THE SOURC$_{\mathrm{E}}$

ISSUE 2
Pizza Party, 2004

Cory Arcangel

☯ 💻 ;-)

2013

Arcangel Surfware

www.coryarcangel.com

```
                                                         .-
                                                       ,(   `-.
                                                     ,': `.    `.
                                                   ,` *   `-.   \
                                                  ,'   ` :+  = `.  `.
                                                ,-  (o):  .,    `.  `.
                                              ,'  ; :   ,(--) x;`.  ;
                                            ,'  :'  itz  ;  ; ; -,-'
                                          .'O ; = -' C ; ;'-,- ;
                                        ,;  -;   ` : ;'-,-'   i'
                                      ,` `;(_)  0 ; ','        :
                                    .';6     ; ' ,-'-
                                  ,' Q  ,& ;',-.'
                                ,( :` ; -,-'-  ;
                              ,-.`c -','
                            .';^-,-' -
                          ,' -;-''
                         ,,-
                         i'
                         :
 ----
/    \                              //\         PPPPP IIII ZZZZZ ZZZZZ    A
  u  u|      -------               // O \       PP  PP II    ZZ    ZZ    A A
    \ |  .-''#%&#&%#``-.          || O o  \     PPPPP  II   ZZ    ZZ    AAAAA
   = /  ((%&#&#&%&VK&%&))         ||__o__O__\   PP    IIII ZZZZZ ZZZZZ A     A
    |    `-._#%&##&%_.-'
 /\/\`--.    `-."".-'
 |  |    \   /`./
 |\/|  \  `-'  /
 || |   \     /            VK
```

Pizza Party (0.1 beta)

Cory Arcangel & Michael Frumin, 2004[1]

http://www.coryarcangel.com
http://www.frumin.net/ation/

Introduction
----

pizza_party is a command-line utility for ordering customized Domino's Pizza (tm) to your door.

Overview

This software is written for people who spend so much time at the command-line that they don't have time to pick up the phone and call Domino's, or even to go thru the many-step process of using Domino's web interface.

pizza_party comes with one executable, written in Perl, and a man page.

Installation

As pizza_party is written for command-line junkies, so too are these installation instructions.

```
1) presumably, you've already downloaded and extracted the .tar.gz file which
contained this README.

2) make sure you've got the following Perl modules (and all their dependencies)
installed:

      LWP::UserAgent;
      HTTP::Request;
      HTTP::Response;
      HTTP::Cookies;
      URI::Escape;
      Getopt::Mixed;

   if you don't know how to install Perl modules, you're clearly not hard enough to
be ordering pizza at the command line.

3) place the pizza_party executable (./pizza_party) somewhere in your path -- /usr/
local/bin should work fine.

4) place the man page (./man/pizza_party.1) in the usual man location -- /usr/local/
man/man1/ is typical.

5) go to http://www.dominos.quikorder.com/ and get an account

6) if you want to, make a .pizza_partyrc file in your home directory with your
username and password and, optionally, other default settings.  The structure of this
 file described in the man page.

7) order pizza, pay for it, tip your delivery guy well, and enjoy.

Licensing
_________

pizza_party, is distributed under the GPL.

See http://www.gnu.org/copyleft/gpl.html if you don't know what the GPL is.

                                                                 ---
                                                                 |  ----.
                                                                 |%=@%%/
                                                                 |o%%%/
                     --                                          |%%o/
               _,----- |     ..                                  |(_/
            ,/'   m%%%%|    /   `\.                              |o/
           /' m%%o(_)%|   /o%%m `\                               |/
         /' %%@=%o%%%o|   /(_)o%%%  `\
        /  %o%%%%%=@%%|  /%%o%%@=%%  \
       |  (_)%(_)%%o%%| /%%%=@(_)%%%  |
       | %%o%%%%o%%%(_|/%o%%o%%%%o%%% |
       | %%o%(_)%%%%%o%(_)%%%o%%o%o%% |
       |  (_)%%=@%(_)%o%o%%(_)%o(_)%  |
        \ -%%o%%%%%o%o%=@%%o%%@%%o%- /
         \. -o%%(_)%%%o%(_)%%(_)o- ,/
           \_ -o%=@%(_)%o%%(_)%- _/
             `\_--o%%%o%%%%%---_/'
                `--..----,,--'

"I always wanted to be a basketball player." - Ronnie James Dio
```

```perl
#!/usr/bin/perl

use strict;
use LWP::UserAgent;
use HTTP::Request;
use HTTP::Response;
use HTTP::Cookies;
use URI::Escape;
use Getopt::Mixed;

use constant FRONT_PAGE_URL => 'http://www.dominos.quikorder.com/scripts/mgwms32.dll?MGWLPN=MWEBLINK&wlapp=
    QUIKORDER&guid=';
use constant LOGIN_URL => 'http://www.dominos.quikorder.com/scripts/mgwms32.dll';
use constant ORDER_URL => 'http://www.dominos.quikorder.com/scripts/mgwms32.dll';
use constant CHECKOUT_URL => 'http://www.dominos.quikorder.com/scripts/mgwms32.dll';

use constant PPRINT_FATAL => -1;
use constant PPRINT_QUIET => 0;
use constant PPRINT_OUTPUT => 1;
use constant PPRINT_VERBOSE => 2;
use constant PPRINT_DEBUG => 3;

use constant MIN_QTY => 1;
use constant MAX_QTY => 5;

sub pprint;

our $PPERROR = "";

our $PPRINT_LEVEL = PPRINT_OUTPUT;
#our $PPRINT_LEVEL = PPRINT_DEBUG;

#dont fuck with the ordering of these toppings. it is keyed to the web page
our $TOPPINGS = [{ short => 'o', long => 'onions', type => '', default => '0', desc => 'with onions'},
        { short => 'g', long => 'green-peppers', type => '', default => '0', desc => 'with green peppers'},
        { short => 'm', long => 'mushrooms', type => '', default => '0', desc => 'with mushrooms'},
        { short => 'v', long => 'olives', type => '', default => '0', desc => 'with olives'},
        { short => 't', long => 'tomatoes', type => '', default => '0', desc => 'with tomatoes'},
        { short => 'h', long => 'pineapple', type => '', default => '0', desc => 'with pineapple'},
        { short => 'x', long => 'extra-cheese', type => '', default => '0', desc => 'extra-cheese'},
        { short => 'd', long => 'cheddar-cheese', type => '', default => '0', desc => 'with cheddar cheese'},
        { short => 'p', long => 'pepperoni', type => '', default => '0', desc => 'with pepperoni'},
        { short => 's', long => 'sausage', type => '', default => '0', desc => 'with sausage'},
        { short => 'w', long => 'ham', type => '', default => '0', desc => 'with ham (sWine, Wilbur)'},
        { short => 'b', long => 'bacon', type => '', default => '0', desc => 'with bacon'},
        { short => 'e', long => 'ground-beef', type => '', default => '0', desc => 'with ground beef'},
        { short => 'c', long => 'grilled-chicken', type => '', default => '0', desc => 'with grilled chicken'},
        { short => 'z', long => 'anchovies', type => '', default => '0', desc => 'with anchovieZZZZ'},
        { short => 'u', long => 'extra-sauce', type => '', default => '0', desc => 'with extra sauce '},
  ];

### non-topping options
our $OPTIONS = [
        { short => 'U', long => 'username', type => ':s', default => undef, desc => 'user name'},
        { short => 'P', long => 'password', type => ':s', default => undef, desc => 'password'},
#MAYBE CAP I
        { short => 'I', long => 'input-file', type => '=s', default => '', desc => 'input file to read batch of
            pizza (see man page)'},
        { short => 'V', long => 'verbose', type => '', default => '', desc => 'verbose'},
        { short => 'Q', long => 'quiet', type => '', default => '', desc => 'quiet'},
        { short => 'F', long => 'force', type => '', default => '', desc => 'don\'t ask for confirmation before
            ordering'},
        { short => 'H', long => 'help', type => '', default => '', desc => 'show the command options/arguments
            for pizza_party'},
        ];

#combine the toppings list into the commandline args
foreach(@$TOPPINGS) {
  push(@$OPTIONS, $_);
}

#get list of the long option names
our (@TOPLIST) = map {$_->{long};} @$TOPPINGS;
```

```

#possible sizes
our $SIZES = {
  s => { name => 'small', aliases => ['s', 'small']},
  m => { name => 'medium', aliases => ['m', 'med', 'medium']},
  l => { name => 'large', aliases => ['l', 'large']},
};

#possible crusts
our $CRUSTS = {
  t => { name => 'thin', aliases => ['t', 'thin']},
  r => { name => 'regular', aliases => ['r', 'reg', 'regular']},
  d => { name => 'deep', aliases => ['d', 'deep']},
};

#default default is 1 Large Regular [crust]
our (@DEFAULTS) = qw(1 l r);

my $rc_file = "$ENV{HOME}/.pizza_partyrc";
my $rc = readRCFile($rc_file);

$DEFAULTS[0] = $rc->{quantity} unless !defined($rc->{quantity});
$DEFAULTS[1] = $rc->{size} unless !defined($rc->{size});
$DEFAULTS[2] = $rc->{crust} unless !defined($rc->{crust});

our $optHash = hashifyOptions($OPTIONS);
markToppings($optHash, \@TOPLIST);

my $args = getOrder(\@ARGV, \@DEFAULTS, $OPTIONS, \&asshole);
if($args->{help}) {
  printUsage();
  print "\n";
  printHelp();
  pprint "\n", PPRINT_FATAL;
}


my (@orders);

if(!$args->{I}) {
  my $order = \%{$args};
  push(@orders, prepareOrder($order));
}
else {
  push(@orders, parseBatchFile($args->{I}));
}

if(scalar(@orders) < 1) {
  pprint "You have not specified any orders.\n\n", PPRINT_FATAL;
}

my ($user, $pass) = getAccountInfo($args, $rc);

setOutputLevel($args);

pprint "Order:\t" . join("\n\t", map { orderStr($_) } @orders) . "\n\n", PPRINT_OUTPUT;


my $ua = LWP::UserAgent->new;
$ua->agent("PizzaParty");

my ($formData, $cookies) = getLoginPage($ua, LOGIN_URL);
($formData, $cookies) = login($user, $pass, $formData, $cookies, $ua, ORDER_URL);
#foreach order, do something
my $price = 0;
for(my $i = 0; $i < scalar(@orders); $i++) {
  ($formData, $cookies, $price) = order($orders[$i], $i + 1, ($i < scalar(@orders) - 1 ? 0 : 1), $formData,
      $cookies, $ua, CHECKOUT_URL);
}
my $o = $orders[0];
$o->{price} = $price;

if($args->{F} || confirmOrder($o)) {
  ($formData, $cookies) = checkout($o, $formData, $cookies, $ua);
}

###################### WEB REQUESTING FUNCTIONS ###############

sub getLoginPage {
  my ($ua, $nextURL) = @_;

  pprint "Getting login page...\n", PPRINT_OUTPUT;

  my $req = nextReq("GET", FRONT_PAGE_URL, {}, "");

  my $res = $ua->request($req);

  printPage($res->content, "loginpage");

  if ($res->is_success) {
    pprint "LOGIN PAGE REQUEST SUCCESS\n", PPRINT_DEBUG;

    my $hid = parseHiddens($res->content);
    my $cooks = mergeCookies($req, $res, $nextURL, {});

    pprint "LOGIN PAGE RESPONSE HIDDENS: " . join(" ; ", map {"$_ : $hid->{$_}"} keys %$hid) . "\n",
        PPRINT_DEBUG;
    pprint "LOGIN PAGE RESPONSE COOKIES: $cooks\n", PPRINT_DEBUG;

    return ($hid, $cooks);

  }
  else {
    pprint "LOGIN PAGE REQUEST FAILURE\n", PPRINT_DEBUG;
    return ({}, "");
  }

}

sub login {
  my ($user, $pass, $hiddens, $cooks, $ua, $nextURL) = @_;

  pprint "Logging in as $user...\n", PPRINT_OUTPUT;


  my $data = {
    B2 => "Let's Go",
    UNAME => $user,
    UPWORD => $pass,
  };

  my $req = nextReq("POST", LOGIN_URL, mergeHashes($hiddens,$data), $cooks);

  my $res = $ua->request($req);

  printPage($res->content, "loginresp");

  if ($res->is_success) {
    pprint "LOGIN REQUEST SUCCESS\n", PPRINT_DEBUG;

    if(storeClosed($res->content)) {
      pprint "\nSorry, but your dominos is currently unavailable for ordering over the internet.\n\n" ,
          PPRINT_FATAL;
    }

    my $hid = parseHiddens($res->content);
    my $cooks = mergeCookies($req, $res, $nextURL, {});

    pprint "LOGIN RESPONSE FORM VARIABLES: " . join(" ; ", map {"$_ : $hid->{$_}"} keys %$hid) . "\n",
        PPRINT_DEBUG;
    pprint "LOGIN RESPONSE COOKIES: $cooks\n", PPRINT_DEBUG;

    return ($hid, $cooks);

  }
```

```perl
  else {
    pprint "LOGIN REQUEST FAILURE\n", PPRINT_DEBUG;

    return ({},"");

  }
}


sub order {
  my ($order, $nO, $final, $hiddens, $cooks, $ua, $nextURL) = @_;

  pprint "Submitting order for " . orderStr($order) . " ...\n", PPRINT_OUTPUT;


  my $data = {
    "PQTY$nO" => $order->{qty},
    "PTYP$nO" => pizzaSelect($order),
  };

  if($final) {
    $data->{B3} = 'Checkout';
  }
  else {
    $data->{B1} = 'Update Order';
  }


  for(my $i = 1; $i < $nO; $i++) {
    $data->{"DELPI$i"} = "ON";
  }

#no toppings
  for(my $i = 1; $i <= 16; $i++) {
    $data->{"PINGI${nO}T$i"} = "01;";
  }

#no side orders
  for(my $i = 1; $i <= 30; $i++) {
    $data->{"POTHI$i"} = "0";
  }

#just the toppings i want
  foreach(@{$order->{toppings}}) {
    $data->{"PINGI${nO}T" . $optHash->{$_}->{topping_id}} = "11;";
  }

  my $req = nextReq("POST", ORDER_URL, mergeHashes($hiddens, $data), $cooks);

  my $res = $ua->request($req);

  printPage($res->content, "orderresp$nO");

  if ($res->is_success) {
    pprint "ORDER REQUEST SUCCESS\n", PPRINT_DEBUG;

    my $hid = parseHiddens($res->content);
    my $cooks = mergeCookies($req, $res, $nextURL, {});

    $_ = $res->content;
    my (@prices) = /(\$[\d\.]+)/igs;

    my $price = $prices[-1];

    pprint "ORDER RESPONSE FORM VARIABLES: " . join(" ; ", map {"$_ : $hid->{$_}"} keys %$hid) . "\n",
        PPRINT_DEBUG;
    pprint "ORDER RESPONSE COOKIES: $cooks\n", PPRINT_DEBUG;

    return ($hid, $cooks, $price);

  }
  else {
    pprint "ORDER REQUEST FAILURE\n", PPRINT_DEBUG;
    return ({},"", 0);

  }
}


sub confirmOrder {
  my $o = shift;
  print "Confirmation: order for $o->{price} (y|yes|n|no)? ";

  while(<STDIN>) {
    chomp($_);
    $_ = lc($_);
    if($_ eq 'y' || $_ eq 'yes') {
      return 1;
    }
    elsif($_ eq 'n' || $_ eq 'no') {
      return 0;
    }
    print "Please type 'y' or 'yes' or 'n' or 'no': ";
  }

}


sub checkout {
  my ($order, $hiddens, $cooks, $ua, $nextURL) = @_;

  pprint "Checking out for your order of $order->{price}...\n", PPRINT_OUTPUT;

#TESTING!! return ;

  my $data = {
    DELIVER => "Delivery",
    PAY => "Cash",
    PAOK => "ON",
    B1 => 'Submit Order',
  };

  my $req = nextReq("POST", CHECKOUT_URL, mergeHashes($hiddens, $data), $cooks);

  my $res = $ua->request($req);

  printPage($res->content, "checkoutresp");

  if ($res->is_success) {
    pprint "CHECKOUT REQUEST SUCCESS\n", PPRINT_DEBUG;

    $_ = $res->content;

    my ($statLink) = /href\=\"([^\s\"]+qordstat[^\s\"]+)\"/is;
    if(defined($statLink) && $statLink) {
      $statLink = "http://www.dominos.quikorder.com$statLink";
      pprint "Checkout successful! To view the real-time status of your order, please go to:\n\t$statLink\n",
          PPRINT_OUTPUT;
    }

  }
  else {
    pprint "CHECKOUT REQUEST FAILURE\n", PPRINT_DEBUG;

    return ({},"");

  }
}


sub storeClosed {
  $_ = shift;

  return /sorry.*store.*unavailable/is || /sorry.*not.*taking.*orders/is;

```

```
}

########################## OPTION/ORDER HANDLING ############################

sub prepareOrder {
  my $order = shift;

  $order->{toppings} = getToppings($order, \@TOPLIST);
  checkOrder($order);

  return $order;
}




sub parseBatchFile {
  my $fname = shift;

  pprint "OPENING BATCH FILE: $fname\n", PPRINT_DEBUG;

  (open BATCH, "< $fname") || pprint("Couldn't open batch file $fname for reading\n\n", PPRINT_FATAL);

  my (@orders);

  my $LINE = 0;

  while(<BATCH>) {
    $LINE++;

    chomp($_);

    if(/^\#/ || /^\s*$/) {
      next;
    }

    my (@lp) = split(/\s/);

#   unshift(@lp, "");

#   print "READ LINE: $_, parts: " . join(" ; ", @lp) . "\n";

    my $order = getOrder(\@lp, \@DEFAULTS, $OPTIONS, sub { batcherr(@_, $LINE, $fname); });

    push(@orders, prepareOrder($order));

  }

  if(scalar(@orders) < 1) {
    pprint qq{Your batch file "$fname" does not contain any valid orders\n}, PPRINT_OUTPUT;
  }

  return (@orders);
}


sub readRCFile {
  my $fname = shift;


  my $rc = {};
  if(! -e $fname) {
    return $rc;
  }

  if(open INIFILE, "< $fname") {
    my $ln = 0;
    while(<INIFILE>) {
      $ln++;
      chomp($_);
      my $line = $_;

      next if($line =~ /^\s*\#/i || $line =~ /^\s*$/i);
      my ($k, $v) = split(/\=/, $line, 2);
      if(defined($k) && defined($v)) {
        $k =~ s/^\s+//ig;
        $v =~ s/^\s+//ig;
        $k =~ s/\s+$//ig;
        $v =~ s/\s+$//ig;

        $k =~ s/^default\_//ig;

        $rc->{lc($k)} = $v;
      }
      else {
        pprint "Line $ln of your rc file is malformed: \"$line\"\n", PPRINT_OUTPUT;
      }
    }

    (!defined($rc->{quantity}))
      || ($rc->{quantity} = parseQty($rc->{quantity}))
      || pprint "'default_quantity' invalid in rc file: $PPERROR\n", PPRINT_OUTPUT;

    (!defined($rc->{size}))
      || ($rc->{size} = parseSize($rc->{size}))
      || pprint "'default_size' invalid in rc file: $PPERROR\n", PPRINT_OUTPUT;

    (!defined($rc->{crust}))
      || ($rc->{crust} = parseCrust($rc->{crust}))
      || pprint "'default_crust' invalid in rc file: $PPERROR\n", PPRINT_OUTPUT;
  }
  else {
    pprint "NO INI FILE FOUND\n", PPRINT_DEBUG;
  }

  return $rc;

}



sub setOutputLevel {
  my $options = shift;

  if($options->{quiet}) {
    $PPRINT_LEVEL = PPRINT_QUIET;
  }
  elsif($options->{verbose}) {
    $PPRINT_LEVEL = PPRINT_VERBOSE;
  }
}



sub getAccountInfo {
  my $options = shift;
  my $rc = shift;

  #read from file....
  my $user = $rc->{username} || "";
  my $pass = $rc->{password} || "";

  if(defined($options->{username}) && $options->{username}) {
    $user = $options->{username};
  }
  if(defined($options->{password}) && $options->{password}) {
    $pass = $options->{password};
  }

  if(!($user && $pass)) {
#   pprint "You haven't entered a username and password\n", PPRINT_FATAL;
    argError ("You haven't entered a username and password");
  }

  return ($user, $pass);
```

```
}



sub checkOrder {
  my $o = shift;

  if($o->{size} eq 's' && $o->{crust} eq 'd') {
    pprint "You cannot order Small pizzas with a Deep Dish crust.\n", PPRINT_FATAL;

  }

}

sub pizzaSelect {
  my $o = shift;

  my $s = $o->{size};
  my $c = $o->{crust};

  if($s eq 'l' || $s eq 'm') {
    return join(" ", (ucfirst($SIZES->{$s}->{name}), ucfirst($CRUSTS->{$c}->{name}), ($c eq 'd' ? "Dish" : "
        Crust")));
  }
  else {
    return ucfirst($SIZES->{$s}->{name}) . " " . ($c eq 'r' ? "Reg Crust" : "Thin");
  }

}

sub orderStr {
  my $order = shift;

  return "$order->{qty} $SIZES->{$order->{size}}->{name} $CRUSTS->{$order->{crust}}->{name} pizza"
  . ($order->{qty} > 1 ? "s" : "")
  . (scalar(@{$order->{toppings}}) > 0 ? " with " . join(", ", @{$order->{toppings}}) : "" )
  . ($order->{comment} ? " ($order->{comment})" : "")
  ;
}

sub markToppings {
  my ($optHash, $toppings) = @_;

  for(my $i = 0; $i < scalar(@$toppings); $i++) {
    my $id = $i+1;
    my $t = $toppings->[$i];
    $optHash->{$t}->{topping_id} = $id;
  }
}

sub hashifyOptions {
  my $optHash = {};

  map {$optHash->{$_->{long}} = $_;
    $optHash->{$_->{short}} = $_;
    } @$OPTIONS;

  return $optHash;
}

sub getToppings {
  my ($order, $toppings) = @_;

  my (@mytops) = map { (exists($order->{$_}) && $order->{$_} ? $_ : () ) } @$toppings;

  return \@mytops;
}



sub getOrder {
  my ($args, $DEFAULTS, $OPTIONS, $errfn) = @_;

  my (@tempARGV) = (@ARGV);
  (@ARGV) = @$args;

  my $res = {};
  map {$res->{$_->{long}} = $_->{default};
    $res->{$_->{short}} = $_->{default};
   } @$OPTIONS;


  my $optStr = join(" ", map {"$_->{short}$_->{type}" . ($_->{long} ? " $_->{long}>$_->{short}" : "")} @$OPTIONS
      );

  Getopt::Mixed::init($optStr);
  $Getopt::Mixed::badOption = $errfn;
  while (my ($opt, $val, $pretty) = Getopt::Mixed::nextOption()) {
    $opt = $optHash->{$opt};
    if($opt->{type} =~ /s/i) {
      $val = lc($val);
    }
    elsif(!$opt->{type}) {
      $val = 1;
    }
    $res->{$opt->{long}} = $val;
    $res->{$opt->{short}} = $val;
  }

  Getopt::Mixed::cleanup();

  $res->{qty} = parseQty(shift(@ARGV) || $DEFAULTS->[0]) || argError($PPERROR);
  $res->{size} = parseSize(shift(@ARGV) || $DEFAULTS->[1]) || argError($PPERROR);
  $res->{crust} = parseCrust(shift(@ARGV) || $DEFAULTS->[2]) || argError($PPERROR);

  $res->{comment} = join(" ", @ARGV);


  (@ARGV) = @tempARGV;

  return $res;
}

sub parseQty {
  my $q = shift;

  if($q !~ /^\d+$/i) {
    $PPERROR = "quantity '$q' must be a numeric integer";
    return undef;
  }
  elsif(!($q >= MIN_QTY && $q <= MAX_QTY)) {
    $PPERROR = "quantity $q must be between " . MIN_QTY . " and " . MAX_QTY . " pizzas";
    return undef;
  }

  return $q;
}

sub parseSize {
  my $s = shift;

  my $os = $s;
  $s = matchArg(lc($s), $SIZES);
  if(!defined($s)) {
    $PPERROR = "size '$os' was invalid";
    return undef;
  }

  return $s;
}

sub parseCrust {
  my $c = shift;
  my $oc = $c;
  $c = matchArg(lc($c), $CRUSTS);
  if(!defined($c)) {
    $PPERROR = "crust '$oc' was invalid";
    return undef;
  }

  return $c;
}

sub matchArg {
  my ($a, $vals) = @_;

  foreach my $k (keys %$vals) {
    my $v = $vals->{$k};
    foreach my $v2 (@{$v->{aliases}}) {

      if($a eq $v2) {
        return $k;
      }
    }
  }

  return undef;
}

sub printArg {
  my ($name, $vals,) = @_;

  while(my ($k, $v) = each(%$vals)) {
    foreach my $v2 (@{$v->{aliases}}) {
      print "$name: $k, $v2\n";
    }
  }

}


sub asshole {
  my ($pos, $arg) = @_;
  argError("Argument $arg was invalid");
}

sub batcherr {
  my ($pos, $arg, $line, $file) = @_;
  argError("Error on line $line of batch file: argument $arg was invalid.");
}



sub argError {
  my $s = shift;

  print "$s\n";
  printUsage();
  pprint "Try `pizza_party --help' for more information.\n\n", PPRINT_FATAL;

}

sub printUsage() {
  print "Usage: pizza_party [OPTIONS] [QUANTITY=1] [SIZE=large] [CRUST=regular]\n";
}

sub printHelp {
  print MIN_QTY . " <= QUANTITY <= " . MAX_QTY . ". Default is $DEFAULTS[0].\n";
  print "SIZE can be: (small|s) or (medium|med|m) or (large|l). Default is large.\n";
  print "CRUST can be: (thin|t) or (regular|reg|r) or (deep|d). Default is regular.\n";
  print "Example: `pizza_party -pmx 2 medium regular` orders 2 medium regular crust pizzas\n with pepperoni,
      mushrooms, and extra-cheese, right to your door!\n";
  print "\n";

  map {print "[-$_->{short} " . ($_->{long} ? "| --$_->{long}" : "") . "] " . (strMult(17 - length($_->{long}),
      " ")) . "$_->{desc}\n"} @$OPTIONS;
  print "\n";
  print "See the man page for more details on accounts, confiuration files, and batch ordering.\n";

}

sub strMult {
  my $n = shift;
  my $s = shift;

  my $r = '';
  while($n-- > 0) {
    $r .= $s;
  }
  return $r;
}



############ HELPER ############################

sub pprint {
  my ($s, $level) = @_;

  if($level <= $PPRINT_LEVEL) {
    print $s;
    if($level == PPRINT_FATAL) {
      exit;
    }
  }
}


sub printPage {
  my ($html, $name) = @_;

  if($PPRINT_LEVEL < PPRINT_DEBUG) {
    return;
  }

  my $fname = "$name.html";

  open OUTFILE, "> pages/$fname" || die "couldn't open $fname for writing";

  print OUTFILE $html;

  close OUTFILE;
}


############## WEB ORDERING HELPER #########################################################

sub nextReq {
  my ($type, $url, $data, $cooks) = @_;

  my $req = HTTP::Request->new($type => $url);

  set_post_data($req, $data);
  set_cookie_data($req, $cooks);

  return $req;
}



sub mergeHashes {
  my $res;

  foreach my $h (@_) {
    map {$res->{$_} = $h->{$_}; } keys %$h;
  }
  return $res;
}
```

```
sub mergeCookies {
  my ($oldReq, $resp, $newReq, $extra) = @_;

  my $cjar = new HTTP::Cookies(
              file => "",
              autosave => 0,
              hide_cookie2=> 1,
              ignore_discard => 1
              );


  my $reqC = parseCookies($oldReq->header('Cookie') || "");

  if(!ref($newReq)) {
    my $t = $newReq;
    $newReq = new HTTP::Request();
    $newReq->uri($t);
  }


  my $host = $newReq->uri()->host();
  $host =~ s/^[^\.]*(\..*)$/$1/is;
  my $port = $newReq->uri()->port();

  while(my ($k, $v) = each(%$reqC)) {
    $cjar->set_cookie(0, "$k", $v, "/", $host, $port, 0, 0, 10000, 0);
  }

  $cjar->extract_cookies($resp);

  if(!defined($extra)) {
    $extra = {};
  }
  elsif(!ref($extra)) {
    $extra = parseCookies($extra);
  }

  while(my ($k, $v) = each(%$extra)) {
    $cjar->set_cookie(0, $k, $v, "/", $host, $port, 0, 0, 10000, 0);
  }

  # if they just pass the uri, we need an HTTP::Request object
  # for the cookie jar to work on.
  if(!ref($newReq)) {
    my $t = $newReq;
    $newReq = new HTTP::Request();
    $newReq->uri($t);
  }
  $cjar->add_cookie_header($newReq);


  return ($newReq->header('Cookie') || "");
}


sub parseCookies {
  my $s = shift;

  if (ref($s) =~ /HASH/) {
return $s;
  }

  my $r = {};

  $_ = $s;

  my (@pairs) = split(/\;/);

  foreach(@pairs) {
    my ($k, $v) = split(/\=/);
    $k =~ s/^\s*//igs;
    $r->{$k} = $v;
  }
  return $r;

}

sub parseHiddens {
  $_ = shift;

  my (@in) = m|<input[^>]+>|igs;

  my $res = {};

  map {(lc(getAtt($_, "type")) eq 'hidden' ? $res->{getAtt($_, "name")} = getAtt($_, "value") : "");} @in;

  return $res;
}


sub getAtt {
  my ($html, $att) = @_;

  my $var = '(?:\"[^\"]*\"|\'[^\']*\'|[^\'\"][^\s\>]*)';


  $_ = $html;
  my ($v) = /$att\=($var)/is;
  $v =~ s/^[\'\"](.*)[\'\"]$/$1/is;

  return $v;

}

sub set_post_data {
  my ($self, $data) = @_;

  if (defined($data) && $data) {
    my (@c, $content, $len);

      if (ref($data) =~ /HASH/) {
        while (my($k, $v) = each(%$data)) {
          push(@c, "$k=" . uri_escape($v));
        }

      $content = join('&', @c);
      } elsif (ref($data) =~ /ARRAY/) {
      for (my $i = 0; $i < scalar(@$data); $i += 2) {
        push(@c, "$data->[$i]=" . uri_escape($data->[$i + 1]));
      }

      $content = join('&', @c);
    } else {
        $content = $data;
      }

      $len = length($content);

      if($len > 0) {
        $self->method('POST');
        $self->content($content);
        $self->content_type("application/x-www-form-urlencoded");
        $self->content_length($len);
      }
  }
}

sub set_cookie_data {
  my ($self, $data) = @_;

  if(defined($data) && $data) {
    $self->header('Cookie' => $data);
  }

}
```

```
.TH pizza_party 1 "April 29th 2004" "Pizza Party"
.SH NAME
pizza_party \- text-based client for ordering pizza.

.SH SYNOPSIS
.B pizza_party
.RB [ \-o | \-\-onions ]
.RB [ \-g | \-\-green-peppers ]
.RB [ \-m | \-\-mushrooms ]
.RB [ \-v | \-\-olives ]
.RB [ \-t | \-\-tomatoes ]
.RB [ \-h | \-\-pineapple ]
.RB [ \-x | \-\-extra-cheese ]
.RB [ \-d | \-\-cheddar-cheese ]
.RB [ \-p | \-\-pepperoni ]
.RB [ \-s | \-\-sausage ]
.RB [ \-w | \-\-ham ]
.RB [ \-b | \-\-bacon ]
.RB [ \-e | \-\-ground-beef ]
.RB [ \-c | \-\-grilled-chicken ]
.RB [ \-z | \-\-anchovies ]
.RB [ \-u | \-\-extra-sauce ]
.RB [ \-U| \-\-user=
.IR username ]
.RB [ \-P | \-\-password=
.IR pasword ]
.RB [ \-I | \-\-input\-file=
.IR input-file ]
.RB [ \-V | \-\-verbose ]
.RB [ \-Q | \-\-quiet ]
.RB [ \-F | \-\-force ]
.RB [ QUANTITY ]
.RB [ SIZE ]
.RB [ CRUST ]

.SH DESCRIPTION
The
.B pizza_party
program provides a text only command line interface for ordering DOMINOS pizza from the
terminal. This program is intended to aid in the throwing of
.IR PIZZA
.IR PARTIES
which are also sometimes known as
.IR ZA
.IR PARTIES

.SH USAGE
.TP
pizza_party -pmx 2 medium regular
.TP
Orders 2 medium regular crust pizzas with pepperoni, mushrooms, and extra-cheese.

.SH DIRECTIONS
You will first need to go to www.dominos.quikorder.com and sign up for an account. Then
use your user name and password in the .pizza_partyrc file, or in your command line
request.

.SH OPTIONS
```

```
.TP

.BR \-o | \-\-onions
Order your pizza with onions.
.TP
.BR \-m | \-\-mushrooms
Order your pizza with mushrooms.
.TP
.BR \-v | \-\-olives
Order your pizza with olives. To remember this try to think that often olives go in [v]
odka.
.TP
.BR \-t | \-\-tomatoess
Order your pizza with tomatoes.
.TP
.BR \-h | \-\-pineapple
Order your pizza with pineapple. To remember this try to think of [h]awaii
.TP
.BR \-x | \-\-extra-cheese
Order your pizza with extra cheese. To remember this try to think of [x]tra, a common
misspelling of extra.
.TP
.BR \-d | \-\-cheddar-cheese
Order your pizza with cheddar-cheese.
.TP
.BR \-p | \-\-pepperoni
Order your pizza with pepperoni.
.TP
.BR \-s | \-\-sausage
Order your pizza with sausage.
.TP
.BR \-w | \-\-ham
Order your pizza with ham. To remember this try to think of the pig from chroltette's web
named [w]ilber. Or actually better yet, lets think of "House of Pains" second album title
"Same as it ever [w]as". fact: House of Pain started a pizza place in LA names HOUSE OF
PIZZA.
.TP
.BR \-b | \-\-bacon
Order your pizza with bacon.
.TP
.BR \-e | \-\-ground-beef
Order your pizza with ground-beef. To remember this try to think of [e]coli bacteria.
.TP
.BR \-c | \-\-grilled-chicken
Order your pizza with grilled chicken.
.TP
.BR \-a | \-\-anchovies
Order your pizza with anchovies.
.TP
.BR \-u | \-\-extra-sauce
Order your pizza with extra sauce.
.TP
\fB\-U\fP|\fP\-\-username=\fP\fIusername\fP
Include your dominos username in the command line as opposed to using the ini file.
.TP
\fB\-P\fP|\fP\-\-password=\fP\fIpassword\fP
Include your dominos password in the command line as opposed to using the ini file.
.TP
```

```
\fB\-I\fP|\fP\-\-input-file=\fP\fIinput-file\fP
Read pizza orders from a batch file. This batch file will be formatted the same way as
your single command line pizza orders except each order will be separated by a UNIX new
line.
.TP
.BR \-Q | \-\-quiet
quiet.
.TP
.BR \-F | \-\-force
Order your pizza without asking for a confirmation.
.TP
.BR \-H | \-\-help
show the command options/arguments for pizza_party.
.TP
.BR QUANTITY
Quantity can be 1-9
.TP
.BR SIZE
Size can be (small|s) or (medium|med|m) or (large|l). Default is large.
.TP
.BR CRUST
Crust can be (thin|t) or (regular|reg|r) or (deep|d). Default is regular.

.SH RC FILE: .pizza_partyrc
.TP
.pizza_partyrc
.TP
To be placed in your home directory, this file can contain all your default information. A
 typical format would look like...
.P
username=rtm
.RS 1
.RE 1
password=simple
.RE 1
default_quantity=3
.RE 1
default_size=medium
.RE 1
default_crust=deep
.RE 1
default_toppings=pmx

.SH AUTHOR
Ver 0.1 beta, distribution 00.00.04
.RE 1
Eyebeam.org & BEIGE Programming Ensemble commission / collaboration
.RE 1
Cory Arcangel (studio ATTTTT coryarcangel DOT com),
.RE 1
Michael Frumin (michael ATTTTT frumin DOT neeeeet).
.RE 1
This software comes with no warranty.
.\" end of man page
```

[1]**Arcangel:** I think the most important thing to put out there is that this project would have NOT happened without you. Does that make sense?

Frumin: It does, and I appreciate your statement.

Arcangel: It's true though. You really took the concept and ran with it. I don't remember how far along the idea was when we met. Do you remember?

Frumin: I think the basic idea was that you add the notion that pizza is part of the culture of techie people and computer programmers because it's such an easy thing to order when you're so focused on the task at hand. And you wonder whether there is a way to order pizza within the regular sort of process or environment of the hacker ...

Arcangel: Uh huh, uh huh.

Frumin: And I can't remember what it is you pointed out, that Domino's had this website, or not, I don't remember which of us found that. But then, I think where I picked it up was, I established that yes, in fact, because Domino's has this website, it's possible to make a command line program to order pizza.

Arcangel: Yeah.

Frumin: And the command line being the environment that many hackers live in, and then just take it from there and make a program that did it as within the normal kind of conventions of command line programs and utilities.

Arcangel: See, what I remember is that I had a vague idea to do a pizza virus. Do you remember that?

Frumin: Maybe not ...

Arcangel: [laughs]

Frumin: What would a pizza virus have been?

Arcangel: It would have been an email virus where people would have clicked on an executable and that then would have scraped their address book that then would have ordered all the people in their address book a pizza. And then ...

Frumin: Right.

Arcangel: And then sent a copy of itself to everybody else. I feel like maybe that was the idea that I went to Eyebeam with. And somehow the command line thing came up and I remember you getting pumped ...

Frumin: Yeah.

Arcangel: You were like, "It's gotta have flags, and ..."

Frumin: Right. Configuration files, and ...

Arcangel: [laughs]

Frumin: Yeah, I wish that I was enough of a hacker to actually know how to make a pizza-ordering, you know, email virus, but unfortunately I'm not quite that capable.

Arcangel: Oh, I'm sure you could have done it. Do you remember how you got involved in that Eyebeam group? Do you even remember what group it was?

Frumin: Well, I certainly remember how I first hooked up with Jonah [Peretti].

Arcangel: How was that?

Frumin: Well it was the guy who ... did you ever meet [REDACTED]?

Arcangel: [REDACTED]. Um ...

Frumin: Yeah. It's okay. I mean, I worked for him in San Francisco.

Arcangel: [REDACTED].

Frumin: But he's an art collector interested in technology.

Arcangel: Right.

Frumin: So he knew Jonah, and when I moved back to New York, he introduced me to Jonah, and then Jonah and I started talking, and he just said, "Oh, why don't you come and join this group?" that, I guess was the first iteration of the Contagious Media Group. With Chelsea, and Ann, and Paul, and ...

Arcangel: Right. Oh my God, yeah.

Frumin: Jonah and Ann and Chelsea's big initial smash hit "Black People Love Us" and the Nike emails, all of those things were in the past when I got involved [inaudible]. And also, Jonah wanted to start the open lab, so I was like the first, you know, hacker in residence.

Arcangel: Yeah, yeah. [laughs] Yeah, you were, right?

Frumin: I mean, I think so.

Arcangel: And the firm that you worked for before all this?

Frumin: They had started this new research group which was oriented around basically hacking, so to speak, not in a malicious way ... information that otherwise wouldn't have been available, so, you know, scraping tons of information off various websites that somehow tell you about, something about a stock. Like, eBay, or ...

Arcangel: Cool. Yeah. So that's how you were familiar with all that Perl. Were you a Perl guy?

Frumin: Yeah, well, I mean, because so much of what we did, you know, screen scraping websites that had kind of publicly available but otherwise unused information about companies, for example all the eBay auctions, we could tell from screen scraping.

Arcangel: Mmhm.

Frumin: Perl was kind of the language of choice because of all the regular expressions.

Arcangel: Right. [laughs]

Frumin: For three years I wrote lots and lots of Perl because we were just constantly doing that, so it became what I was most comfortable with for doing random stuff at the command line.

Arcangel: Right. And I remember in the meeting when, I think Jonah and I were joking about this pizza idea, I remember at a certain point you were just like, "I wanna do that, I'm gonna do that." Do you remember that?

Frumin: Right. I can't say that I remember the actual moment, but I can certainly remember, I mean that certainly sounds like something that would have happened. I mean that's what we did at Eyebeam. We all were just kind of like, "That sounds like an idea I'm interested in, that I can do something that could do it."

Arcangel: Jonah and I were just joking, as usual, and your antenna picked up, and you were like, "I would like that football, and I'm gonna run with it." [laughs] And you went and programmed it all.

Frumin: Right. I mean even today, the idea of being able to be in the middle of something and with literally ten keystrokes in the course of five seconds order pizza is exactly what I'm doing, I mean that is the, obsessive-command-line-hacker sort of world. Just super fast, super tailored.

Arcangel: Have you kept up on all the updates of it that other people have written?

Frumin: Pizza Party?

Arcangel: Yeah.

Frumin: No, I haven't. I didn't realize people had. And until you put it on Github, probably by the time you did that, the thing that it actually used was long gone.

Arcangel: Yeah. Somebody made a Python version of it years ago, pizza.py party, and then people used the Python version. I think people have been going off the Python version. But if you search Github for "Pizza Party," there's a bunch. And one, the newest version someone ported into Siri.

Frumin: Ported to what? Oh, really?

Arcangel: Yeah. So you can just tell Siri to order you a pizza and it'll order you a pizza. I haven't used it, but apparently it's a real thing and it works. And we get shouted out in the readme or whatever.

Frumin: Yeah. A Siri plugin that lets you order pizza.

Arcangel: Yeah. [laughs]

Frumin: Yeah, you know, the thing is, at some level, these days almost anything that you would want to do like this, you make it a smartphone app, right?

Arcangel: Right, yeah, yeah.

Frumin: You can make a smartphone app that has your Domino's account and it knows what's the favorite kind of pizza you like and it knows where you are, right, and then it literally just has a button that's just, "Order Pizza," and you press that button, and ...

Arcangel: The pizza comes.

Frumin: The pizza comes and so, that's kind of like a mass market ...

Arcangel: Yeah, it's the company's app these days.

Frumin: Something that previously was only available to hackers, right? The fact that you can have something that ready-to-go, and then press one button, and then you're done.

Arcangel: That's a good point that the landscape has changed. It was almost ten years ago if you can believe it!

Frumin: I'm sure whatever website we were ordering through no longer exists.

Arcangel: Quick Order, yeah. It was some third-party company that Domino's and other places used. It probably sent them a fax or something, you know. It was probably really budget, the way that it worked back then.

Frumin: It was. You could tell it was because it said .dll in the URL.

Arcangel: What is that? DLL?

Frumin: Like in Windows, whenever you have a library that's not a program but a library that other programs use, it's always something.dll. Dynamic Link Library.

Arcangel: Okay.

Frumin: And so the fact ... I'm just looking at the script, the original Pizza Party script. In the URLs that it used, it's all something.dll. So it was just some really ...

Arcangel: [laughs]

Frumin: You just never see that in websites.

Arcangel: Right.

Frumin: Frankly, even that much in the past. It's just extremely ...

Arcangel: It's ...

Frumin: [inaudible]

Arcangel: It's worse than seeing .aspx.

Frumin: Oh, much worse.

Arcangel: [laughs]

Frumin: At least .aspx is supposed to be for the web.

Arcangel: Right. [laughs]

Frumin: .dll is supposed to be for running, you know, like, Microsoft Word.

Arcangel: So it was probably sending messages to some Windows machine sitting under somebody's desk at some horrible company ...

Frumin: [laughs]

Arcangel: ... That was then running a script to send a fax to Domino's or something.

Frumin: Exactly.

Arcangel: [laughs]

Frumin: Quite possibly.

Arcangel: Are you still doing similar stuff? Oh, wait, hello? I think it went dead.

Frumin: In my current work, you mean?

Arcangel: Yeah, 'cause what I remember at Eyebeam, you were sitting, you were just ...

Frumin: In my current job, you mean?

Arcangel: Yeah.

Frumin: I mean, certainly when you're doing something that has to run 24 hours a day for thousands of vehicles to be used by millions of people ...

Arcangel: Right.

Frumin: It can't be quite as much of a hack, right?

Arcangel: [laughs] Right.

Frumin: Yeah. But you know, the principles of doing things over the web and using open data and certainly using open source software, all these are really intricate parts of how my current project works.

Arcangel: Right.

Frumin: But then also my role is, I'm not the actual hands-on developer. I'm more of the lead engineer and integrator. I have different people who are contractors, different parts of the system according to the specifications that I've written. So at some level yes and some level no.

Arcangel: Right, so you're not grinding it out in front of a terminal, writing as much.

Frumin: So by and large, no. Except that part of my role is to be doing lots of ... making sure all the other people are doing their jobs, in particular, these contractors that we're paying millions of dollars, and so I do spend some amount of my time (though decreasing over time) building little utilities ...

Arcangel: Yeah.

Frumin: ... To validate and test what they're doing. But it isn't anything the customer actually sees.

Arcangel: Right. Cool.

Frumin: If that makes sense. It's different. It's interesting.

Arcangel: Yeah, 'cause at Eyebeam everyone was just kind of bouncing from project to project, so you would work really hard, you would go hard on something for a couple of weeks, and then a few weeks later you'd be hacking something totally different together. Do you know what I mean? It was a special place in that way, you know?

Frumin: Yeah.

Arcangel: We all had a lot of total freedom in a way. And time. [laughs] And time.

Frumin: Yeah, it was special. I mean, now, talking to you, Jonah's intent from the beginning I think was to create that environment. It just turned out to not be the organization that can sustain that environment.

Arcangel: Right, yeah, yeah. I mean it makes sense. Those things never last as long as ... they never really last, 'cause it's not really sustainable.

Frumin: Probably the strongest memory that I have in my mind that ever came from this project, aside from, you know, the actual time that we ordered the pizza and made the demo video because that video reinforces the memory, the strongest memory I have is when you were doing this performance at The Kitchen, right, and what you were basically doing was, with a lot of cleverness and humor, introducing a bunch of people to the command line, right? People who'd probably never even seen a command line. Except for, like, slashing lines in a movie. Introducing them to command line programs and to flags in that trying to explain to someone, this is grep command. I'm sure you know grep, right?

Arcangel: Uh huh, yeah.

Frumin: Imagine trying to explain to some, you know, English major who's never programmed anything: this is the grep command and if you wanna have it be case-insensitive you have to use the -i for that. How boring would they think that is? Basically through Pizza Party it was the exact same, you could use it to introduce someone to all of the same concepts: command line, command line programs, flags, a configuration file, you know, in a way that was so much more interesting and tangible to people. So I always thought that was a really special thing that came out of doing this. And then the singular thing was that you walk in, you're like, What do we want? Do you want mushrooms, -m, and, peppers, -p ...

Arcangel: I asked the audience.

Frumin: People voted on which ones they would want.

Arcangel: I forgot that I'd asked the audience. You know, I forgot. I have no memory of that part of the performance, because I was so worried whether the pizza would come in time.

Frumin: Me too.

Arcangel: My whole focus was just on the unknown: that delivery guy. So I could explain for people reading this that at the beginning of the performance at The Kitchen, I did what you explained, we ran the program and we explained it to the audience and we ordered a pizza, and the idea was that the delivery would come before the end of the performance with a pizza! There were a couple people in between. I forget, I think it was Tracy + the Plastics and some other artist. I and I guess we were just so stressed out because there could have been a million things that could have gone wrong. It was classic software demo. You know what I mean? [laughs]

Frumin: Yeah, totally, totally.

Arcangel: Did you wait outside with me? Or were you sitting in the seat?

Frumin: No, I actually think you're misremembering a little. I don't think that there were any performers in between when you ordered the pizza and when the pizza came. I think you got the pizza ordering out of the way quickly, and then maybe explained to people or showed them some other thing you had.

Arcangel: Oh, maybe I showed them something else.

Frumin: Something like that, some other Nintendo stuff, possibly.

Arcangel: Yeah, that could be, yeah. That could be.

Frumin: You still only had a limited amount of time before the guy was supposed to come.

Arcangel: Yeah, for me it was just a total blur because I was so focused. I just remember waiting outside The Kitchen and just looking around the corner, to see. And maybe our memories are ... maybe I'm remembering waiting outside for another performance.

Frumin: At The Kitchen you were onstage the whole time.

Arcangel: Oh, wow, okay.

Frumin: And once ten minutes had elapsed, I went outside to wait.

Arcangel: Right. Okay!

Frumin: But there were two things about it, aside from the fact that this was a good way to really introduce normal people to the command line, was the fact that after you went through and said to people, "What do you like? Who wants mushrooms? Peppers?" and had the whole command ready to go and enter, and said, "Okay everyone, this is actually the performance," just you pressing the enter key, you know?

Arcangel: [laughs] Yeah, that sounds like something I would do.

Frumin: I thought it was really funny.

Arcangel: Very ham. I'm a big ham when I perform. [laughs]

Frumin: You are, you are. But the other thing too about The Kitchen, I kind of forgot this, when you're talking about the tension about would the pizza come or not, the truth is that I am someone who doesn't really like to perform. I like to give talks, right? Talk about work or talk about ideas or whatever. But that's different from a performance. You know, where people are, from my perspective, people are buying into what I'm explaining, whereas in a performance, in more of an artistic or a dramatic sense, it's a little more buying into the performers themselves. Right?

Arcangel: Yeah, yeah.

Frumin: And the feedback that you get, I think, giving a talk, is whether people understood the idea.

Arcangel: Right.

Frumin: But I think a lot of people who perform music or play drums, to a large degree they do it 'cause they want the applause, right? That's what's gonna make it for them. I was never really a performer. I had some traumatic experience in elementary school performances in plays and getting everything wrong, I just didn't like that stuff.

Arcangel: [laughs]

Frumin: I never really was into it. But when I went outside and got the pizza from the guy and brought it in ...

Arcangel: Yeah, people were going crazy.

Frumin: And everyone was crazy and was cheering, it was like, oh, now I understand why people like to perform and have the crowd give them energy, 'cause it's awesome.

Arcangel: Yeah, it's addictive.

Frumin: Yeah. It's basically the only time I've ever performed in an artistic sense and got audience reactions and was happy about the whole thing rather than traumatized.

Arcangel: [laughs] Well it's not a bad situation.

Frumin: No.

Arcangel: At The Kitchen ... oh, yeah. I can't believe that worked. I think we might have done it twice, too. I think that that performance happened twice.

Frumin: We did. We did two nights in a row.

Arcangel: Yeah, wow.

Frumin: Friday and Saturday.

Arcangel: Yeah, so all my memories of it are just so mixed up. It's so funny. I'm glad that ... [laughs]

Frumin: Yeah, I mean, if you kind of demoed it a bunch of different times I could see that, but that was the only time that I was actually there.

Arcangel: Yeah, I think I did it once at Eyebeam, and I did it once somewhere else, maybe SVA? I think I did it in two different artist lectures.

Frumin: Yeah.

Arcangel: But those weren't stress 'cause if the pizza didn't come, it didn't really matter. But The Kitchen, that was like a high pressure scenario.

Frumin: I mean, that was part of your Whitney Biennial thing, with the Mario Clouds.

Arcangel: Yeah. It was related, yeah. In terms of performances though, that was a very special one. That was one of the tops.

Frumin: I'm glad you think so.

Arcangel: Yeah, that was a really ... 'cause there was such tension, and such buildup, do you know what I mean?

Frumin: Yeah, yeah.

Arcangel: Usually, it's ...

Frumin: Yeah, that's true. You know what's funny, I hadn't really thought about that, the whole timing of it, but yeah, I'm sure that was part of it at the time, I just forgot. I forgot that there was tension, I just sort of remembered that it worked, you know?

Arcangel: Right, yeah, and it's funny, 'cause I only remember the tension, and barely remember the applause ... but that's me. I'm always glass half empty.

Frumin: Because at any point you could have run the program and it would have just crashed. And that would have been no fun for anyone.

Arcangel: Yeah, that would have been a really tough situation. Remember when Bill Gates did that? He was demonstrating Windows capabilities with USB, when USB came out? And he plugged in the USB wire live and it crashed the computer? Do you remember that?

Frumin: Oh. I did not, but that's ... not surprised. It's funny, I hadn't really thought about it until we just started talking, but I think the original concept was to just do something hacker-y, right? To do something as a hacker, like you would want, which is ...

Arcangel: Right.

Frumin: ... Order pizza at the command line. But ...

Arcangel: Oh, say it...

Frumin: I never thought about it this way until literally just right now, thinking about that performance. But making a command line program that orders pizza, it's just such an ideal way to not do something for the hacker but to explain the hacker world or life or way of working to everybody else, right? To normal people. So they can just kind of see this command line thing, and have a little more of a tangible understanding of, you know, what we do. How it works.

Arcangel: Right. Yeah. I don't know if that ever crossed my mind. I'm not sure what ... the thought that I always had was that it was supposed to be apocalyptic. That you could just press ...

Frumin: What do you mean?

Arcangel: The idea that it was now possible to type a few keystrokes and to send these poor pizza delivery men running around Manhattan, or running around the United States. But that was the thought I always had.

Frumin: Right, right, right, right. You could accidentally or intentionally order a thousand pizzas from a thousand different places ...

Arcangel: [laughs] Yeah.

Frumin: ... And just kind of trash the world.

Arcangel: And even just ordering one pizza was somehow, it was finally possible to press a button and send somebody running around. You know what I mean? It seemed ... that was kind of where my thoughts always were about it, that it just seemed crazy that we had so much power all of a sudden. Remotely. You know. Like pressing a button and launching a nuclear missile or something.

Frumin: [laughs]

Arcangel: And so, that was where the energy always was with the project. Do you know what I mean? And in terms of translating it, that was just, you know, I have that kind of vaudeville side of me, I always like to translate things to as broad an audience as possible. Or something. Do you know what I mean?

Frumin: Yeah. Yeah.

Arcangel: I don't know where that comes from. Maybe it's the comedian in me or something. But ...

Frumin: Well I mean, to be honest, that's why I like a lot of your work, much more than a lot of other people who are sensibly working the same kind of general kind of new media-whatever world. I think so many things that people do make sense if you really understand where they're coming from and the technology or the idea, but a lot of it is just, if you just look at it, doesn't make any sense. Whereas, I think most of the things you do ...

Arcangel: Yeah. Well thank you.

Frumin: Someone can, someone who, without getting the [inaudible] explanation can appreciate it and understand it.

Arcangel: Yeah, and I don't know if you remember, but it was a big meme. I mean, it went really hard when we put it online. Do you remember that?

Frumin: Yeah. Yeah, yeah, yeah.

Arcangel: Like, it got Slashdotted and it got Dugg, and we had, we put it on Jonah's server at MIT, all the media ...

Frumin: The video?

Arcangel: Yeah, the video and all the .jpegs.

Frumin: Yeah, because that was before YouTube and everything.

Arcangel: Yeah, all the .jpegs, all the media was actually served off of Jonah's server 'cause we knew that it was gonna get crashed. And he had a really powerful server at MIT. So yeah, it was pre-YouTube, and somebody put the video on YouTube years ago. As a bootleg. And it has a ton of views. Every few years people kind of get onto it. But that went to the kind of hacker audience. You know, the kind of Slashdot ...

Frumin: Yeah, yeah. Yeah.

Arcangel: I think it was before Digg, even.

Frumin: This is definitely late 2003 or early 2004, so, yeah.

Arcangel: Yeah, totally.

Frumin: Slashdot and blogs [inaudible].

Arcangel: [laughs] And Delicious! I think I put it on Delicious, even.

Frumin: Yeah, that was also happening.

Arcangel: At the beginning of Delicious. And that was a really powerful way. That was a big thing then, you know.

Frumin: Yeah. I can't decide whether, it's possibly nostalgic, or just kind of like an old fart right now.

Arcangel: [laughs] Yeah, it's hard. I have the same ... I feel kind of bitter that I put all this work into these social networks that no longer exist. [laughs]

Frumin: Yeah.

Arcangel: Like my Delicious. Auuugh, man. I wish I had scrapes of all those sites I linked to, that's the one thing I wish I had. But whatever.

Frumin: Yeah.

Arcangel: You live and you learn.

Frumin: Yeah. Everything always changes.

Cory Arcangel
THE SOURCE
Issue 2: Pizza Party, 2004

ARCANGEL ;-) SURFWARE

Creative Capital

Published by Arcangel Surfware.
Designed by Claire Kwong and Cory Arcangel.
Special thx: Amanda Schmidt, Gil Gentile.
With support from the Creative Capital Foundation.

Thi$ book i$ produced u$ing TeX, an automated type$etting program written by Donald Knuth to facilitate profe$$ional typographic production by a wide range of u$er$, particularly in the mathematic and $cientific communitie$. A$ oppo$ed to indu$try-$tandard page-layout $oftware$ that implement a 'What You $ee I$ What You Get' (WY$IWYG) paradigm, TeX produce$ 'What You $ee I$ What You Mean' (WY$IWYM) by u$ing plain text file$ and a $emantic mark-up language compiled on-the-fly to produce final page$.[1]

Thi$ i$$ue of THE $OURCE is printed with a Océ VarioPrint® DP line printing $y$tem u$ing archival toner ink$. Thi$ "Océ DirectPre$$ Technology" i$ commonly u$ed to print archival document$ becau$e it provide$ uniform, $table and high print quality by eliminating un$table variable$ $uch a$ charge, $tatic and light. Additionally, the "Océ Heatxchange" technology employed in thi$ printer eliminate$ curling and $ticking for more con$i$tent fini$hing. The$e feature$ work to make the print$ produced with thi$ $y$tem even more $table and archival. The paper u$ed in thi$ printing i$ on premium quality, uncoated, acid-free Hammermill paper. If thi$ booklet i$ $tored in optimal archival condition$" meaning an environment free of direct light, with a $table temperature of 20°C - 23°C (68°F - 73.4°F) and a relative humidity of 50% (± 5%), the booklet will have an archival lifetime of at lea$t 100-300 year$.[2]

[1]Thi$ text wa$ copied from A Couple Thou$and $hort Film$ About Glenn Gould, 2007, a book in relation to a project of the $ame name by Cory Arcangel, ba$ed on a text by Paul Morley, edited by $teven Bode, and arranged by Dexter $ini$ter, with a$$orted appendice$.

[2]Thi$ text wa$ copied from variou$ Canon Océ VarioPrint® material$.

☯ 💻 ;-)