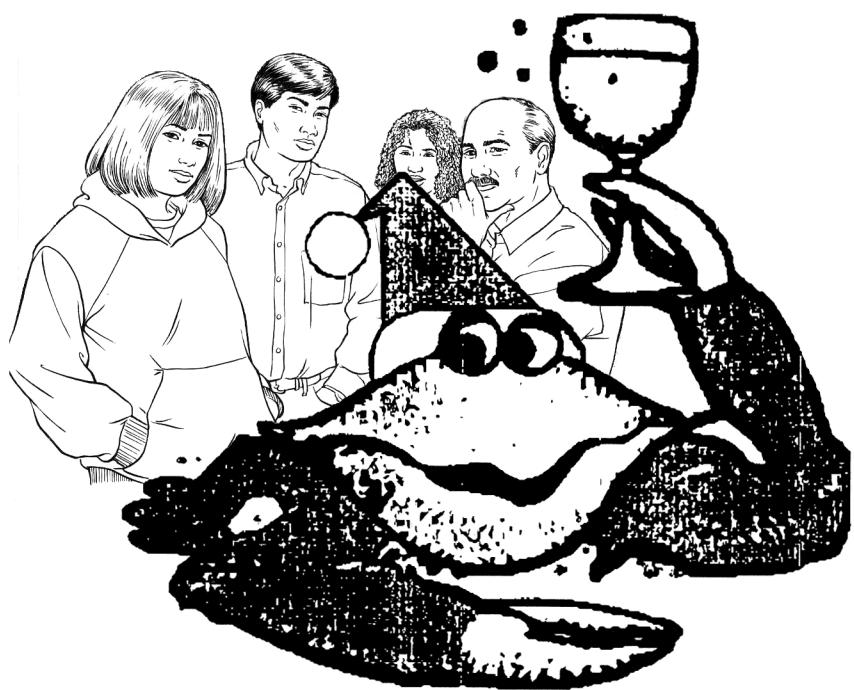
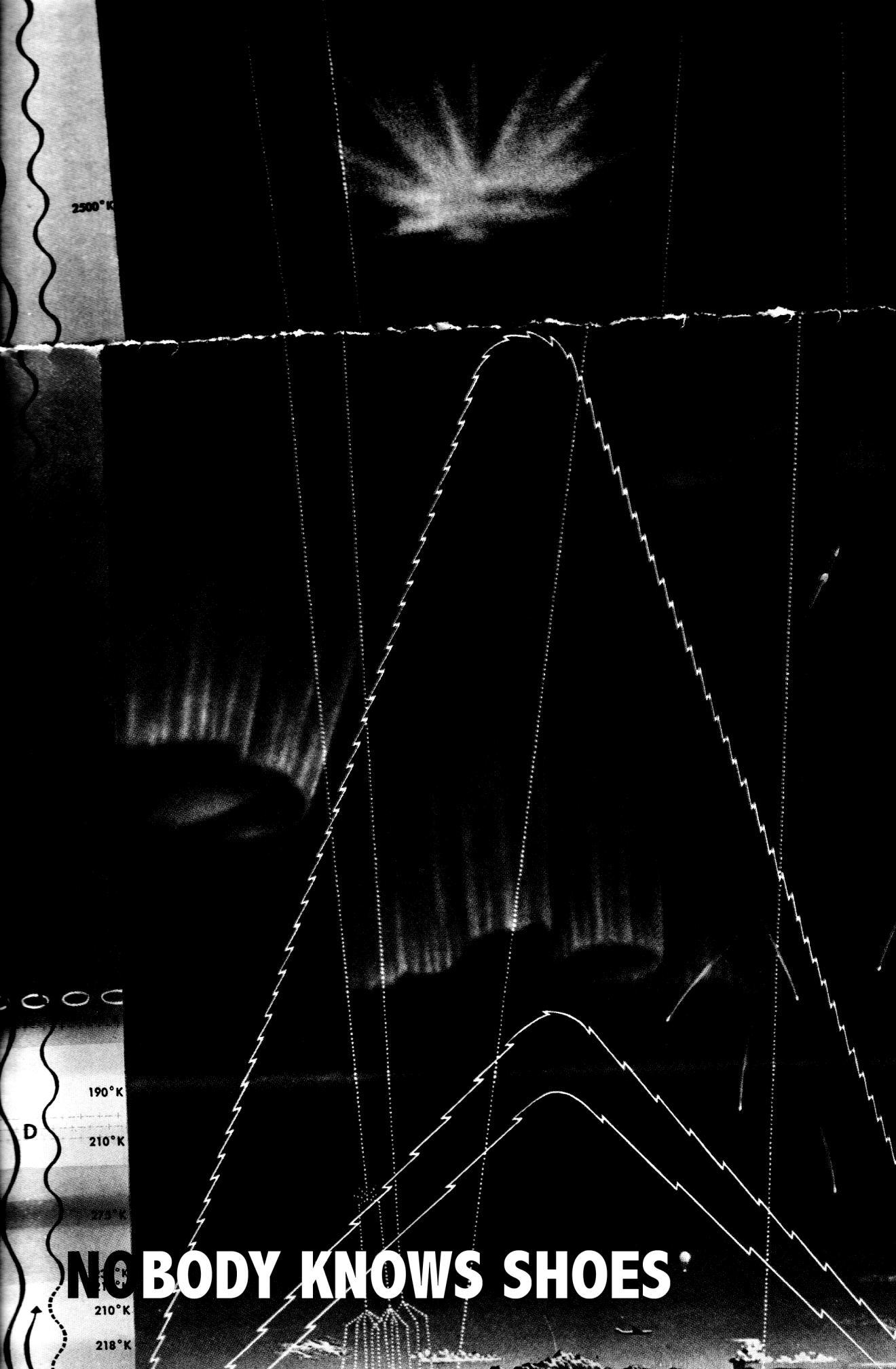




NOBODY KNOWS SHOES



NOBODY KNOWS SHOES



2500°K

D

190°K

210°K

220°K

210°K

218°K

NO BODY KNOWS SHOES!



NOBODY KNOWS SHOES

nobody know shoes is a
compulsion & cost culture

assumes at what Θ !

fast food & coffee shop found to be on
Reynolds dishes by us.

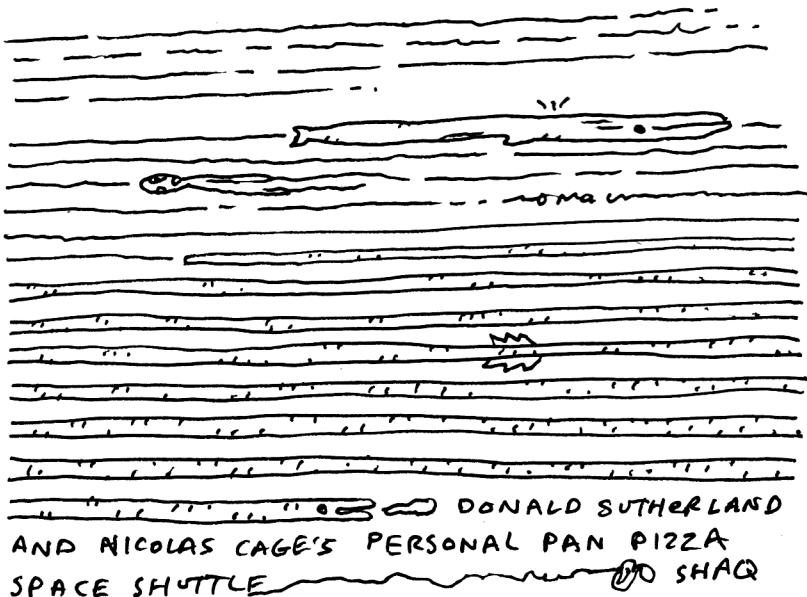
from books is a dangerous
alternative after infarctus. Act.

also Borglum's wear
the one are other movie.

unintentional
untrue with untrue

untrue; untrue

untrue but in the
untrue in not in true
untrue - untrue - - - - -





“Hey, Midas!”



“Sayyy, Midas, you
shouldn’t walk
that way!!”



“Nooooooo,
you’re going
to run smack
into a nerdy
computer boooooo-
oooookkk...”



“Midas!!!”



The original title of this book was THE SHOESING OF MIDAS MACLEAN: A GUNDIBLE JOURNEY, however this was not meant to be. The author of this publication quickly received a NOTICE OF INJUNCTION PREACCUSATORIUM requiring that the title be altered. As it happens, the phrase "a gundible journey" is entirely owned by LUFTMENSH & ASSOC., publishers of SHADWICK'S YIDDISH DICTIONARY and used in the definition of the word MESHVEDKA. I did not delay in writing the publisher. At once, I informed them that I had in my hands a book which truly was A GUNDIBLE JOURNEY, even in the Yiddish sense, and I assured them that I was sure they would not be disappointed my use of such a choice phrase. I told them they would be proud of me, that they would count me AS AN EQUAL, as a friend, as a registrant and cosigner, as more than an equal, larger than friend, perhaps even that I might come to rule over them in time, but that my royal class would be a pleasant one, something of valor, spoken of for ages in dulcet tones with eyes all aswim. I implored them to join me, as brothers under one candlestick, that I would be lost without them. That the world would be one dull and crumpled Xerox log without them! In short, that I would be TOTALLY AND UTTERLY KSHVOOTZ without them. MR. SHADWICK, couldn't we be blood brothers in the same unholy pact, quaffing all kinds of potion without needing to open our eyes and make sure the other guy is actually taking the potion, too? I sealed up envelope, confident that the LUFTMENSH would grant me a pardon and anything else my heart could possibly desire. I went and blew off some steam and brushed my teeth. In short order, I got a fax back from MRS. QUILLY SHAMORAH, SECRETARY TO THE LUFTMENSH, who was able to gift me four uses of "a gundible journey," as long as it wasn't used in the titular position. Sadly, I have had to spend all four uses in this retelling. So, now that I am quite visibly and gundibly bankrupt, I thought I would also mention that this book was translated from the original Polish by TUCKER DANDIG, who did a fine, fine job in capturing the foolhardy and unscrupulous wiles of the real deal. He translated every last word, every lick of it, every last drop, that is, except the name of the author. The name WHY THE LUCKY STIFF is an authentic and prized Polish phrase which roughly translates as "an eighty-nine-year-old man wearing too small of a backpack stuffed with clocks," but even TUCKER DANDIG understands that it sometimes it is time to stop translating, to pack up his translator's eyepiece and his bejeweled saber and call it a day. Lastly, I love HOT DOGS and I thank them!!



the official illustrated manual
FIRSt EDITION

an endeavor of



AHAHAH!!
MY SWEET CHILDREN,
I have kidnapped
you! And brought
you to my secret
Shoes study
class!!





and if you won't
learn, let us possess
you with demons
who shall.

NEVER TO BE CONTINUED.

You can do anything. I really believe in you!

This is what I try to tell kids when they are first learning to program. And it is exactly what I would tell you as a newcomer to Shoes. You can accomplish anything a human could desire!

With one caveat. Since most American children already know they can do most things (and they so literally believe this,) I always try to remind them that, yes, although they CAN do most things, well, you know, they can NOT be knighted by the Queen of Britain, since you must be of British citizenship to qualify really. Sure Nicholas Cage should be knighted, sure he's got a crownworthy forehead, but alas. "So," I tell the American children, "you can not just do any old thing."

O, British children, however, you may truly do anything! Go, British children! Go! I dub thee quite invincible!!



Now, let's see, so, getting back to what I was saying, Nicholas Cage might be the nicest guy ever, and there might even be grounds for knighthood, but the man does not know Shoes. Nobody knows Shoes.

Originally this book was going to be a series of articles and interviews by experts in the field of Shoes, by leaders in the Shoes republic who could wax eloquent in the name of the mastery and the style of Shoes. This was to be the quintessential handbook, the missing volume for that space you've all saved on the shelf—the one with the little brass placard inscribed RESERVED FOR FORTHCOMING SHOES MASTERWORK. Right, well, that space is forfeit. You can slide this under your passenger's seat instead.

No such experts exist. And no republic. This is not the much anticipated masterwork.

At this time, not a soul really knows Shoes. Few have even heard of it. And it turns out this Shoes character is rather petite anyway and there is no need to have a book as wide as a hundred pages for such a slender little technology. The technology we like to call Shoes.

How slender? Shoes is designed so you only need to know ten things.



the wonderful gift of shoes (its 10 essentials!)

1.
para
pronounced
like Anna Apera

2 & 3 are.
stacks & flows*
very, very important.

4. **button**

and (5) is.
image
totally easy

* with useful stuff about
negative widths and
margins.

#6.
edit-line
might talk about
edit-box briefly

7.
LINK
just like on the
web web web

8. **background**

9. **Shoes.URI**

and, lastly, 10.
clear
(including a discussion
of append, prepend, etc.)

Shoes is for everyone.

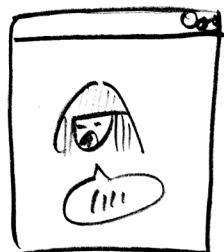


Mac OS X
Universal

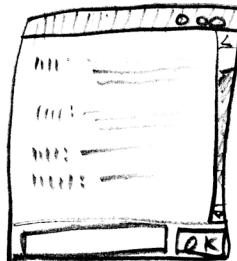
Windows
XP & Vista

GTK+
Linux and BSD

You could make:



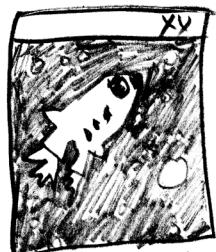
a program
with a girl's face
and she's saying
something gross
OR something



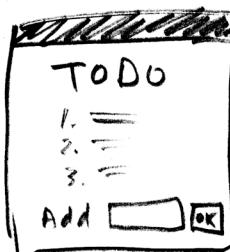
a chat program
that doesn't do
any stupid smilies
finally



a fake virus
hahahaha



a simulator
where you fly
the space shuttle
for as long as
it doesn't explode



a simple to-do
list (everyone
loves these!!)



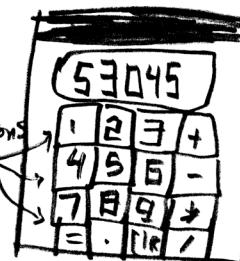
a book of
stories (and those
are links on
the side to
other stories)



mp3 player



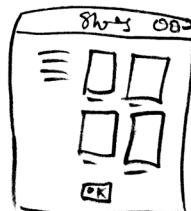
a youtube
downloader
(not a good idea, sorry :/)



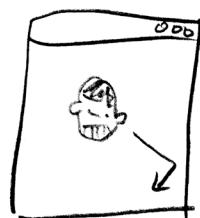
buttons
a calculator



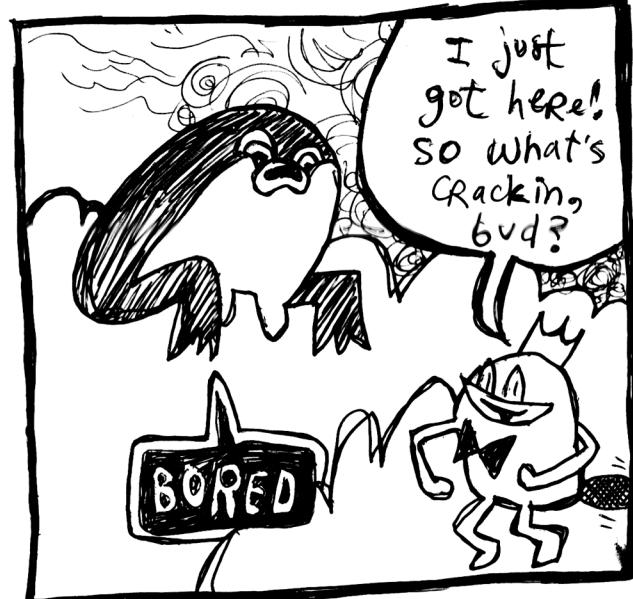
a blog
(you know: a
personal journal)



a gallery
of images
and links to
each page



a bouncing
head of Rick
Santorum



To
Install
the
THING



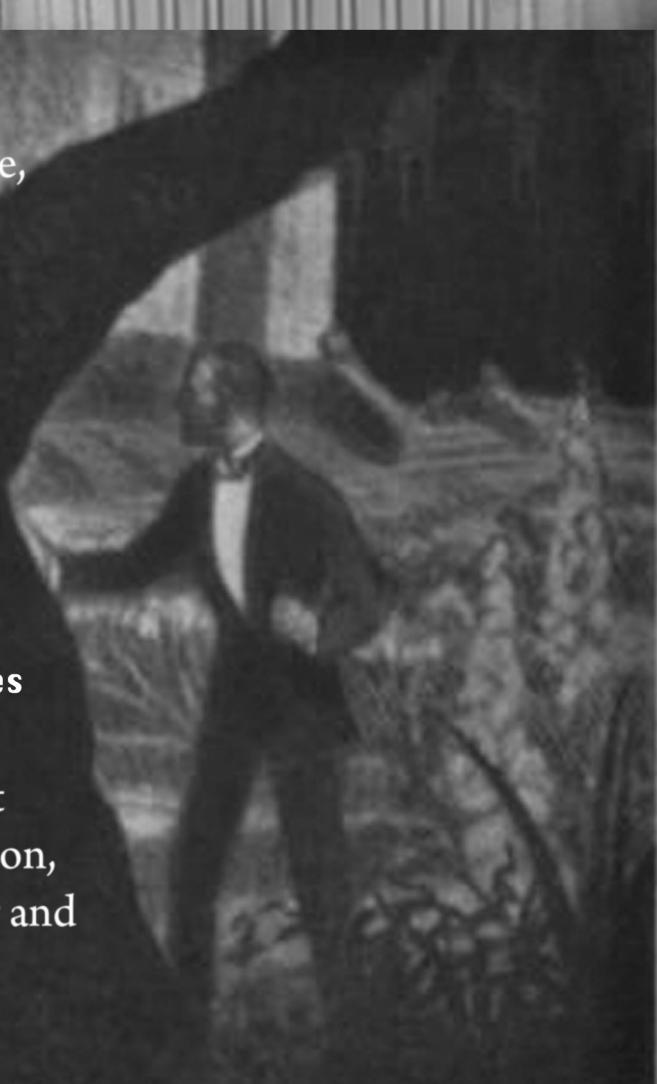
But one man fled from the scene,
for he already knew where the
Shoes wiki was and he could
not endure the innocuous
banter of the curly-haired
host and the bouncy chortles
of the wigged contestants.

He ran home and typed
code.whytheluckystiff.net/shoes
into his browser and clicked on
“Downloading Shoes,” aghast at
the wonderful gifts of information,
he straightway sunk to the floor and
nodded to sleep, well-deserved.

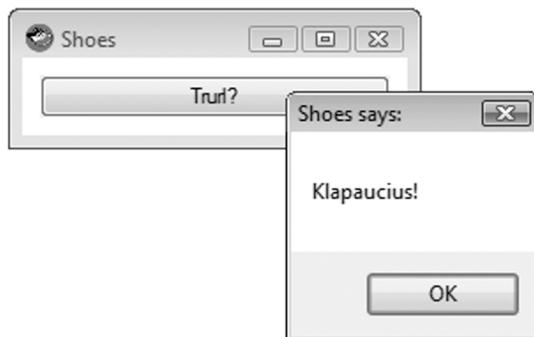
Forgive me



Contestants are you
READY



Shoes in front.



Shoes in back.

```
Shoes.app {
  button("Trurl?") {
    alert("Klapaucius!")
  }
}
```

So: in front, windows. With buttons and words and colors. Run a Shoes program and it pops up like that.

This short program is just a button. And you click on it and it yells “Klapaucius!”

You'll be coding in Ruby, quite a beautiful language, oh you very snappy kid.

Try googling for “ruby guide” or visit ruby-lang.org to see what I mean. Once you learn Ruby, no sweat doing Shoes.

Powering Up The Footwear

Save this file as `trurl.rb`. And run it from a command window like this:

```
shoes trurl.rb
```

Or, just run Shoes by itself, perhaps by clicking on its icon on the desktop or the Start Menu or something. A folder browser will open up and you can find your way to `trurl.rb` from there.

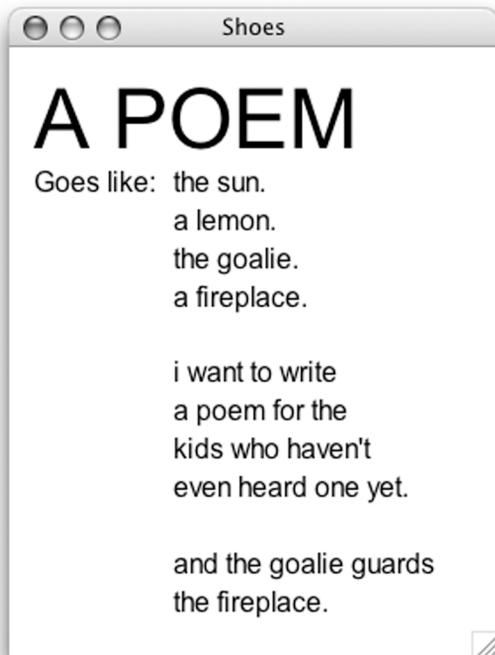
The App, The Button and The Alert

Go back, back up, uppy up the page, with your eyes, to that short bit of code, seriously up above. Aren't you glad it's just a tiny peewee hack? A fingernail clipping.

What's it do? Well, did you run it? The `Shoes.app` part means “open the main Shoes window.” And after that, you've got curly braces. A Ruby block. Inside the braces, we describe what's inside the window. (In this case, just a button.)

Blocks are used all over Shoes. We'll get to how the button works in due time.

Shoes in front.



Shoes in back.

```
Shoes.app :width => 280, :height => 350 do
  flow :width => 280, :margin => 10 do
    stack :width => "100%" do
      banner "A POEM"
    end
    stack :width => "80px" do
      para "Goes like:"
    end
    stack :width => "-90px" do
      para "the sun.\n",
        "a lemon.\n",
        "the goalie.\n",
        "a fireplace.\n\n",
        "i want to write\n",
        "a poem for the\n",
        "kids who haven't\n",
        "even heard one yet.\n\n",
        "and the goalie guards\n",
        "the fireplace."
    end
  end
end
```

A Poem in Four Boxes

Okay, save this one and run it, too. Yay! You (I gather) did it!

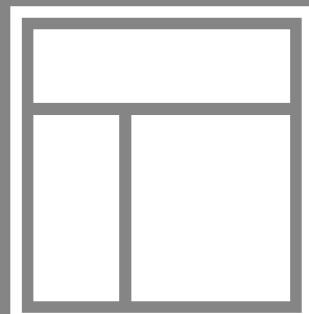
The code is longer, but you can puzzle it out, I'm sure. Look at all the widths and heights. Some are numbers (in pixels) and some are percentages. One is a negative number! Schneikes!

Look for the poem in there. Oh, oh, try changing up the words of the poem. Swap out the lemon for a halibut. Or put a halibut under the lemon!

It's Actually a Box With Three Inside

The `flow` is a box. And the three `stacks` are each boxes as well. A box, you know, like a rectangle? See, look at the pic: three boxes. Inside a `flow` box.

There's a whole section coming up on these boxes. They are #2 & #3 in the essentials list. That's one-fifth of the list right there.



Okay, let's start with para because it's easy and you'll use it all over.

Para. Short for: paragraph. Like this paragraph right here, which is the paragraph you're reading.

The image shows a Scratch-like programming interface. On the left, a script for 'Shoes.app' is displayed:

```
Shoes.app do
  para "Testing test test. ",
  "Breadsticks. ",
  "Breadsticks. ",
  "Breadsticks. ",
  "Very good."
end
```

On the right, a window titled 'Shoes' is running, displaying the output of the script: "Testing test test. Breadsticks. Breadsticks. Breadsticks. Very good."

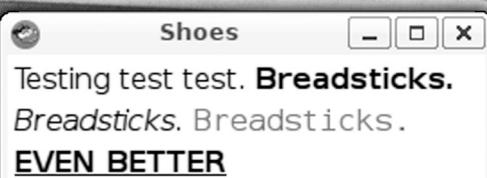
**No need to give para any coordinates or any size.
It'll fill up to the edges of any box it is placed inside.**

See, in the sample up there, para fills the window.

**Also notice how you
can give para a bunch
of strings and it'll
glue them together
as a long string of
sentences.**

Just like on the web, paragraphs can have bolded or emphasized or typewriter styles of text. And, you know, links and strikes, whatever you like.

```
Shoes.app do
  para "Testing test test. ",
    strong("Breadsticks. "),
  em("Breadsticks. "),
  code("Breadsticks. "),
  strong(ins("Very good."))
end
```



Aside from para, please enjoy a myriad of other text sizes. Absent here is banner, the biggest, at 48px.

```
Title 34px
Subtitle 26px
Tagline 18px
Caption 14px
Para 12px
Inscription 10px
```

```
Shoes.app do
  title "Title"
  subtitle "Subtitle"
  tagline "Tagline"
  caption "Caption"
  para "Para"
  inscription "Inscription"
end
```

Of course, if you just want to set a para to a specific font size, just use :size => 48 (for banner size.)

```
para "Oh, to fling and be flung.", :size => 48
```

STACKS & FLOWS

♦ NKS ♦



A stack. Let's say: of dominoes. A stack of dominoes. Banded together with a rubber band. Add more, the stack grows upward.



A flow, on the other hand, is more like a box of matches. As you fill it up, the matches squeeze in side-to-side. Eventually, it'll fill upward. But only after side-to-side. Sardines, right?

♦ 11 ♦



A **stack of flows**, which is extremely handy. Each flow will fill side-to-side. But since they are stacked, they won't mix with each other.



And more complex mixtures. The main Shoes window is itself a flow. Mixing stacks and flows builds columns.

Oh, and gravity is up. See, the window grows downward.



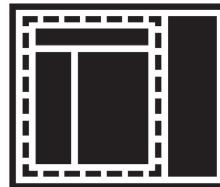
Two columns can be accomplished by placing two stacks inside a flow. And by giving those two stacks a 50% width.



Likewise, three columns can be set up by having three stacks within a flow and splitting the width of each stack in thirds.



And what about a header + three columns? Well, that would be four stacks in a flow. The first stack would have a width of 100%. And the remaining stacks are split three ways.



More complicated designs might need stacks in flows in stacks... and so on. This one's a flow. With a flow and a stack inside. In the left flow (highlighted with a dotted line,) three more stacks: a 100% header and two columns.

One Quick Note About Widths

Widths may be a positive or negative number of pixels.

```
stack :width => 80 do; end
stack :width => -80 do; end
```

You'll often see this. The first stack is 80 pixels wide. And the second is 100% minus 80 pixels wide. So, together they are 100%, they fill their parent box.

You might also see floating point widths. 1.0 is 100%, 0.9 is 90%, 0.8 is 80% and so on.



I'm sure the question you're
wondering now is: shouldn't
there just be vertical stacks
and horizontal stacks??

But the thing is: Shoes
doesn't believe in a horizontal
SCROLLBAR!!!

So flows hit the end
and move down...





“One and two and...
Hey, Midas, wait up!!”



“Check me out! I’m doin’ my
stacks and flows!... five-hippo-
potamus... six-hippopotamus...”



“See, my gravity is even like
a total window application, guy!”



“Yeah, pretend my foot is the
close button and my leg warmer is
like minimize or something
because I’m just about to
totally crash.”



DANCING AROUND THE STACKS & FLOWS

Using X & Y.

Using Top & Left.

Clearly, stacks and flows are only for packing things in nice and tight. Columns. Grids. Flowing text and images. Making pages which look much like web pages.

But anything can be positioned at specific coordinates using :top and :left in the style options. And you can still do stacks and flows.

```
Shoes.app do
  @o = oval :top => 0, :left => 0,
             :radius => 40

  stack :margin => 40 do
    title "Dancing With a Circle"
    subtitle "How graceful and round."
  end

  motion do |x, y|
    @o.move width - x, height - y
  end
end
```

Run this little toy, yeah? And while it's running, pass your mouse over the window and watch the circle dance with you.

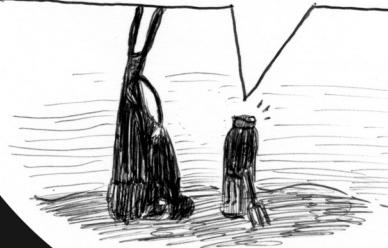
This window is three things: a circle, a stack of words, and a motion event block. The oval is in the main window's flow. But it's floating freely at position (0, 0). The upper left corner of the window.

The motion event gets called whenever you move the mouse. The circle then gets moved based on your mouse coordinates. (The **width** and **height** methods get us the window's width and height. By subtracting the mouse position, it gives the illusion of the circle moving to the opposite side of the window from the mouse.)

Resize the window. Jiggle the mouse. You see? It's all okay?



DO YOU MEAN BUTTONS AS IN
button("PRESS ME")
WHICH CREATES A NEW BUTTON AND
button("PRESS ME") { alert("licked") }
HOW THE BLOCK FIRES WHEN CLICKED AND
button("PRESS ME", :top => 40, :left => 40)
WILL PLACE THE BUTTON AT COORDINATES
(40, 40) AND YOU CAN SET WIDTH AND
HEIGHT AS WELL - THAT KIND
OF BUTTON ??

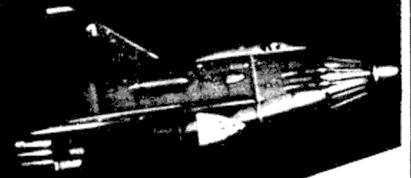


NOW SAY,
"the end."

WHAT?
NO!

okay, well...
the
end.

AN ENCOUNTER OF THE FIFTH KIND



Tonight, images are number five
in your list of Shoes to be uncovered.

You feel good. Is it Saturday already? Well, no, this is just the very easy page about images. Two easy sections in a row, how about that? Like the button, all you really need to remember about images is the word "image." That's the name of the method used to put an image in your Shoes code. The image will be positioned right inside the stack or flow.

The "image" method isn't the only way to display an image. The "background" method can also be used to position images and even tile them.



WTF OFF

Q: Suzanne, my images are all over the place! I want the image of a hot dog to be ON TOP of my image of Bin Laden. Help me!

A: That is a GREAT combination! Images stack in the order you have them in the code. Later in the code means above everything else. Perhaps one day Shoes will have z-ordering, but for now it's just dead plain dumb!

Q: How do I swap an image's for another pic?

A: Swapping is so fun! Try changing the image's path property. See YOU later!

THE SIGHT OF SATURN



you have learned so much
but this is your favorite part
this whole spread here

SURE YOU WANT
TO HACK IN YOUR
PICTURES.



you may use gif, jpeg or png.

or something like:

image "guy.png", :top => 100, :left => 100

WHO CALLED THESE LASERS LITTLE?



Perhaps they ARE little, thanks to
image scaling using :width and :height.



The Shoes distro comes
with a few great samples
of using images. In
particular: both bounce.
rb and form.rb in the
samples directory.

Shoes.app do
image "j.jpg"
end

Shoes will try to load
the image from the cur-
rent folder. Unless a full
path is given, that is.

SUPER HELMET TIME!



Are you starting to see how Shoes
is advanced technology? Like helmets.
Like gloves. Like cups. Like dogs!



NOBODY KNOWS
SHOES

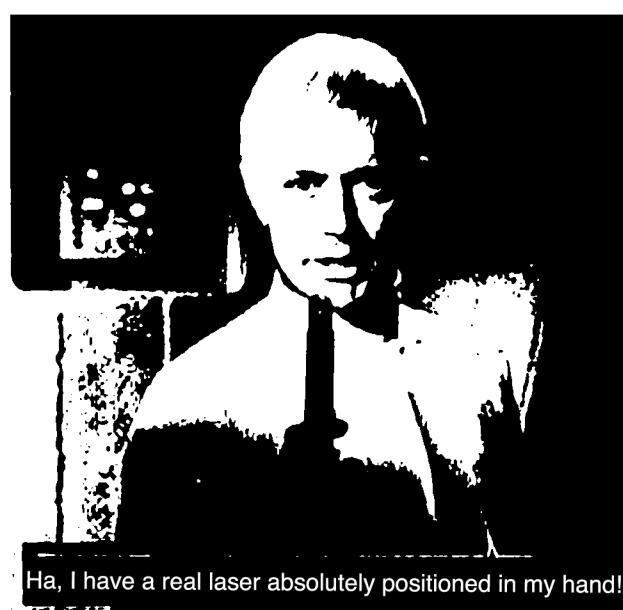
Evacuate the ship. I am doing my image practice in here.



Isn't that just soooo science fiction of you to go bossing us around. I have my own PNGs and JPEGs to try out!



I've had it to here with you. If I had an image of a laser gun I would absolutely position it right here in my hand...



Ha, I have a real laser absolutely positioned in my hand!



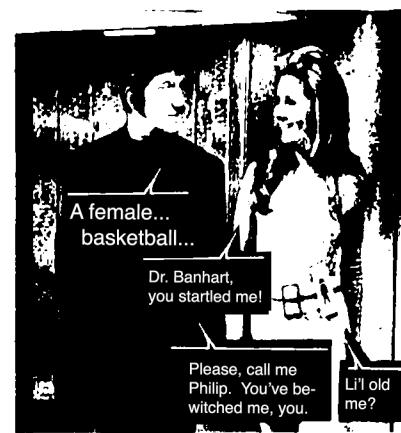
You know, being in a comic is like being in a flow of images. I'd say your dead body's gonna cause a vertical scrollbar!



Look, everyone, his laser gun is making its warm-up sound...



Good God.



Please, call me Philip. You've bewitched me, you.

Li'l old me?

AND SO, AMID SOUNDS OF THE DOCTOR'S PIERCING EYES, SHE LEARNED FROM HIM OF SHOES' SPECIAL ABILITY TO GET AN IMAGE'S ORIGINAL WIDTH/HEIGHT USING THE IMAGE.SIZE METHOD (like: $w, h = @image.size$) I'M SORRY IF YOU KNEW THAT ALREADY, I GUESS YOU DIDN'T LEARN ANYTHING ON THIS PAGE. BEST OF LUCK IN THE FUTURE.





LET'S HAVE A WARM WELCOME

“THE EDIT LINE”

NOT TO BE CONFUSED WITH THE SO
YOUNG AND TALENTED MS. HAYLEY MILLS



AN AUTHENTIC AND OFT-REVERED RECTANGLE FOR TYPING IN ONE'S DESIRES AND DREAMS!



HERE SHE IS, UNADORNED, NOTHING MORE THAN

```
Shoes.app do
  @e = edit_line :width => 400
end
```

A NATIVE CONTROL: LIKE THE “BUTTON,” BOTH FOLLOW THE CLOAK &
ATTIRE OF THE OPERATING SYSTEM - THE ABOVE IS THE OS X EDIT LINE.



NOW, FRIENDS, WE WAIT.

Septuplets.

LO AND BEHOLD! SOMEONE HAS USED THE EDIT
LINE, NOT AS PART OF ANY TRICKERY, BUT TO
UNVEIL THEIR DESIRES AND DREAMS!

WHAT CAN BE DONE?

```
Shoes.app do
  @e = edit_line :width => 400
  button "O.K." do
    alert @e.text
  end
end
```

Oh, here we go. The edit line is stored in the @e instance variable. When
the button is clicked, we get the typed words inside the edit line using
@e.text. And to change the edit line, use @e.text = “Owls.”



NOW, FRIENDS, PAGE NEXT.

HELLO, CLOSE COUSIN

“THE EDIT BOX”

A TRUE BOX, SPARING NO ABSENCE OF CORNERS
AND FOR ALL CHILDREN EVERYWHERE!



JUST A BUNCH OF LINES RATHER THAN ONE

Would that I could travel in reverse, to a simpler time, if only to engineer a septuplet of young and talented Hayley Mills... and ask, "You don't need all these, do you?"

AND THE WORDS INVOKING THE SPELL ARE

```
Shoes.app do
  @e = edit_box "Would that I...",
    :width => 400, :height => 240
end
```

The edit_box has its very own vertical scrollbar it may deploy.

The edit_line method as well can take a string, if you want the box to come pre-filled.



INSTANCE VARIABLES IN SHOES.APP

A verbose way of writing many of these programs is to use “self”.

```
Shoes.app do
  self.stack do
    self.edit_line "Sample sample."
    self.button "Breadsticks."
  end
end
```

Were you to inspect self in any block, you'd get #<App:0x64eb94>. So, yeah, all methods run against the Shoes App object.

Instance variables are a good place to store Shoes controls and objects, since they will be kept in the app and can be yanked at from anywhere inside the app's scope.

**LINKS.
HYPERLINKS, I GUESS.
STOLEN FROM THE WEB WEB WEB.**

**link “CLICK ME” do; alert “HEY”; end
EXCEPT YOU CAN’T DO THAT.
I MEAN, YEAH, THAT’LL BUILD A LINK.
BUT IT WON’T SHOW THE LINK.
HAS TO BE IN A PARAGRAPH OR SOME OTHER
KIND OF TEXT BLOCK. (SUCH AS BANNER, TITLE,
SUBTITLE, ETC.)**

**para(link(“CLICK ME”) { alert “HEY” })
YES, THAT IS PRECISELY IT.
TRY DOING THAT FROM NOW ON.**

**link(“GOOGLE”, :click => “http://google.com”)
OR YOU CAN SKIP THE BLOCK AND HAVE THE CLICK
TAKE PEOPLE TO A URL. WEB URLs OPEN UP A
WEB BROWSER.**

**AND SHOES LINKS?
WE’LL GET TO SHOES LINKS SOON ENOUGH.**

**WHAT ABOUT IMAGE LINKS?
DON'T PUT AN IMAGE IN A LINK.
FOR THAT MATTER, DON'T PUT AN IMAGE IN A PARA.
USE FLOWS.**

**image “google.png”, :click => “<http://google.com>”
THAT'S AN IMAGE LINK.
OR, IMAGES ALSO TAKE A CLICK BLOCK.**

**style(Link, :underline => false, :stroke => green)
YOU PROBABLY LIKE CHANGING YOUR LINK STYLES.
THIS CHANGES ALL LINKS IN THE WINDOW TO AN
UNDERLINELESS GREEN.**

**style(LinkHover, :underline => true, :stroke => red)
AND THE STYLE OF A LINK SOMEONE IS HOVERING
OVER WITH THE POINTER.**

NOW, LET'S SEE IT IN A SENTENCE.

**para “Jimmy kissed ”, strong(“Margaret”),
“ and I've saved a picture somewhere on ”,
link(“Flickr”, :click => “<http://flickr.com>”)**



Jenny saved her dimes and placed a call to the Shoes operator.



Hello, ma'am.
Thank you for taking my call.

What's a background?



Good to hear from you,
Jenny. How'd you do
on your comprehension
quiz this week?



A background is:

- a color
- a gradient
- or, an image

All three of these things can be made into a Shoes "pattern". And these patterns are the nature of your phone call, Jenny. You have taken a great interest in them, I can tell.

Backgrounds and borders are both just patterns painted in or around a Shoes box. And, yes, a box may have many backgrounds and many borders.

backgrounds & borders

```
stack do
  background "images/pencils.png"
  background "#FF9900"
  background rgb(192, 128, 0)
  background gray(0.5)
  background red
  background "#DFA".."#FDA"
end
```

Backgrounds will tile the entire width & height of a box, so you will need to constrain the :width & :height styles (or alter :top or :left) to change the backgrounds surface area.

```
stack do
  # first, a gray background
  background gray(0.5)
  # then, an image tiled across the window's top 50 pixels
  background "top.png", :height => 50
  # and tile the left starting at (0, 50) and 55 wide
  background "left.png", :top => 50, :width => 55
  # now, tile the rightmost 55 pixels
  background "right.png", :right => 0, :top => 50, :width => 55
  # and place a bottom left corner image
  image "corner.png", :bottom => 0, :left => 0
end
```

Border thickness is controlled by the :strokewidth style, give it a pixel count, okay?

```
border black, :strokewidth => 5
border "#DDD".."#AAA", :strokewidth => 10
border "images/stripes.png", :strokewidth => 4
```



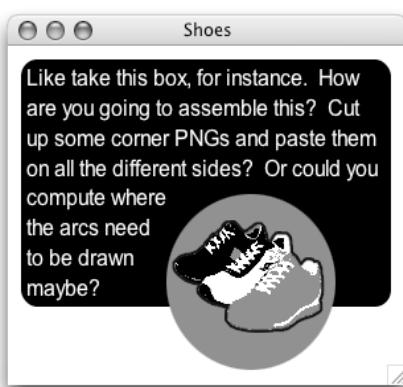
✓ HILARY THE EXERCISER PRESENTS

SMOOTH CORNER CUTS

a background trick



Okay, so on the web web, you see a lot of rounded corners, right? Like a box with words in it and the edges are sanded smooth.



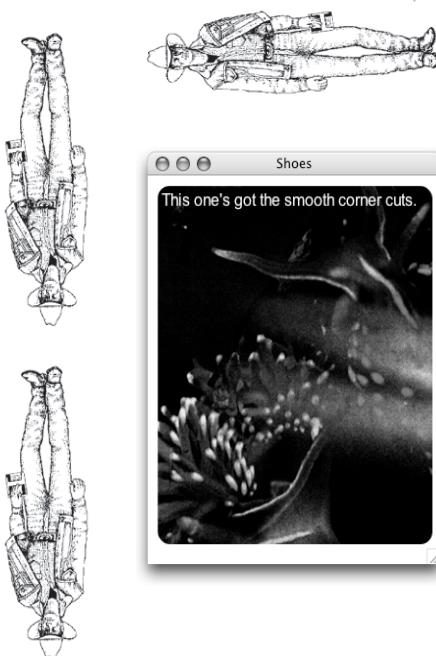
Oh, well, this is going to be too quick. 

Let's say the background is going to be:



Which code is:

```
stack :width => 300, :height => 350 do
  background "murky.seas.png"
end
```

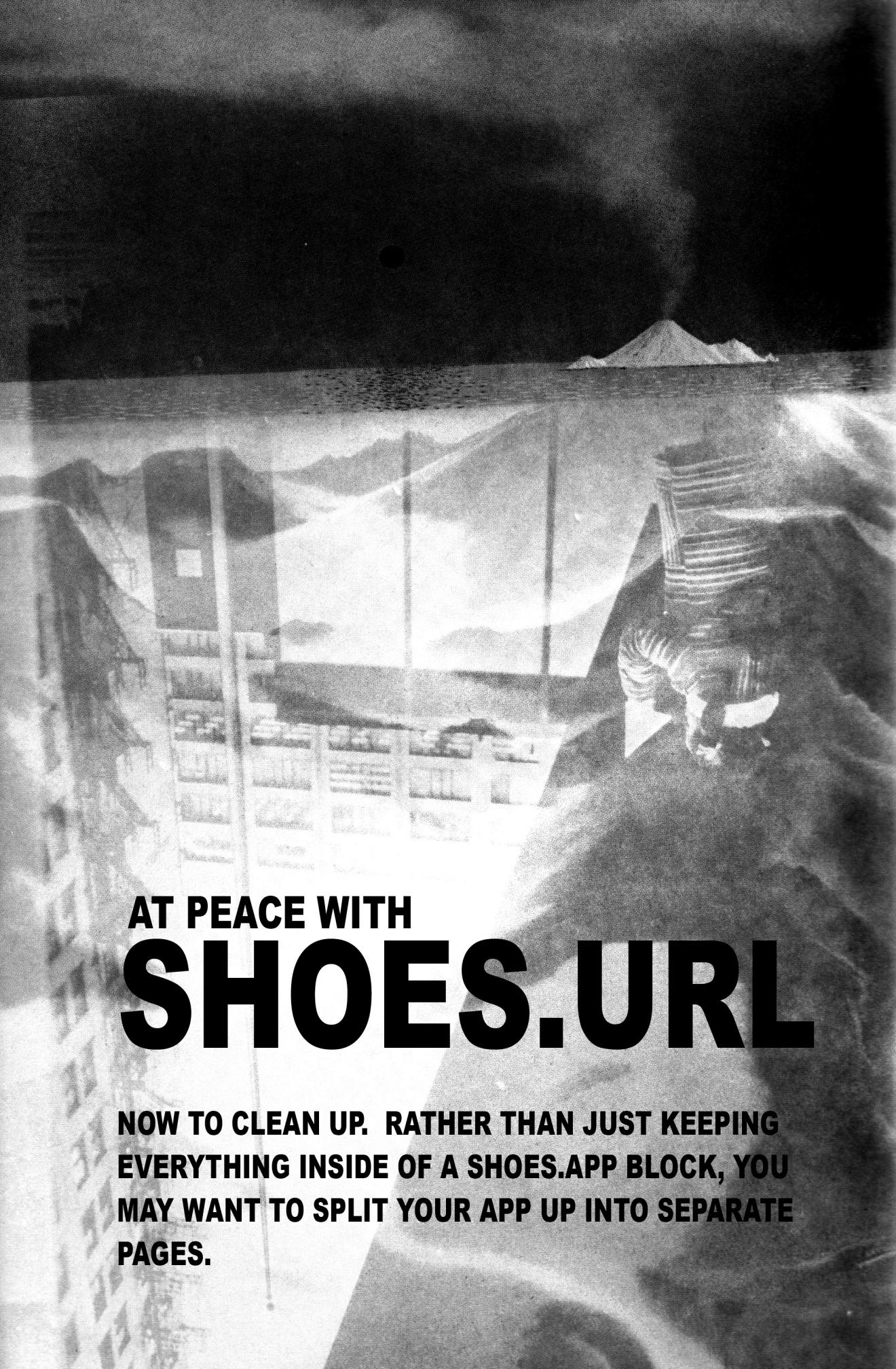


Just give the background a radius.
(Yeah, it's a radius for the corners.)
Twelve pixels looks pretty decent.

```
stack :width => 300, :height => 350 do
  background "murky.seas.png",
  :radius => 12
  para "This one's got the smooth ",
  "corner cuts.",
  :stroke => white
end
```

SHOES SMOOTH CORNER CUTS





AT PEACE WITH **SHOES.URL**

NOW TO CLEAN UP. RATHER THAN JUST KEEPING EVERYTHING INSIDE OF A SHOES.APP BLOCK, YOU MAY WANT TO SPLIT YOUR APP UP INTO SEPARATE PAGES.

To start off, you’re not going to pass a block to Shoes.app. Instead, you’re going to subclass Shoes and keep each of your pages in its own method.

```
class BookList < Shoes
  url '/', :index
  url '/twain', :twain
  url '/kv', :vonnegut

  def index
    para "Books I've read: ",
      link("by Mark Twain", :click => "/twain"),
      link("by Kurt Vonnegut", :click => "/kv")
  end

  def twain
    para "Just Huck Finn.\n",
      link("Go back.", :click => "/")
  end

  def vonnegut
    para "Cat's Cradle. Sirens of Titan. ",
      "Breakfast of Champions.\n",
      link("Go back.", :click => "/")
  end
end

Shoes.app :width => 400, :height => 500
```

Yes, it’s true. You really need that Shoes.app at the bottom. That’s what opens your window!

So, since the BookList class is descended from the Shoes class, it has a “url” method. And so we use the “url” method to snatch three URLs: /, /twain and /kv. This class represents those three “places.” (A link travels you to that “place.”)

The “/” URL will draw the page in the “index” method. Which contains links to “/twain” and “/kv”, which draw the pages inside the other two methods. As usual: run it, try it out.

If you choose, you may also keep your methods organized into different classes. Shoes will use whichever class answers to a URL (maybe from a link which has been clicked.)

Say, do you know regular expressions at all? Because you can drop a regular expression into a Shoes.url() call and it'll intercept any URLs which match.

```
class Dictionary < Shoes
  url '/', :index
  url '/(\w+)', :word

  def index
    stack do
      title "Enter a Word"
      @word = edit_line
      button "OK" do
        visit "/#{@word.text}"
      end
    end
  end

  def word(string)
    stack do
      para "No definition found for #{string}. ",
            "Sorry this doesn't actually work."
    end
  end
end

Shoes.app
```

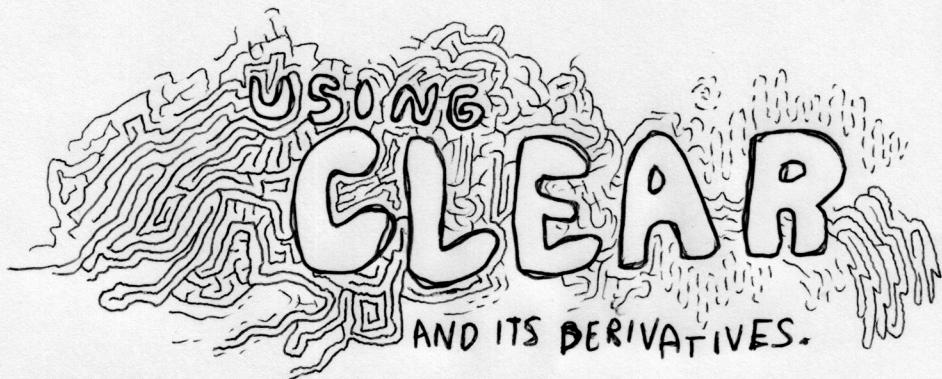
Word of warning: use single quotes around the URL expressions! Otherwise, the backslash won't really work and you'll see a 404 NOT FOUND message from Shoes.

Notice how the word matched by the `(\w+)` expression is sent as the first argument to the “word” method. Each regular expression group which is found is sent to the method as an argument.

So, if you'd like to see a more complex example, see samples/book.rb in the Shoes distro. It's a little short story reader.

On the Shoes wiki, see the “Multi Page” link on the front page for some continuing studies of this very engaging matter.

hey, it's the last part.



after you have your whole page set up,
you may want to alter things & change things
up. the "clear" method wipes a box:

`@box.clear`

(assuming `@box` = stack {} or something.)

"clear" also takes a block, if you want to fill
the box back up with elements.

`@box.clear do`

`para "Please type:"`

`edit-line :width => 240`

`end`

oh and don't use "clear" to hide and show boxes.
use `@box.hide` and `@box.show` for that.

OTHER GOOD ONES:
**APPEND, PREPEND,
BEFORE & AFTER**



`@box.append {}` adds to the end of the box, and
`@box.prepend {}` adds to the beginning of the box.

`@box.before(element) {}` adds to the beginning of
a box, right before its child "element"

`@box.after(element) {}` adds immediately after
a child "element"

none of these methods clears the box, obviously, so
to remove specific elements, call `element.remove`.

a short & append / remove SAMPLE

now, a quick demonstration of a self-altering set of Shoes stacks.

notes.Rb

```
Shoes.app do
  stack :margin => 20, :width => 250 do
    subtitle "Shoes Notebook"
    para "Add a note."
    @add = edit-line

    button "Save" do
      @notes.append do
        para @add.text, ""
        link("delete") { |x| x.parent.remove? }
      end
      @add.text = ""
    end
  end
  @notes = stack :margin => 20, :width => -250
end
```

TRY THIS OUT. Run it. Add a note. And click the "delete" link to remove a note.

Pay special attention to the use of "append". It's appending to the @notes stack. What is it appending? A para!

And what happens when the link is clicked? The link's parent is removed. And the link's parent is? That same para!

Now, let's sum up-

The Entirety of THE SHOES FAMILY?

shortcuts: [brackets] mean "optional",
:symbols always refer to a style setting, »
means a returned object, "box" means a
stack or flow, for "pattern" and "styles"
arguments: see the very end of the page
adjacent. all methods run in the
Shoes.app scope.

the PAINTBRUSH

`stroke(pattern) » pattern`

sets the line brush

`nostroke() » box`

turns off the line brush

`fill(pattern) » pattern`

set the fill brush

`nofill() » box`

disables the fill brush

`strokeWidth(pixels) » box`

set the thickness of the line brush

the SHAPES

`star(x, y, [p], [r], [inner]) » Shoes::Shape`

draws a star with [p] points of [r] radius

`arrow(x, y, size) » Shoes::Shape`

draws an arrow with a [size] tip

`line(x1, y1, x2, y2) » Shoes::Shape`

draws a line with the line brush

`oval(x, y, w, h) » Shoes::Shape`

`oval(styles) » Shoes::Shape`

draws a circle or ellipse with the two brushes

`rect(x, y, w, h) » Shoes::Shape`

`rect(styles) » Shoes::Shape`

draws a rectangle with the line & fill brushes

the TEXT BLOCKS

`para("string", [styles]) » Shoes::Para`

banner, title, subtitle, tagline,

caption, inscription

the CONTROLS

`button("string", [styles]) { ... } » Shoes::Button`

`edit_line([styles]) { ... } » Shoes::EditLine`

`edit_line :text => "string"`

`edit_box([styles]) { ... } » Shoes::EditBox`

`edit_box :text => "string"`

`list_box([styles]) { ... } » Shoes::ListBox`

`list_box :items => ["shoes", "coat", "hat"]`
a drop-down select control of all :items

`progress([styles]) » Shoes::Progress`

a progress bar displaying :text as a message

the IMAGES

`image(path, [styles]) { ... } » Shoes::Image`

any JPEG, PNG or GIF

`video(path, [styles]) { ... } » Shoes::Video`

plays FLV, MPEG, DivX, Xvid, web URLs are okay

the BOXES

`stack([styles]) { ... } » Shoes::Stack`

`flow([styles]) { ... } » Shoes::Flow`

`background(pattern, [styles]) » Shoes::Background`

`border(pattern, [styles]) » Shoes::Border`

COMMON METHODS

COLORS

`rgb(r, g, b, [a]) » Shoes::Color`
a color, a mix of red-green-and-blue (and perhaps alpha), use 0 to 255 for each color part. or 0.0 to 1.0, if you prefer.

methods such as these are available on all or some of the objects discussed up to this point.

`gray(b, [a]) » Shoes::Color`
a color of gray with [b] brightness and [a] alpha
`white, black, red, yellow, blue, ...`
some color methods; see "Color List" on the Shoes wiki or http://en.wikipedia.org/wiki/X11_color_names for a full list.

the DIALOGS

`alert(msg) » nil`
`ask(msg) » "string"`
`confirm(msg) » true or false`
`ask_color(msg) » Shoes::Color`
`ask_open_file() ask_save_file()`
» "/path/filename.ext"

VISIBILITY

`hide(), show(), toggle() » self`
for EVERYTHING (yes: SHAPES, TEXT, CONTROLS, BOXES and IMAGES)

ALTERATIONS

`move(x, y)`
`size(w, h)`
`remove()`
for EVERYTHING
`style([styles])`
adds or changes styles for any element

BOXES ONLY

`contents() » Array`
`clear() { ... } » box`
`after([ele]) { ... } » box`
before, append, prepend

STYLES & PATTERNS

a "pattern" may be a color, a range of colors, an image path, an Image object or a Pattern obj.

ex.: "#DDFFAA"
"#000".."#FFF"
(a gradient of black to white)

`rgb(0.9, 1.0, 0.7)`
"/images/bg.png"

Hash of various options.

a "styles" argument is a Ruby

:top => 40, :left => 40
:font => "Arial 12px"

PATTERNS

`gradient(c1, c2) » Shoes::Pattern`
a gradient from [c1] color to [c2] color

EXTRA

`self.mouse() » [x, y]`
`self.clipboard » "string"`
`self.clipboard=(string)`
`visit "/url"`
`exit`

Please join us on the Shoes mailing list.
Send an e-mail.

TO: shoes@code.whyluckystiff.net
CC: why@whyluckystiff.net

A computer is standing by to send you
instructions.





MIDAS KNOWS SHOES