraordinary Characters) and have those join the group.

**Other needed resources**

To create a Special place, you must prepare four different resources.

1. Create the map of the place using ScenarioMaker, as always. Remember to select the “Special” radiobutton in the Place Info dialog box.
2. Create the non-standard menu for the place, as detailed in the following

paragraph.
3. Write the text that will appear when the first menu item is selected by the player, as detailed below.
4 (and optional). Prepare an Extraordinary Character, following the rules exposed earlier in this chapter (see Resources/Extraordinary characters).
These four resources ought to have the same numeric identifier.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the 'Scenario Resources.rez' file. Then create the resource. Fill in every field of the template.

First of all, type a name for the place (something like Prison, or Prairie, or Hole): it will be used as a title for the menu.

Then type the text for the first menu item. When (and if) the player selects this item, the TEXT with the same id as the Menu resource will be shown.

If you wish to have an Extraordinary character associated with the place then write a third line (a second menu item). When the player chooses this last menu item, the Extraordinary character joins the group.

**Notes**

Handling of Special places and Extraordinary characters will become more sophisticated under later versions of the Dream software.

**Compatibility**

Special places are supported by version 1.3 and greater of the Dream application. If you choose to insert one or more you should make certain that your Info resource asks for version 1.3 of the Dream engine. If you don't, a player using Dream 1.0 will be able to open the scenario, but the software will then declare the scenario damaged or faulty as soon as the player enters the Special place, and quit on the user.

**Spells**

**Resource signature**

'Spel': description of a spell

**What's it for?**

Spells can be created for magic users and clerics to learn and use. Or, special spells might be concocted for making up a truly special magic item.

**Other needed resources**

You should draw an icon to represent the spell before you create the spell itself. The icon is not needed for spells which will only be cast via a magic item.

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the 'Scenario Resources.rez' file. Fill in every field of the template, as detailed here:

Level: the level of the spell. See Inside Dream/Spells/Spell level and IDs, or use zero for a spell to be cast by a magic item.

Kind code: The type of magic the spell works. See Inside Dream/Spells/Other spell attributes and Inside Dream/Fighting/Attack forms.

Reserved: Write a zero here.

Wizard or cleric: Click on zero for a wizard spell, one for a cleric spell. It doesn't make any difference if the spell is to be used for items only.

ST for none and ST for half: See Inside Dream/Spells/Saving Throws. Only one of the two may be set to “1”, meaning “yes” (or both might be zero).
Dmg is per level: If set to true (“1”), then the spell is meant to damage the enemy, and the amount of damage depends on the experience level of the spell caster; in this case, write the amount of damage per level in the damage die size and Damage die quantity boxes, below.
Can cast in fight: If the spell makes sense during a fight, click “1”.
Can cast in peace: If the spell makes sense during everyday adventuring, click “1”.

Area code: This indicated what target the spell has. See Inside Dream/Spells/Area of effect. If the spell doesn't have any obvious area of effect (e.g. Light), write a zero.
Distance in units: Useful only for those spells which can be used during a fight, this is the maximum distance in squares from the spellcaster to the target. For spells whose area code is zero (the target is the caster), write zero. For spells where the caster has to touch the target (e.g. Cure graze or Raise dead), use 1.
Area size: This is only meaningful for spells whose area code is greater than 4. For circular area spells, this is the radius of the spell area. For square area spells, this is the length of the side of the square. For straight path spells, this is the length of the path.
Damage die size and Damage die quantity: Together these numbers help fix the damage a spell will inflict. See Inside Dream/Fighting/Damage.
Duration base: For spells whose length is meaningful, this is the number of minutes the effect will last. Use zero for permanent spells.
Duration per level: For spells whose length is meaningful and it depends on the experience level of the spellcaster, this is the number of minutes per level. For example, if a spell is meant to work for 3 minutes + 1 minute per level of the spellcaster (that is, 4 min for a level 1 wizard; 5 min for a level 2 wizard; 6 min for a level 3 wizard and so on) then type a duration base of 3 and a duration per level of 1.
Material required: This is a code for the material component required to cast the spell, if any. See Inside Dream/Spells/Material components.
Code selector: This code distinguished between various types of the same base spell. See Inside Dream/Spells/Spell codes
Name: The name of the spell, as it will appear inside the spellbooks.
Icon: Resource ID for the icon that will represent the spell.

**Notes**

If the spell can be learned by the characters, or is cast by a magic item which the character can export to other scenarios, remember to insert it in the AdDB resource.

**Compatibility notes**

Under Dream 1.0, scenario Designers wishing to create a new spell had to use an ID between 200 and 369. Specific numeric IDs were allocated to each class (wizard or cleric) and level. This was found to be limiting (no more than 10 spells per spell level were allowed) and dangerous (two Designers could create overlapping spells). From Sword Dream 1.1 onward, any numeric ID is suitable for any spell: a Designer is, thus, requited to use IDs between 1000 and 9999, just like any other resource.

**Tips and tricks**

If a spell is meant to be cast only through magic items, then disregard the "Wizard or cleric" attribute, and don't require any material components. Use

zero as the spell level. Then write the ID of the spell inside the item description resource, as shown in Scenario resources/Objects.
You can create a spell which produces objects, either permanent or temporary (like the Wizard Sword). To do so, use code 21 for the Kind of magic, and type the ID of the object to be created inside the Selector code box.

**Texts**

**Resource signatures**

‘TEXT’: Written text to be spoken aloud on PlainTalk-equipped Macintoshes.

‘styl’: Styling information with font, size, style to be applied to the text when shown in the text windoid of Dream.

**What’s it for?**

‘TEXT' & 'styl': for the descriptions of the places and any other text which is to be presented to the player.

**Other needed resources**

No other resources are pre-required.

**How to create one**

Use ResEdit: create the TEXT resource and apply the font, style and size attributes which you prefer. Keep in mind that, unlike the picture windoid, the text windoid in Dream is fixed in size, so you can’t come up with anything too big. ResEdit will create the appropriate ‘styl’ resource.

Jot down the resource ID you gave to your text somewhere, as you’ll be required to type it to reference your text.

**Notes**

On a Macintosh with PlainTalk software installed, the TEXTs whose ID ends with “1” (e.g. 1001, 1011, 1021…) shall be spoken aloud with a male voice. Those whose ID ends with a 2 shall be spoken with a female voice. Others shall simply appear in the windoid.

The TEXT resource, ID 1128, will be displayed when the user chooses “About this scenario” in the Apple menu. You can use this feature to display your own shareware/freeware/beerware/anythingware notice.

**Tips and tricks**

You can also created colored text . To do so you need an application which supports styled TextEdit cut and paste (almost all modern word processors do, with the exception of Microsoft Word). To see if your word processor supports styled text, cut a piece of text from it and paste it inside the ScrapBook. If you see ‘styl’ at the bottom line of the window, then your application does support styled text. In that case, create the text inside the application, then paste it in ResEdit.

## Traps

**Resource signature**

‘Trap’: a mechanic or magic device which gets a change to work when the player characters enter a specific location.

**What’s it for?**

To make life more difficult to the player you can create traps and place them around. A trap is meant to be placed in some specified location, and can be configured to produce a wide variety of special effects.

**Other needed resources**

Before you create the Trap resource, write two pieces of text. The first will be shown if the trap snaps, and the second will be shown if the group ruffian recognizes and dismantles the trap (it’s a 5% chance per experience level).

**How to create one**

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the ‘Scenario Resources.rez’ file.

Trap base dmg; dmg num dice; dmg dice size

Together, these three values help determine the amount of damage a trap inflicts onto each character in the group. For example: (5, 0, 0) means exactly 5 HP of damage. (0, 1, 6) means 1 to 6 (checked separately for each player character). (1, 2, 6) means 3 to 13.

Invoke spell ID

If you write a non-zero id here, the Dream application will execute the spell whose id you wrote, and target it onto the characters in the group. The spell's area of effect will always be transformed to "every character in the group", no matter what the spell template says. For example, you may create a poisoned trap using spell ID 322 (the code for the third level cleric spell, Poison).

Text if sprung

Id for the TEXT piece which will be shown when the trap snaps (or zero for none)

Text if found

Id for the TEXT piece which will be shown if the group ruffian keeps the trap from functioning. Can be zero (for none) if you won't allow ruffians to disarm the trap.

Ruffian can disarm

If set to "1" (and it should), the group ruffian gets a chance to disarm it.

Won't spring if group invisible (Δ 2.0)

Set to "1" when it makes sense, that is if the trap shouldn't spring when the group is invisible. Under Dream version 1.5 this feature is not implemented, and invisible characters will fall into the trap.

Won't spring if group flying

Set to "1" if the trap is mechanical in nature and depends on the characters' weight. In this case a flying character gets the chance to avoid the trap.

Trap springs only once

Set to "1" if the trap works only once. This is how traps are supposed to work, anyhow.

Sound (Δ 2.0)

Write here the id for a 'snd ' resource which will be played if the trap snaps.

**Compatibility**

Traps are supported only by version 1.1 and following of the Dream application. If you choose to insert one or more you should make certain that your Info resource asks for version 1.1 of the Dream engine. If you don't, a player using Dream 1.0 will be able to open the scenario, but the software will then declare the scenario damaged or faulty when the trap is found, and quit on the user.

**Version number**

**Resource signature**

'vers': Version number. IDs: 1 and 2.

**What's it for?**

The version number is used by the Finder

**Other needed resources**
None
**How to create one**
With ResEdit.
**Notes**
The Dream application doesn’t require the presence of a ‘vers’ resource. Dream takes the version information it needs from the ‘Info’ resource. For the convenience

of your game players, I suggest that you do include a ‘vers’ resource, and keep it in sync with the information of the ‘Info’ resource.

### Wandering monsters list

#### Resource signature

‘Wndr’: a list of monsters to be met randomly during adventuring.

#### What’s it for?

You may wish to add spice to your scenario with wandering monsters. Without wandering monster, a game scenario would only present dangers where encounters are planned and accounted for. With wandering monsters, a place presents dangers of its own.

#### Other needed resources

Before creating a wandering monsters list, you must complete creation of all monster types involved (see Monsters). The place where monsters will appear must also be ready (See Places).

For every wandering monsters list there ought to be an arena where encounters will take place. Such an arena must have the same ID as the wandering monsters resource, and can be created either before or after the wandering monsters list.

#### How to create one

Use ResEdit. Before you create the resource, check that you have opened both your scenario file and the ‘Scenario Resources.rez’ file.

Create a Wndr resource whose numeric ID is the same as the place the monsters will infest. Then write inside the numeric IDs of all the monster races the player might encounter there. Order is not an issue. The probability of meeting each monster kind is the same. The number of monsters met depends on the “Number appearing” field inside the monster definition (see Monster).

Example: the Dawn Valley (‘Plac’ ID 1000) is infested with Kobolds ('Mstr' ID 1176) an Goblins ('Mstr' ID 1255). Simply create a 'Wndr' resource, ID 1000 (same ID as the 'Plac') and put inside it the numbers 1176 and 1255.

#### Notes

Avoid excessive use of wandering monsters. They should be a nuisance, not a major peril. If you choose to put wandering monsters everywhere (like I did in “Spirit of Darkness”), you ought to put a safe haven somewhere (so that players can rest and cure their wounds during adventuring: maybe a holy temple of some good deity was left inside that black-charred enemy land they are visiting).

#### Tips and tricks

If you wish to reuse the same monster race for multiple wandering monsters list, with varying difficulty, do this. First create the monster race (see Monsters). Put a low number in the “Number appearing” box. Now (using ResEdit) select the monster race, copy and paste it. You just created a perfect

duplicate: open it and put a higher number in the “Number appearing” box. You may now use the original in the Wandering monster list for a low-peril place, and the variant for a high-peril place (maybe a lower level of the same dungeon?)
To have a monster race appear more frequently than another, you can use more than once the same monster ID inside the list. In the example before, if you wrote a list consisting of the numbers 1176, 1176, 1250 you’d have a double occurrence for kobolds than for goblins.

## Sometime soon

There is nothing sacred or immutable in Dream: I started the Dream project with a very broad view of its possibilities, and I'm not finished. Dream, inside, is totally object-oriented and modular. This means that I can change major parts of the architecture without disturbing the rest of the game engine. Animated displays, three-D displays, more intelligent monsters, more encompassing spells: all of this can be done. And I will do it, if given the possibility.

I thought I'd say something about future enhancements to the Dream engine, so that you, the scenario Designer, can plan in advance your developments.
A few gaming ideas, in fact, would be difficult to implement right now, but much easier later on, when these enhancements to Dream will be yours to tap into. Other game ideas can be developed into game scenarios right now, and enhanced later: future versions of Dream will, in fact, run current game scenarios unmodified, and be able to run new scenarios which the current version is unable to support.
A few enhancements will be transparent to your scenario. For example, when players will be able to drive Dream entirely through speech commands, your scenario will play and there will be no differences in the quality of game play or its pathos.
I'll be creating future versions of Dream if I get enough shareware fees to keep my interest in the program. You, as a game designer, most certainly share my interest in seeing our labor rewarded: your own game scenarios, in fact, are independent software, which you can release — as public domain, freeware or shareware — yourself.

**DreamBasic**
Dream v 2.0 will feature a programming language, DreamBasic. With

DreamBasic, you'll be able to write short pieces to code that will be attached to monsters, objects, encounters and spells.
For example, you'll be able to write things like "if there are two or less adventurers, then have three Kobolds at them; if there are three or four player characters, let's dispatch four; otherwise, send five".
Or maybe: "when this monster is killed, the others will panic and flee".
If you can't program, don't fear. DreamBasic will be extremely simple to use. If you can create macros in Excel or QuicKeys, you'll be able to create DreamBasic code.
The DreamBasic engine is already written: I'm just waiting to find time enough to insert it into Dream and debug it.

**Interaction with monsters**

Right now, it’s kill them or flee (well, there are good-aligned monsters, but they don’t really make much of a difference). In the future, the characters will be able to try to give food to monsters, or try to converse with a few of them.

### More character races and spells

Every new version of the Dream application features more spells. This is sure to influence play. For example, suppose you create a scenario whose main feature is a complex labyrinth. Then I go ahead and create a “destroy walls” spell. Poof: the scenario isn’t worth a thing. If you fear this, there are a couple of things you could do: first, drop me a note, so that I won’t step blindly ahead and ruin your work unknowingly. Second, include a short note in your scenario documentation: advice players to avoid use of the unlisted spells if they don’t wish to spoil the fun.

## Spells

### The spell icon

In release 2.0 spells will be shown by icon, and not by name. (They would already if I had been able to convince Eugenio to draw them — now I’m going to corner him). If you create a spell, do come up with an appropriate icon, and store a reference to it in the appropriate field of the spell template.

# Tips and tricks

**Overriding**

In exceptional cases, a scenario can contain resources with IDs lower than 1000, to obtain an effect known as “overriding”.

Overriding takes place when you copy a resource from inside the Dream Database, modify it to suit your own nefarious purpose, and then place the copy inside your game scenario. If you do this, Dream will use your modified resource instead of its own internal resource.

**Effects of overriding**

By overriding, you can obtain very interesting effects. Suppose that you wish to create a scenario where spell components are very hard to find. Still, you wish to put a mage tower inside the scenario (just in case the player wishes to add a wizard to his/her group). This is easy to do through overriding, and impossible to do otherwise.

Here’s how: open a copy of the Dream application with ResEdit. Locate and copy the MENU resources used inside wizard towers. Open your own game scenario file. Paste the MENU in it. Now open the MENU resource, and uncheck the “Enabled” attribute of every item that refers to a spell component. That’s all.

By overriding, a scenario can change the rules by which games are played. For example, during a game scenario all spells which affect undead creatures might not work (to reflect a powerful spell from a demon lord). To do so, the scenario designer puts a copy of the spell resources (whose original copy stays inside the Dream Database), where the effect of the spell is nullified. The copy inside the scenario overrides the original description, and the effect is obtained.

**Massive overriding**

With overriding, the sky’s the limit. Suppose you wish to create a science fiction adventure.

First, draw the needed graphics and icons.

In your science fiction adventure, there shouldn’t be wizards and clerics, of course, but spaceship pilots and scientists. So, you override the character class icons and their names.

There are no spells, so you avoid use of spellcasting classes. This leaves you with two classes (rogues and fighters), but that should be enough. So, you don’t use wizard towers, shrines, inns and temples, but stick to banks and brothels. There’s no place in your space station for a brothel? No problem: override it and transform it into a spaceship pilots’ guild. Or a teleporting station.

One last touch: create an ‘Info’ resource where you specify that a maximum of zero characters should be allowed to enter your scenario. This to make sure

that nobody will take a paladin inside your space station (nor a fighter that would be instantly transformed into a space pilot).

**Warning**

Future versions of the Dream application will, of course, need changes from the current form. If you do use overriding, problems might arise. For example, the “inn” menu was changed during the update from Dream 1.0 to Dream 1.5. If you had created a scenario that overrides the Inn menu when 1.0 was the current version, Dream 1.5 would use your menu (based on Dream 1.0) instead of its own. The player would be unable to rest inside inns.

There’s a fast and easy rule to this problem: limit yourself to overriding those contents of the Dream Database which are listed by ID in the Dream Database chapter of this manual.
Don’t touch things defined inside the Dream application itself. This way you are somewhat limiting the power of the overriding mechanism, but are guaranteeing yourself future compatibility.