Portals in CWG Buddha and Sun-Tzu Releases

Ken Ray

<kenr86@hotmail.com>

September 2003

Abstract

One neat feature of the CWG releases is the addition of portals. These are one-way access points from one room to another room, anywhere on the mud. Consider them as a teleport device, a rent in the time-space continuum, or just a neat way for players to get to some other location. This article will lead you through creating a variety of portals in your mud.

Version, Copyright and License Information

The code described in this document is based on the Circle "cwg-buddha" release, dated August 09, 2003. Refer to that for the various software levels and versions. The "cwg" versions of the CircleMUD code have been put together by Mark Garringer, zizazat@hotmail.com.

Enhancements to that code documented here are available under the same terms as the standard CircleMUD license, as documented in the CircleMUD code. The three basic requirements are:

1) You must not use CircleMUD to make money or be compensated in any way.

2) You must give the authors credit for their work.

3) You must comply with the DikuMUD license.

This documentation has been provided to help CircleMUD administrators, builders and coders better understand how to use the Portal enhancements in the cwg releases, and it is the intent of the author that this receive widespread distribution through the CircleMUD community. While all care has been taken in the design and development of the code enhancements, and in the writing of this documentation, no warranty expressed or implied is associated with either the code described herein or this documentation. You use it entirely at your own risk.

This is Version 1.0 of the documentation, dated September 12 2003.

Contents

1 BASIC PORTAL OBJECTS

1.1 Create a Portal Object.

Find a free object vnum within the zone that the portal is to be located in, and either manually edit the .obj file for that zone, or use oedit to create the object on line.

```
500H 100M 82V > oedit 3091
-- Item number : [3091]
1) Namelist : portal shimmering
2) S-Desc   : a shimmering portal
3) L-Desc   :-
Suspended in midair, and shimmering with a soft pink glow, a
portal leads to who knows where.
4) A-Desc   :-
As you step into the shimmering energy field, you feel a
momentary tingling over your body.  You think you heard a faint
sizzling, sparking noise, but you are not certain.
5) Type        : PORTAL
6) Extra flags : NOBITS
7) Wear flags  : NOBITS
8) Weight      : 0
9) Cost        : 0
A) Cost/Day    : 0
B) Timer       : -1
C) Values      : 3170 0 -1 -1
D) Applies menu
E) Extra descriptions menu
M) Min Level   : 0
P) Perm Affects: NOBITS
S) Script      : Not Set.
Q) Quit
Enter choice :
```

Things to note:

i. The "A-Desc" (action description), if present, will be displayed to a character who steps into the portal.

ii. Extra Descriptions can be added as you desire. As with most objects, so players can actually see some information if they enter "look at portal", this has to be done as an extra description:

```
Enter choice : e
Extra desc menu
1) Keyword: portal shimmering
2) Description:
   The portal appears to be a circular glowing area of energy,
infinitely thin, suspended somehow just above the floor.  It is
large enough for two people to step through (or into) it at the
same time.  You have absolutely no idea where it leads.

3) Goto next description: <Not set>

0) Quit
```

iii. There are four object values that you need to set. The first is the vnum of the room that the portal will lead to (3170 in the case above). Note that a portal can lead to a room in another zone. If you haven't created this room yet, set the destination room to -1 (NOWHERE). The second and third values deal with whether the portal also has container-like capabilities; these are discussed later in this document. The final value determines the portal's appearance. A negative value means the portal is transparent – that is, you will be shown the name of the destination room if you look in the portal, values from zero upwards correspond to a preset list of portal appearances.

```
Enter choice : c
Which room number is the destination? : 3110
1) CLOSEABLE
2) PICKPROOF
3) CLOSED
4) LOCKED
Container flags: NOBITS
Enter flag, 0 to quit : 0
Vnum of key to unlock portal (-1 for no key) : -1
What is the portal's appearance? (-1 for transparent) : -1
```

Earlier releases of OasisOLC (up to 2.0.1) had a bug where if you only altered the object values, the code didn't record this as making a change to the object, and hence not prompt you to save the object.

iv. You probably want to clear the "take" bit of the object wear flags. After all, you don't want your characters walking around with a portal. Another option would be to allow it to be taken, but make the weight so they can't pick the thing up.

v. You also need to set the timer (menu item B) to –1. This will prevent the portal from automatically "expiring".

1.2 Load the Object in the Zone

Add commands into the zone file to load this new object into the desired room. You can edit the .zon file directly:

```
* Portals
R 0 3000 3091 -1 0      Portal
O 1 3091 1 3000 0
```

Or use zedit:

```
500H 100M 82V > zedit
Room number: 3000               Room zone: 30
Z) Zone name      : Northern Midgaard Main City
L) Lifespan       : 15 minutes
B) Bottom of zone : 3000
T) Top of zone    : 3099
R) Reset Mode     : Normal reset.
[Command list]
0 - Remove a bulletin board [3099] from room.
1 - Load a bulletin board [3099], Max : 2    % 0
2 - Remove a shimmering portal [3091] from room.
3 - Load a shimmering portal [3091], Max : 1    % 0
4 - <END OF LIST>
```

```
N) New command.
E) Edit a command.
D) Delete a command.
C) Change Percentage of command.
Q) Quit
Enter your choice :
```

Notice how the portal is first removed, and then a new one loaded. This is always good practice for objects where only one should exist.

1.3 Test the Portal

We can see the portal is in the room:

```
500H 100M 82V > look
The Reading Room
   You are in a small, simple room which is mostly empty, save
a few wooden desks and benches.  To the southeast you hear the
bustle of the Temple of Midgaard, a sharp contrast to the
relative quiet of this peaceful
room.
[ Exits: se out ]
Suspended in midair, and shimmering with a soft pink glow, a
portal leads to who knows where.
A large bulletin board is mounted on a wall here.

500H 100M 82V > look at portal
   The portal appears to be a circular glowing area of energy,
infinitely thin, suspended somehow just above the floor.  It is
large enough for two people to step through (or into) it at the
same time.  You have absolutely no idea where it leads.
```

Lets try to do things with this portal now.

```
500H 100M 82V > look in portal
After seconds of concentration you see the image of Cityguard
HeadQuarters.

500H 100M 82V > enter portal
As you step into the shimmering energy field, you feel a
momentary tingling over your body.  You think you heard a faint
sizzling, sparking noise, but you are not certain.

Cityguard HeadQuarters
   You are inside a tidy office.
[ Exits: None!]
A safe is placed in a dark corner of the room.
A desk is set against the western wall.
A cityguard stands here, looking very upset.
A cityguard stands here, looking very upset.
A cityguard stands here, looking very upset.
A cityguard stands here, looking very upset.
The Chief Guard is looking very upset.

500H 100M 82V >
```

A few things to notice. We can look through the portal, since when we created it we specified it as transparent. So, when we look in the portal, we are shown the name of the destination room. And, when we entered the portal, because we had set the action description, which is listed just before we are moved to that new room.

Now, lets change the portal's transparency to 0 (opaque). Now if we try to look in it, this is what we see:

```
500H 100M 82V > look in portal
All you can see is the glow of the portal.
```

Alternatively, even if the portal is "transparent", if the destination room is dark and we are not able to see in the dark, we get a different message:

```
500H 100M 82V > look in portal
It is pitch black...
```

1.4 "Use the Source, Luke"

OK, how are some of the key elements implemented?

1.4.1 Defining The New Object Type

First, we need to define an object type of ITEM_PORTAL. In structs.h, there is the list of possible item types. Add ITEM_PORTAL as the next available type:

```
#define ITEM_PORTAL    28  /* Item is a portal           */
```

You also have to make sure NUM_ITEM_TYPES in oasis.h is updated as well:

```
#define NUM_ITEM_TYPES         29
```

When the objects are initially loaded (object_parse in db.c), we need to set the object timer to –1. This is to prevent the portal from “expiring”, since the portal spell creates a portal object that only lasts for a short period of time.

```
/* *** make sure portal objects don't expire *** */
if (GET_OBJ_TYPE(obj_proto + i) == ITEM_PORTAL) {
  GET_OBJ_TIMER(obj_proto + i) =
              GET_OBJ_VAL(obj_proto + i, 2);
}
```

1.4.2 Entering a Portal

Entering a portal is implemented with the "do_enter" command. This routine handles a number of situations, including vehicles and doors, as well as our portal. This routine is in act.movement.c, with the relevant portion of the code from ACMD(do_enter) shown below. Because this has to handle either entering an enterable object or entering a door, the first test to see if the argument of the enter command is an object that is in the room, and then we check to see if the character can see the object.

Once we have an object, we see if it is an object that can be entered – either a vehicle or a portal. If the portal is closed, the standard “It is closed” message is sent to the character. Otherwise, we move the character to the destination (unless the portal leads nowhere), providing appropriate messages to the character, and to other people in the portal’s room who can see the character.

Once the character arrives at the destination, an arrival message is send to all people who can see the character in the destination room. We also get the character to look at the room he has arrived in. Note that this happens even if they didn't go anywhere – now this could add confusion if the portal was in a maze of twisty passages, all alike. You can start to see how devious portals can be.

```
if (*buf) {
  /* Is the object in the room? */
  obj = get_obj_in_list_vis(ch,buf, NULL,
                    world[ch->in_room].contents);
  /* Is the object in the character's inventory? */
  if (!obj)
    obj = get_obj_in_list_vis(ch,buf, NULL, ch->carrying);
  /* Is the character carrying the object? */
  if (!obj)
    obj = get_obj_in_equip_vis(ch, buf, NULL, ch->equipment);
  /* We have an object to enter */
  if (obj) {
    if (GET_OBJ_TYPE(obj) == ITEM_VEHICLE ||
        GET_OBJ_TYPE(obj) == ITEM_PORTAL) {
      if ((GET_OBJ_TYPE(obj) == ITEM_PORTAL) &&
           OBJVAL_FLAGGED(obj, CONT_CLOSED)) {
        send_to_char(ch, "But it's closed!\r\n");
      } else {
        if ((GET_OBJ_VAL(obj, 0)            != NOWHERE) &&
            (real_room(GET_OBJ_VAL(obj, 0)) != NOWHERE)) {
          act(obj->action_description, TRUE, ch, obj,
                     0, TO_CHAR);
          act("$n enters $p.", TRUE, ch, obj, 0, TO_ROOM);
          char_from_room(ch);
          char_to_room(ch, real_room(GET_OBJ_VAL(obj, 0)));
          if (GET_OBJ_TYPE(obj) == ITEM_PORTAL)
            act("$n arrives from a puff of smoke.", FALSE,
                   ch, 0, 0, TO_ROOM);
        }
        look_at_room(ch, 1);
      }
    } else {
      send_to_char(ch, "You can't enter that!\r\n");
    }
    return;
  }
for (door = 0; door < NUM_OF_DIRS; door++)
    if (EXIT(ch, door))
      if (EXIT(ch, door)->keyword)
        if (!str_cmp(EXIT(ch, door)->keyword, buf)) {
          perform_move(ch, door, 1);
          return;
        }
  /* Couldn't find what they wanted to enter. */
  send_to_char(ch, "There is no %s here.\r\n", buf);
}
```

The first series of tests to find an actual object that matches the name supplied on the command first looks for an object in the room, then an object in the character's inventory, and finally in the character's equipment list. This is to allow for portal items that could be taken and held (or worn). If the object is found, but is not a portal (or vehicle), the you are told you can't enter that.

1.4.3 Looking into Portals

Looking in a portal is implemented in act.informative.c, in the look_in_obj routine. This is another fairly complex routine, because of all the situations that it has to cater for. There are six main clauses in the main if-if else – else block.

i. Have they forgotten to specify a target object? Tell them they need to provide a target object.

ii. Are they looking at an object that is not here? Tell them it can't be found.

iii. Is the object a portal that cannot be closed? Then, we look at the portal appearance.

iv. Is the object anything other than a drink container, fountain, container or portal (since we have already tested for non-closing portals, these are the four types of items that can actually contain something) Tell them there is nothing inside that.

v. Is the object a container or a portal? Give a listing of the contents. Note that a portal container will contain nothing – not that the character may know the portal is actually a portal, we want it to behave like any other container, until something is put in it.

vi. The only possibility is the item is a fountain or drink container, so list those contents.

The relevant section of the code is shown below:

```
void look_in_obj(struct char_data *ch, char *arg)
{
  struct obj_data *obj = NULL;
  struct char_data *dummy = NULL;
  int amt, bits;

  if (!*arg) {
    send_to_char(ch, "Look in what?\r\n");
  } else if (!(bits = generic_find(arg,
                     FIND_OBJ_INV | FIND_OBJ_ROOM |
                     FIND_OBJ_EQUIP, ch, &dummy, &obj))) {
    send_to_char(ch,
        "There doesn't seem to be %s %s here.\r\n",
        AN(arg), arg);
  } else if ((GET_OBJ_TYPE(obj) == ITEM_PORTAL) &&
             !OBJVAL_FLAGGED(obj, CONT_CLOSEABLE)) {
    if (GET_OBJ_VAL(obj, 3) < 0) {
      /* You can look through the portal to the destination */
      /* where does this lead to? */
      room_rnum portal_dest = real_room(GET_OBJ_VAL(obj, 0));
      if (portal_dest == NOWHERE) {
        send_to_char(ch,
            "You see nothing but infinite darkness...\r\n");
      } else if (IS_DARK(portal_dest) &&
                 !CAN_SEE_IN_DARK(ch)) {
        send_to_char(ch,
            "You see nothing but infinite darkness...\r\n");
      } else {
        send_to_char(ch,
            "After seconds of concentration you see the
                 image of %s.\r\n",
                 world[portal_dest].name);
      }
    } else if (GET_OBJ_VAL(obj, 3) < MAX_PORTAL_TYPES) {
      /* display the appropriate description from the list */
      send_to_char(ch, "%s\r\n",
          portal_appearance[GET_OBJ_VAL(obj, 1)]);
    } else {
      /* We shouldn't really get here,
         so give a default message */
      send_to_char(ch,
          "All you can see is the glow of the portal.\r\n");
    }
  } else if ((GET_OBJ_TYPE(obj) != ITEM_DRINKCON) &&
             (GET_OBJ_TYPE(obj) != ITEM_FOUNTAIN) &&
             (GET_OBJ_TYPE(obj) != ITEM_CONTAINER)&&
             (GET_OBJ_TYPE(obj) != ITEM_PORTAL))     {
    send_to_char(ch, "There's nothing inside that!\r\n");
} else if ((GET_OBJ_TYPE(obj) == ITEM_CONTAINER) ||
             (GET_OBJ_TYPE(obj) == ITEM_PORTAL)) {
    if (OBJVAL_FLAGGED(obj, CONT_CLOSED))
      send_to_char(ch, "It is closed.\r\n");
    else {
      send_to_char(ch, "%s", fname(obj->name));
      switch (bits) {
        case FIND_OBJ_INV:
          send_to_char(ch, " (carried): \r\n");
          break;
        case FIND_OBJ_ROOM:
          send_to_char(ch, " (here): \r\n");
          break;
        case FIND_OBJ_EQUIP:
          send_to_char(ch, " (used): \r\n");
          break;
      }
      list_obj_to_char(obj->contains,
          ch, SHOW_OBJ_SHORT, TRUE);
    }
  } else {    /* item must be a fountain or drink container */
      <irrelevant lines shipped>
  }
}
```

Lets look at the third if-else section in greater detail. We use the list of portal appearances defined in act.informative.c:

```
const char *portal_appearance[] = {
    "All you can see is the glow of the portal.",
    "You see an image of yourself in the room - my,
        you are looking attractive today.",
    "All you can see is a swirling gray mist.",
    "The scene is of the surrounding countryside, but
        somehow blurry and lacking focus.",
    "The blackness appears to stretch on forever.",
    "Suddenly, out of the blackness a flaming red eye
        appears and fixes its gaze upon you.",
    "\n"
};
#define MAX_PORTAL_TYPES       6
```

If the portal is “transparent”, that is, the appearance value is negative, we are going to show them the destination – that is, if the portal goes somewhere, and the character could actually see in that room. In that case, they get the “infinite darkness” message. Assuming there is actually a destination, and they can see in that room, then they get a message giving the name of the room.

If one of the portal appearance types has been specified, then this is shown to the character. Of course, more appearance types can be defined, but don’t forget to update the MAX_PORTAL_TYPES to match the number of entries in the array.

There is also a catchall message if we somehow fall through – if you defined an appearance type that doesn’t actually exist, for example.

1.4.4 OasisOLC Oedit Changes

Changes have also been made to the oedit command in OasisOLC (file is oedit.c), in the oedit_disp_val1_menu, oedit_disp_val2_menu and oedit_disp_val3 routines, adding case blocks to the switch statements for the ITEM_PORTAL situations:

```
(oedit_disp_val1_menu)

case ITEM_PORTAL:
  write_to_output(d,
       "Which room number is the destination? :");
  break;

(oedit_disp_val2_menu)

case ITEM_CONTAINER:
case ITEM_PORTAL:
  /*
   * These are flags, needs a bit of special handling.
   */
  oedit_disp_container_flags_menu(d);
  break;

(oedit_disp_val3_menu)

case ITEM_PORTAL:
  write_to_output(d,
      "Vnum of key to unlock portal (-1 for no key) : ");
  break;

(oedit_disp_val4_menu)

case ITEM_PORTAL:
  write_to_output(d,
  "What is the portal's appearance? (-1 for transparent) : ");
  break;
```

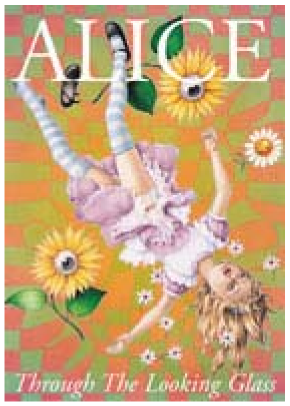
Note that we make use of the existing ITEM_CONTAINER logic for the second flag value. Also, the processing for oedit_disp_val2_menu needs to be modified to include ITEM_PORTAL with the code for ITEM_CONTAINER. This does the actual container flag processing. Note that if you are editing an existing object, the current container flags are not “remembered” when you are editing the second value item.

```
case ITEM_CONTAINER:
case ITEM_PORTAL:
  /*
   * Needs some special handling since we are dealing with
   * flag values here.
   */
  if (number < 0 || number > 4)
    oedit_disp_container_flags_menu(d);
  else if (number != 0) {
    TOGGLE_BIT(GET_OBJ_VAL(OLC_OBJ(d), 1), 1 << (number - 1));
    OLC_VAL(d) = 1;
    oedit_disp_val2_menu(d);
  } else
    oedit_disp_val3_menu(d);
  break;
```

2 MORE ADVANCED PORTALS

Here we shall look at some more examples that make use of some of the advanced capabilities available. What we want to do is have portals that are not obviously portals.

2.1 Through the Looking Glass



Let's pretend there's a way of getting through into it, somehow, Kitty. Let's pretend the glass has got all soft like gauze, so that we can get through. Why, it's turning into a sort of mist now, I declare! It'll be easy enough to get through--' She was up on the chimney-piece while she said this, though she hardly knew how she had got there. And certainly the glass WAS beginning to melt away, just like a bright silvery mist.

In another moment Alice was through the glass, and had jumped lightly down into the Looking-glass room. The very first thing she did was to look whether there was a fire in the fireplace, and she was quite pleased to find that there was a real one, blazing away as brightly as the one she had left behind. `So I shall be as warm here as I was in the old room,' thought Alice: `warmer, in fact, because there'll be no one here to scold me away from the fire. Oh, what fun it'll be, when they see me through the glass in here, and can't get at me!'

"Through the Looking-Glass and What Alice Found There",
Chapter 1, Looking-Glass House
Lewis Carroll (1872)

Lewis Carroll started something with mirrors not being all they seem. Here we have what looks like a plain mirror:

```
-- Item number : [3092]
1) Namelist : mirror
2) S-Desc   : a tall mirror
3) L-Desc   :-
A tall mirror reflects the contents of the room.
4) A-Desc   :-
<not set>
5) Type         : PORTAL
6) Extra flags : NOBITS
7) Wear flags  : NOBITS
8) Weight      : 0
9) Cost        : 0
A) Cost/Day    : 0
B) Timer       : -1
C) Values      : 3161 0 -1 1
D) Applies menu
E) Extra descriptions menu
M) Min Level   : 0
P) Perm Affects: NOBITS
S) Script      : Not Set.
Q) Quit
```

```

Extra desc menu
1) Keyword: mirror
2) Description:
The mirror is fixed in a tall wooden frame, with fancy gilt
flower patterns carved into the wood.

3) Goto next description: <Not set>

0) Quit
Enter choice : 0
```

As you can see, we have chosen the portal appearance number 1 (a mirror reflecting the contents of the room). Now lets start to play.

2.1.1 Mirror, Mirror on The Wall

We first loaded the mirror into the bar of our old favorite watering hole, the Grunting Boar Inn.

```
> load obj 3092
You create a tall mirror.
```

The place could do with some sprucing up, anyway. Of course, normally we would put the “O” (load object) command in the zone file, but this will suffice for the purposes of this example. Now lets have a look around and see what is here.

```
> look
The Grunting Boar
   You are standing in the bar.  The bar is set against the
northern wall, old archaic writing, carvings and symbols cover
its top.  A fireplace is built into the western wall, and
through the southeastern windows you can see the temple square.
This place makes you feel like home.
A small sign with big letters is fastened to the bar.
[ Exits: w ]
A tall mirror reflects the contents of the room.
A Peacekeeper is standing here, ready to jump in at the first
sign of trouble.
A cityguard stands here.
A singing, happy Drunk.
A bartender watches you calmly, while he skillfully mixes a
drink.

> look at mirror
The mirror is fixed in a tall wooden frame, with fancy gilt
flower patterns carved into the wood.

> take mirror
A tall mirror: you can't take that!

>
Zapper has arrived.

> look in mirror
You see an image of yourself in the room - my, you are looking
attractive today.
```

As we would expect, of course – we can't take the mirror, but it does appear to work as a mirror.

2.1.2 Into the Mirror

Notice that while we were doing this, another player character arrived. Lets step into the mirror.

```
> enter mirror
Park Cafe
   You are inside Park Cafe, a very cozy place with many tables
and seats where you can relax and enjoy the view.  To the east
you can see the entrance to the park.
[ Exits: w ]
A Sexton is sitting here, drinking hot tea.
The Maid is waiting for your order.
An oozing green gelatinous blob is here, sucking in bits of
debris.
```

2.1.3 What do Others See?

Again, just as we expected. But what did Zapper see?

```
>
Maclir enters a tall mirror.

> look
The Grunting Boar
   You are standing in the bar.  The bar is set against the
northern wall, old archaic writing, carvings and symbols cover
its top.  A fireplace is built into the western wall, and
through the southeastern windows you can see the temple square.
This place makes you feel like home.
A small sign with big letters is fastened to the bar.
[ Exits: w ]
A tall mirror reflects the contents of the room.
A cityguard stands here.
An oozing green gelatinous blob is here, sucking in bits of
debris.
A Peacekeeper is standing here, ready to jump in at the first
sign of trouble.
A singing, happy Drunk.
A bartender watches you calmly, while he skillfully mixes a
drink.

>
The cityguard leaves west.
```

We see the message when Maclir enters the mirror, and of course he is no longer listed as one of the occupants of the room. Lets follow him:

```
> look in mirror
You see an image of yourself in the room - my, you are looking
attractive today.

> enter mirror
Park Cafe
```

```
   You are inside Park Cafe, a very cozy place with many tables
and seats where you can relax and enjoy the view.  To the east
you can see the entrance to the park.
[ Exits: w ]
Maclir the Implementor is standing here.
A Sexton is sitting here, drinking hot tea.
The Maid is waiting for your order.
An oozing green gelatinous blob is here, sucking in bits of
debris.

>
```

Cool. Did Maclir see us arrive?

```
>
Zapper arrives from a puff of smoke.
> look
Park Cafe
   You are inside Park Cafe, a very cozy place with many tables
and seats where you can relax and enjoy the view.  To the east
you can see the entrance to the park.
[ Exits: w ]
Zapper the Apprentice of Magic is standing here.
A Sexton is sitting here, drinking hot tea.
The Maid is waiting for your order.
An oozing green gelatinous blob is here, sucking in bits of
debris.
```

We can see Zapper arriving from the end of the portal. And when we look in the room, Zapper shows up in the list of occupants.

So, there we have it. You could have added a action description to the portal, so that the person entering the portal sees some appropriate message, but that doesn't affect the basic operation of it.

2.2 Oh dear! Oh dear! I shall be late!

Continuing with our Lewis Carroll inspired portals, now what if we want something slightly more complicated?

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, `and what is the use of a book,' thought Alice `without pictures or conversation?'

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so very remarkable in that; nor did Alice think it so very much out of the way to hear the

Rabbit say to itself, `Oh dear! Oh dear! I shall be late!' (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually took a watch out of its waistcoat-pocket, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

"Alice's Adventures in Wonderland"
Chapter 1, Down the Rabbit Hole,
Lewis Carroll (1865)

Now, it is beyond the scope of this document to show you how to implement mobs that act like the White Rabbit (although some simple scripting would suffice), but can we make some sort of magical rabbit hole?

2.2.1 Create a "Silent" Portal

One trick we can use when we create an object is to leave the long description field blank. When someone does a "look room", objects without a long description do not appear on the list of objects in that room. I refer to these as "silent objects". They are still there, and people can still interact with them in the usual manner, but they don't advertise their presence. In the routine "list_obj_to_char" (in act.informative.c), there is a loop through all of the objects passed to the routine to be listed. At the beginning of the loop are the two tests:

```
if (i->description == NULL)
  continue;
if (str_cmp(i->description, "undefined") == 0)
  continue;
```

This causes these objects not to be listed after the room description.

Now let's create the rabbit hole.

```
-- Item number : [3093]
1) Namelist : hole rabbit burrow
2) S-Desc   : a rabbit hole
3) L-Desc   :-
undefined
4) A-Desc   :-
You start to fall - further and further, deeper and deeper into
the earth.  Surely, this is no ordinary rabbit hole.  You start
to wonder if it will ever stop, when you start to slow down,
and, thump, thump, thump, down you land on a heap of leaves,
without hurting yourself at all.
5) Type        : PORTAL
6) Extra flags : NOBITS
7) Wear flags  : NOBITS
8) Weight      : 0
9) Cost        : 0
A) Cost/Day    : 0
B) Timer       : -1
C) Values      : 1207 0 -1 4
D) Applies menu
```

```
E) Extra descriptions menu
M) Min Level   : 0
P) Perm Affects: NOBITS
S) Script      : Not Set.
Q) Quit
Enter choice :
```

Notice that oedit will put the text “undefined” in the l-desc field if we leave that blank. You will also notice the rather description action description. We also add an e-desc to this object too:

```
Enter choice : e
Extra desc menu
1) Keyword: hole rabbit burrow
2) Description:
This burrow appears to be slightly larger than your average
rabbit hole.

3) Goto next description: <Not set>

0) Quit
Enter choice :
```

This is what will be shown if a character types “look at hole”.

2.2.2 A Large Rabbit Hole

Now, we load our portal object to the target room. Because the rabbit hole is “silent”, we should provide some other clues to players that something is here. We can do that in the actual text of the room description. Finally – a way to get players to read all of your carefully crafted words.

```
500H 100M 82V > look
The Dirt Path
   You are walking along a narrow dirt path through the lush,
green, fresh Midgaard countryside.  You can see to the horizon
to the north and east; the busy city of Midgaard lies to the
south.  All around you is healthy green grass and an occasional
large oak tree.  The sun feels wonderful on your face and a
pleasant wind blows through your hair.  Birds chirp quietly to
themselves and you can smell the faint scent of flowers and
freshly cut grass.  You feel like you could lie down in the
grass and stay here forever, surrounded by powerful beauty in
all directions.
   There is a large rusty gate sitting in a stone archway just
to the west.  To the north is a thick hedge, with what appears
to be a large rabbit hole under the hedge.
[ Exits: e w ]

500H 100M 82V > look at hedge
This thick hedge is about ten feet high, and is made from
tightly packed branches and brambles.  There is no way through
this mass of shrubbery.  You notice a rabbit hole about half
way along the hedge.

500H 100M 82V > look at hole
```

```
This burrow appears to be slightly larger than your average
rabbit hole.
```

So there we have it. A room, and there are at least some extra descriptions added to it. At the moment, there is no indication that the rabbit hole is anything more than another e-desc added to bring a slightly literary touch to the zone. As we can see, the hedge has the desired affect:

```
500H 100M 82V > look north
There is a thick hedge in that direction, preventing you from
going any further that way.

500H 100M 82V > n
Alas, you cannot go that way...
```

That uses the standard trick of defining an exit in that direction, but the exit's destination is –1 (nowhere). The exit description will be shown when someone looks that way, but since the destination is nowhere, you get the "Alas…" message. The relevant definitions from redit are shown below:

```
Enter choice : 5
1) Exit to     : -1
2) Description :-
There is a thick hedge in that direction, preventing you from
going any further that way.

3) Door name   : <NONE>
4) Key         : 0
5) Door flags  : No door
6) Purge exit.
Enter choice, 0 to quit :
```

2.2.3 And Now to Wonderland

Alice had never seen a rabbit with a waistcoat before, nor one with a watch to take out of his pocket. She jumped to her feet and ran after him, just in time to see him pop down a large rabbit hole under the hedge. Alice followed him, never thinking how she was going to get out again.

The rabbit hole went on like a tunnel for some way, but suddenly Alice found herself falling down a very deep well. She was falling slowly, so she had lime to look around and see all sorts of interesting things along the way.

'I wonder how many thousands of miles I've fallen?' she thought.

'I must be somewhere near the centre of the earth. Perhaps I'll fall right through to the other side!'

But just then, thump, thump, thump, down she came on a heap of leaves, without hurting herself at all. Ahead of her, at the end of a long passage, she saw the White Rabbit hurrying along. 'Oh, my ears and whiskers, how late it's getting!' she heard him say as he turned the corner.

"Alice's Adventures in Wonderland"
Chapter 1, Down the Rabbit Hole,
Lewis Carroll (1865)

Well, we have created everything we need here, so it is time to actually enter our rabbit hole.

```
500H 100M 82V > enter hole
You start to fall - further and further, deeper and deeper into
the earth.  Surely, this is no ordinary rabbit hole.  You start
to wonder if it will ever stop, when you start to slow down,
and, thump, thump, thump, down you land on a heap of leaves,
without hurting yourself at all.

A Long Passage
   This is a rather long passageway, with a door at the
northern end, and a large heap of leaves at the other.
[ Exits: None!]

500H 100M 82V > l n
This door is very tiny.
The door is closed.

500H 100M 82V > l leaves
This large heap of leaves has a big indentation it the top.
```

And there we are. Of course, if you wanted to really make this a copy of Lewis Carroll's "Wonderland", there is still the Mad Hatter and his tea party to set up, not to forget Humpty Dumpty, the Queen of Hearts and the other.

2.3 The Lion, The Witch and The Wardrobe

"Narnia? What's that?" said Lucy.

"This is the land of Narnia," said the Faun, "where we are now; all that lies between the lamp-post and the great castle of Cair Paravel on the Eastern Sea. . . ."

—The Lion, the Witch and the Wardrobe

Let's leave Alice looking for the White Rabbit, and change authors. C. S. Lewis's "Narnia" books featured a wardrobe that was a gateway to a different land. Fans of the 1980's BBC comedy "The Young Ones" will recall this device was featured in episode 6 of the first series.

First, let's create the portal:

```
-- Item number : [3130]
1) Namelist : wardrobe
2) S-Desc   : a wooden wardrobe
3) L-Desc   :-
An old, two-door wardrobe is in the corner.
4) A-Desc   :-
As you step into the wardrobe, you feel strangely disoriented.
You blink, and think, "We're not in Kansas anymore, Toto."
5) Type        : PORTAL
```

```
6) Extra flags : NOBITS
7) Wear flags  : NOBITS
8) Weight      : 0
9) Cost        : 0
A) Cost/Day    : 0
B) Timer       : -1
C) Values      : 6760 15 3131 0
D) Applies menu
E) Extra descriptions menu
M) Min Level   : 0
P) Perm Affects: NOBITS
S) Script      : Not Set.
Q) Quit
Enter choice :
```

The container flag value of 15 means it is can be closed, is pickproof, loads closed and locked. Of course, a suitable key must be created too.

2.3.1 No Lions or Witches, But A Wardrobe

Let's try it out

```
36H 118M 87V > look
Papi's Cozy Cottage
   You are inside Papi's living room, a light room at around
three by four meters in size.  The nicely carved oaken front
door is set in the middle of the western wall, just between two
windows.  A small fire crackles in the fireplace, which is set
against the northern wall.
[ Exits: None!]
An old, tarnished key rests here.
An old, two-door wardrobe is in the corner.
A cup has been set here.
Maclir the Implementor is standing here.

36H 118M 87V > l wardrobe
This old wooden wardrobe has seen a lot of use.  The handles
have been polished smooth by decades of hands opening and
closing the doors.  It could be worth a tidy sum, if the right
buyer could be found.

36H 118M 87V > open wardrobe
It seems to be locked.

36H 118M 87V > take key
You get a tarnished key.

36H 118M 87V > l key
This small key appears quite old.  The head is of an
interesting design.

36H 118M 87V > l design
The letters "CSL" have been worked into an interlocking design.

36H 118M 87V > unlock wardrobe
*Click*
```

2.3.2 This is Not Your Father's Wardrobe

Great – this looks like a neat object. Can I take it or use it?

```
36H 118M 87V > take wardrobe
A wooden wardrobe: you can't take that!

36H 118M 87V > inv
You are carrying:
a tarnished key
a canteen

36H 118M 87V > put canteen in wardrobe
You put a canteen in a wooden wardrobe.
What? A canteen disappears from a wooden wardrobe in a puff of
smoke!
```

Well, not only can we not take this valuable antique, I have lost my canteen.

```
36H 118M 87V > enter wardrobe
As you step into the wardrobe, you feel strangely disoriented.
You blink, and think, "We're not in Kansas anymore, Toto.  "

The Abandoned Beacon
   You are standing at the top of the hill, affording views
from the Shire to the west all the way to Midgaard in the east.
The hilltop itself has been leveled and paved with flagstones,
while eight large rectangular stones have been raised in a
circular pattern around the edge of the paved area.

   The center of the paved area is charred and blackened,
suggesting that this was once a beacon, where fires could be
lit to send messages between the Shire and Midgaard.
[ Exits: s ]
A canteen has been set on the ground here.
```

Well, at least I car get my canteen back. Now, what did other people in the room see while all this happened?

```
500H 100M 82V >
Zapper gets a tarnished key.

500H 100M 82V >
Zapper unlocks a wooden wardrobe.

500H 100M 82V >
Zapper opens a wooden wardrobe.

500H 100M 82V >
Zapper puts a canteen in a wooden wardrobe.
What? A canteen disappears from a wooden wardrobe in a puff of
smoke!

500H 100M 82V >
Zapper enters a wooden wardrobe.
```

That was fun, wasn't it. Now, how did we extend the standard container behavior to include portals, and why did the canteen disappear?

2.3.3 Adding Container Behavior to Portals

The container behaviors that we want to incorporate are those of open, close, lock and unlock (which is implemented in the do_gen_door routine in act.movement.c. All we need to change to make is to extend the DOOR_IS_OPENABLE definition, and then all of the tests in do_gen_door will work for portals with the "CLOSE" bit set.

```
#define DOOR_IS_OPENABLE(ch, obj, door) ((obj) ? \
                 ((GET_OBJ_TYPE(obj) == ITEM_CONTAINER) && \
                   OBJVAL_FLAGGED(obj, CONT_CLOSEABLE)) ||   \
                 ((GET_OBJ_TYPE(obj) == ITEM_PORTAL) &&    \
                   OBJVAL_FLAGGED(obj, CONT_CLOSEABLE))     :\
                 (EXIT_FLAGGED(EXIT(ch, door), EX_ISDOOR)))
```

Now that we can open, shut, lock and unlock our container, we need to be able to use it as a container. The "put" and "get" commands are implemented in do_put and do_get respectively in act.item.c. Both of these have test to see if the object things are being put in or taken out of is actually a container. We add another element to those tests in both routines:

```
else if ((GET_OBJ_TYPE(cont) != ITEM_CONTAINER) &&
        .(GET_OBJ_TYPE(cont) != ITEM_PORTAL) )
  act("$p is not a container.", FALSE, ch, cont, 0, TO_CHAR);
```

Once we have determined we can actually put something in our portal, the routine perform_put does the actual "putting". This checks things like weight limits on the container, whether the item being disposed of is "nodrop", and things like that. Finally, we actually drop it in the container.

One change that we have to make is checking the weight limit of the container. A normal container has a weight limit, and you cannot put something in a container that will make the total weight of the contents exceed the capacity of the container. Now, the weight limit of a container is defined in the first object value; but for portals, this is used to store the destination location. So, we need to make this check only it the container object is a true container, not a portal. After all, this does seem reasonable, since the object being placed in there never really stays in the portal container. Here is the modified code:

```
  if ((GET_OBJ_TYPE(cont) == ITEM_CONTAINER) &&
      (GET_OBJ_VAL(cont, 0) > 0) &&
      (GET_OBJ_WEIGHT(cont) + GET_OBJ_WEIGHT(obj) >
                              GET_OBJ_VAL(cont, 0)))
    act("$p won't fit in $P.", FALSE, ch, obj, cont, TO_CHAR);
  else if (OBJ_FLAGGED(obj, ITEM_NODROP) &&
           IN_ROOM(cont) != NOWHERE)
    act("You can't get $p out of your hand.", FALSE, ch, obj,
         NULL, TO_CHAR);
  else {
    /* code to take the object from the character and place
       in the container. */
  }
```

Now, this doesn't allow us to check for putting huge things in a tiny portal, but considering that a portal breaks the laws of physics anyway, the fact you can put an elephant in something the size of a box of matches is small beer.

Once the object has been placed in the container the unique portal behavior takes over, and we move the object to the portal's destination, and tell people in the room what just happened.

```
if (GET_OBJ_TYPE(cont) == ITEM_PORTAL) {
      obj_from_obj(obj);
      obj_to_room(obj, real_room(GET_OBJ_VAL(cont, 0)));
      act("What? $U$p disappears from $P in a puff of smoke!",
          TRUE, ch, obj, cont, TO_ROOM);
      act("What? $U$p disappears from $P in a puff of smoke!",
          FALSE, ch, obj, cont, TO_CHAR);
    }
```

2.3.4 Loading Items in a Portal Container

Now, you may ask, since items placed in the portal never actually stay there, why did we bother to add the portal test to do_get? Well, we could have things designed that some objects load in our portal on zone load and reset. For example, consider the following:

```
Room number: 3171               Room zone: 31
Z) Zone name      : South Midgaard
L) Lifespan       : 40 minutes
B) Bottom of zone : 3100
T) Top of zone    : 3199
R) Reset Mode     : Normal reset.
[Command list]
0 - Remove a wooden wardrobe [3130] from room.
1 - Load a wooden wardrobe [3130], Max : 1    % 0
2 -  then Put a fine silk vest [3063] in a wooden wardrobe
[3130], Max : 4    % 0
3 - Remove a tarnished key [3131] from room.
4 - Load a tarnished key [3131], Max : 1    % 0
5 - Remove a cup [3100] from room.
6 - Load a cup [3100], Max : 500    % 0
7 - Set door west as locked.
8 - <END OF LIST>
N) New command.
E) Edit a command.
D) Delete a command.
C) Change Percentage of command.
Q) Quit
Enter your choice :
```

Let's try it:

```
500H 100M 82V > take key
You get a tarnished key.

500H 100M 82V > unlock wardrobe
*Click*

500H 100M 82V > open wardrobe
Okay.
```

```

500H 100M 82V > look in wardrobe
wardrobe (here):
a fine silk vest

500H 100M 82V > take vest from wardrobe
You get a fine silk vest from a wooden wardrobe.
```

The wardrobe certainly appears to behave like any standard container, at least, until someone tries to put something in the container, or enters the portal. Nothing like being devious, and trying to make the players think.

2.4 *Is That a Portal in Your Pocket?*

2.4.1 Making The Portal Portable

Now, what if we want to create a portal that can be carried? Almost like a portable recall device, provided it will take you somewhere that you want to go.

```
-- Item number : [3094]
1) Namelist : loupe eyepiece
2) S-Desc   : a jeweler's loupe
3) L-Desc   :-
A well-crafted jeweler's loupe is here.
4) A-Desc   :-
As strange as it seems, you find yourself being drawn into the
loupe.  All around you, the world distorts into a swirling
whirlpool of light and dark.  Then, in a puff of smoke, you
emerge somewhere else.
5) Type        : PORTAL
6) Extra flags : MAGIC
7) Wear flags  : TAKE HOLD
8) Weight      : 1
9) Cost        : 10000
A) Cost/Day    : 500
B) Timer       : -1
C) Values      : 3091 0 -1 -1
D) Applies menu
E) Extra descriptions menu
M) Min Level   : 0
P) Perm Affects: NOBITS
S) Script      : Not Set.
Q) Quit
```

Notice the wear flags on this object, allowing it to be taken and held.

```
> look
The Pentagram Chamber
   You are in a large room with a five pointed star etched into
the floor.  A crackling energy field can be seen to the south,
it looks quite dangerous.
[ Exits: s ]
A well-crafted jeweler's loupe is here.
The Master Summoner is here holding gateways open to other
planes.

> take loupe
```

```
You get a jeweler's loupe.

> look at loupe
Designed to fit against an eye, this allows the user to closely
examine gems and precious stones.  There seems to be some fine
engraving around the rim.

> look at engraving
You look closely at the engraving on the eyepiece rim, and can
just make out the words "Home Sweet Home"

500H 100M 82V > enter loupe
As strange as it seems, you find yourself being drawn into the
loupe.  All around you, the world distorts into a swirling
whirlpool of light and dark.  Then, in a puff of smoke, you
emerge somewhere else.

The Silver Serpent
   Intricately worked jewelry sparkles from the many cases and
stands around the room.  A rainbow of gems, both cut and uncut,
blazes from under one of the magically guarded crystal-glass
displays.  Through a door in the back of the store can be seen
a complete lapidaries' work area, including polishing and
grinding tools as well as precious metals to fashion custom
jewelry from.
[ Exits: n ]
Jacob De Beers has the finest jewelry in all of Midgaard.
```
A useful device to have if things start to get unpleasant.

3 THE PORTAL SPELL

As well as the "static" portals as described above, magic users have the ability to create a temporary portal.

3.1 Defining the Spell

This ability is achieved at level 20, but altering the definition in class.c - by default, can change this, the line (around line 1703) in init_spell_levels is:

```
spell_level(SPELL_PORTAL, CLASS_MAGIC_USER, 20);
```

The spell is defined as a manual spell, in the routine mag_assign_spells in spell_parser.c (around line 947):

```
spello(SPELL_PORTAL, "portal", 75, 75, 0,
    TAR_CHAR_WORLD | TAR_NOT_SELF, FALSE, MAG_MANUAL, FALSE);
```

The actual spell is defined in spells.c (look for ASPELL(spell_portal)). This makes use of a portal template object, whose vnum is defined in config.c:

```
/* Portal Object */
/*
 * This should be the VNUM of an object you wish to use as a
 * template for portal spells when cast. If this object is not
 * in your obj files you will get a core dump when casting.
 */

obj_vnum portal_object = 2;
```

This object is in 0.obj:

```
#2
portal~
a glowing portal~
A glowing portal hovers here forming a gateway to another
place.~
~
28 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -1 -1
0 0 0 0
E
portal glowing~
You think you can see some shapes and movement inside the
portal.
~
```

The spell is cast at a target, and this opens a two way portal (actually, creating two portal objects) to the room where the target is located.

3.2 Casting the Portal Spell

Let’s see it in action. Our magic user wants to get to Zapper.

```
500H 100M 82V > cast 'portal' zapper
Okay.
You open a portal out of thin air.

```

```
500H 20M 82V > look in portal
After seconds of concentration you see the image of Upper
Warehouse Level.

500H 25M 82V >
```

3.3 What About the "Victim"?

Meanwhile, what has Zapper seen?

```
28H 112M 22V > u
Upper Warehouse Level
   Layers of dust and thick cobwebs cover everything - broken
crates, rotten casks, jumbles of ropes and cables.  A rusted
winch is next to an opening in the floor, the old winch rope
leading down into the darkness below.  To the south, shattered
double doors open to an old dock.  You think you hear the
scuttling of some small creatures.
[ Exits: n s d ]
A rat scurries among the trash.
A rat scurries among the trash.

28H 112M 21V >
A shimmering portal opens here for you.

28H 112M 21V > look portal
You think you can see some shapes and movement inside the
portal.
28H 112M 21V > look in portal
After seconds of concentration you see the image of The
Hayloft.
```

People at either end can enter the portal, even though it was designed for the "victim" of the spell. Even the caster can enter the portals, and anyone can go back and forth for as long as the portals stay in existence.

```
500H 35M 82V > enter portal
Upper Warehouse Level
   Layers of dust and thick cobwebs cover everything - broken
crates, rotten casks, jumbles of ropes and cables.  A rusted
winch is next to an opening in the floor, the old winch rope
leading down into the darkness below.  To the south, shattered
double doors open to an old dock.  You think you hear the
scuttling of some small creatures.
[ Exits: n s d ]
A glowing portal hovers here forming a gateway to another
place.
Zapper the Medium of Magic is standing here.
A rat scurries among the trash.
A rat scurries among the trash.
```

And now Zapper enters the portal:

```
36H 118M 65V > enter portal
The Hayloft
   This loft runs the length of the building above the stables,
and contains bales of hay, assorted bags of feed, and a
collection of old leatherworking tools.
```

```
[ Exits: d ]
A glowing portal hovers here forming a gateway to another
place.
A young boy is resting here, hoping his master won't find him.

36H 118M 65V >
```

What does Maclir see when Zapper does that?

```
500H 35M 82V >
Zapper enters a shimmering portal.

500H 45M 82V > enter portal
The Hayloft
   This loft runs the length of the building above the stables,
and contains bales of hay, assorted bags of feed, and a
collection of old leatherworking tools.
[ Exits: d ]
A glowing portal hovers here forming a gateway to another
place.
Zapper the Medium of Magic is standing here.
```

3.4 Implementing the Actual Spell

The ASPELL(spell_portal) routine in spells.c creates the actual portals, one where the caster is going to the "victim", and the other from the victim back to the room where the caster is:

```
ASPELL(spell_portal)
{
  struct obj_data *portal, *tportal;

  if (ch == NULL || victim == NULL)
    return;

  /* create the portal */
  portal = read_object(portal_object, VIRTUAL);
  GET_OBJ_VAL(portal, 0) = GET_ROOM_VNUM(IN_ROOM(victim));
  GET_OBJ_TIMER(portal) = (int) (GET_LEVEL(ch) / 2);
  obj_to_room(portal, IN_ROOM(ch));
  act("$n opens a portal in thin air.",
      TRUE, ch, 0, 0, TO_ROOM);
  act("You open a portal out of thin air.",
      TRUE, ch, 0, 0, TO_CHAR);
  /* create the portal at the other end */
  tportal = read_object(portal_object, VIRTUAL);
  GET_OBJ_VAL(tportal, 0) = GET_ROOM_VNUM(IN_ROOM(ch));
  GET_OBJ_TIMER(tportal) = (int) (GET_LEVEL(ch) / 2);
  obj_to_room(tportal, IN_ROOM(victim));
  act("A shimmering portal appears out of thin air.",
        TRUE, victim, 0, 0, TO_ROOM);
  act("A shimmering portal opens here for you.",
        TRUE, victim, 0, 0, TO_CHAR);
}
```

You can see that the only properties of the two objects are the first value, being the portal destination, and the duration, which depends on the level of the caster.

3.5 All Good Things Must Come to an End

Portals created by spells only last for a certain time. The routine point_update in limits.c is used to process various timers and counters. This is called from the heartbeat routine (comm.c), and uses the following logic to see when it should check timers:

```
if (!(heart_pulse % (SECS_PER_MUD_HOUR * PASSES_PER_SEC))) {
    weather_and_time(1);
    affect_update();
    point_update();
  }
```

One section goes through all objects in the world, and for portal objects, the following occurs:

```
if (GET_OBJ_TYPE(j) == ITEM_PORTAL) {
      if (GET_OBJ_TIMER(j) > 0)
        GET_OBJ_TIMER(j)--;

      if (!GET_OBJ_TIMER(j)) {
        act("A glowing portal fades from existence.",
          TRUE, world[j->in_room].people, j, 0, TO_ROOM);
        act("A glowing portal fades from existence.",
          TRUE, world[j->in_room].people, j, 0, TO_CHAR);
        extract_obj(j);
      }
    }
```

So, finally we can see when we create a portal using oedit, and when we load portals initially, we have to set the object timer to –1.

```
500H 100M 82V >
A glowing portal fades from existence.

500H 100M 82V > look
Upper Warehouse Level
   Layers of dust and thick cobwebs cover everything - broken
crates, rotten casks, jumbles of ropes and cables.  A rusted
winch is next to an opening in the floor, the old winch rope
leading down into the darkness below.  To the south, shattered
double doors open to an old dock.  You think you hear the
scuttling of some small creatures.
[ Exits: n s d ]
A rat scurries among the trash.
A rat scurries among the trash.
```

As we can see, our portal has now gone.

4 Object File Code

The various portal objects used as examples in this document are shown below.

4.1 Basic Portal (Section 1)

```
#3091
portal shimmering~
a shimmering portal~
Suspended in midair, and shimmering with a soft pink glow, a
portal leads to who knows where.~
As you step into the shimmering energy field, you feel a
momentary tingling over your body.  You think you heard a faint
sizzling, sparking noise, but you are not certain.
~
28 0 0 0 0 0 0 0 0 0 0 0 0
3170 0 -1 -1
0 0 0 0
E
portal shimmering~
   The portal appears to be a circular glowing area of energy,
infinitely thin, suspended somehow just above the floor.  It is
large enough for two people to step through (or into) it at the
same time.  You have absolutely no idea where it leads.
~
```

4.2 Disguised Portal (Section 2.1)

```
3092
mirror~
a tall mirror~
A tall mirror reflects the contents of the room.~
~
28 0 0 0 0 0 0 0 0 0 0 0 0
3161 0 -1 1
0 0 0 0
E
mirror~
The mirror is fixed in a tall wooden frame, with fancy gilt
flower patterns carved into the wood.
~
```

4.3 Hidden Portal (Section 2.2)

```
#3093
hole rabbit burrow~
a rabbit hole~
Undefined~
You start to fall - further and further, deeper and deeper into
the earth.  Surely, this is no ordinary rabbit hole.  You start
to wonder if it will ever stop, when you start to slow down,
and, thump, thump, thump, down you land on a heap of leaves,
without hurting yourself at all.
~
28 0 0 0 0 0 0 0 0 0 0 0 0
```

```
1207 0 -1 4
0 0 0 0
E
hole rabbit burrow~
This burrow appears to be slightly larger than your average
rabbit hole.
~
```

4.4 Container Portal (Section 2.3)

```
#3130
wardrobe~
a wooden wardrobe~
An old, two-door wardrobe is in the corner.~
As you step into the wardrobe, you feel strangely disoriented.
You blink, and think, "We're not in Kansas anymore, Toto."
~
28 0 0 0 0 0 0 0 0 0 0 0 0
6760 15 3131 0
0 0 0 0
E
wardrobe wooden~
This old wooden wardrobe has seen a lot of use.  The handles
have been polished smooth by decades of hands opening and
closing the doors.  It could be worth a tidy sum, if the right
buyer could be found.
~
#3131
key tarnished~
a tarnished key~
An old, tarnished key rests here.~
~
18 0 0 0 0 a 0 0 0 0 0 0 0
0 0 0 0
1 0 0 0
E
key~
This small key appears quite old.  The head is of an
interesting design.
~
E
head design~
The letters "CSL" have been worked into an interlocking
pattern.
~
```

4.5 Portable Portal (Section 2.4)

```
#3094
loupe eyepiece~
a jeweler's loupe~
A well-crafted jeweler's loupe is here.~
As strange as it seems, you find yourself being drawn into the
loupe.  All around you, the world distorts into a swirling
whirlpool of light and dark.  Then, in a puff of smoke, you
emerge somewhere else.
~
```

```
28 g 0 0 0 ao 0 0 0 0 0 0 0
3091 0 -1 2
1 15000 500 0
E
loupe eyepiece jeweler~
Designed to fit against an eye, this allows the user to closely
examine gems, jewelry and precious stones.  There seems to be
some fine engraving around the rim of the frame.
~
E
engraving rim~
You look closely at the engraving around the rim of the loupe.
You can just make out the words "Home Sweet Home"
~
A
4 2
```