

On course materials

a workshop for L&S TA training

William C. Benton
University of Wisconsin

willb@cs.wisc.edu
<http://pages.cs.wisc.edu/~willb/>

About me

This talk: “why” and “how”

This talk: ~~“who,” “what,” and “how”~~ “why,” “what,” and “how”

Why make course materials?

Forecast

- **When and how to use slides**
- **Why and how to make handouts**
- **How to make your life easier**
- **Teaching the Facebook generation**

Forecast

- When and how to use slides
- Why and how to make handouts
- How to make your life easier
- Teaching the Facebook generation

What makes for good slides?

What makes for bad slides?

- One characteristic of bad slides is that they often have too much text or serve as an outline for a lazy speaker who has failed to adequately prepare, meaning that the audience will be reading instead of listening
 - Also feature many sentence fragments
- Because font size is too small, readability suffers, key points not reinforced, audience asleep
- Fortunately, you can print this out and read it later, gathering much of benefit of attending talk
 - Unfortunately, slides like this make you wonder why you are bothering to attend the presentation in the first place

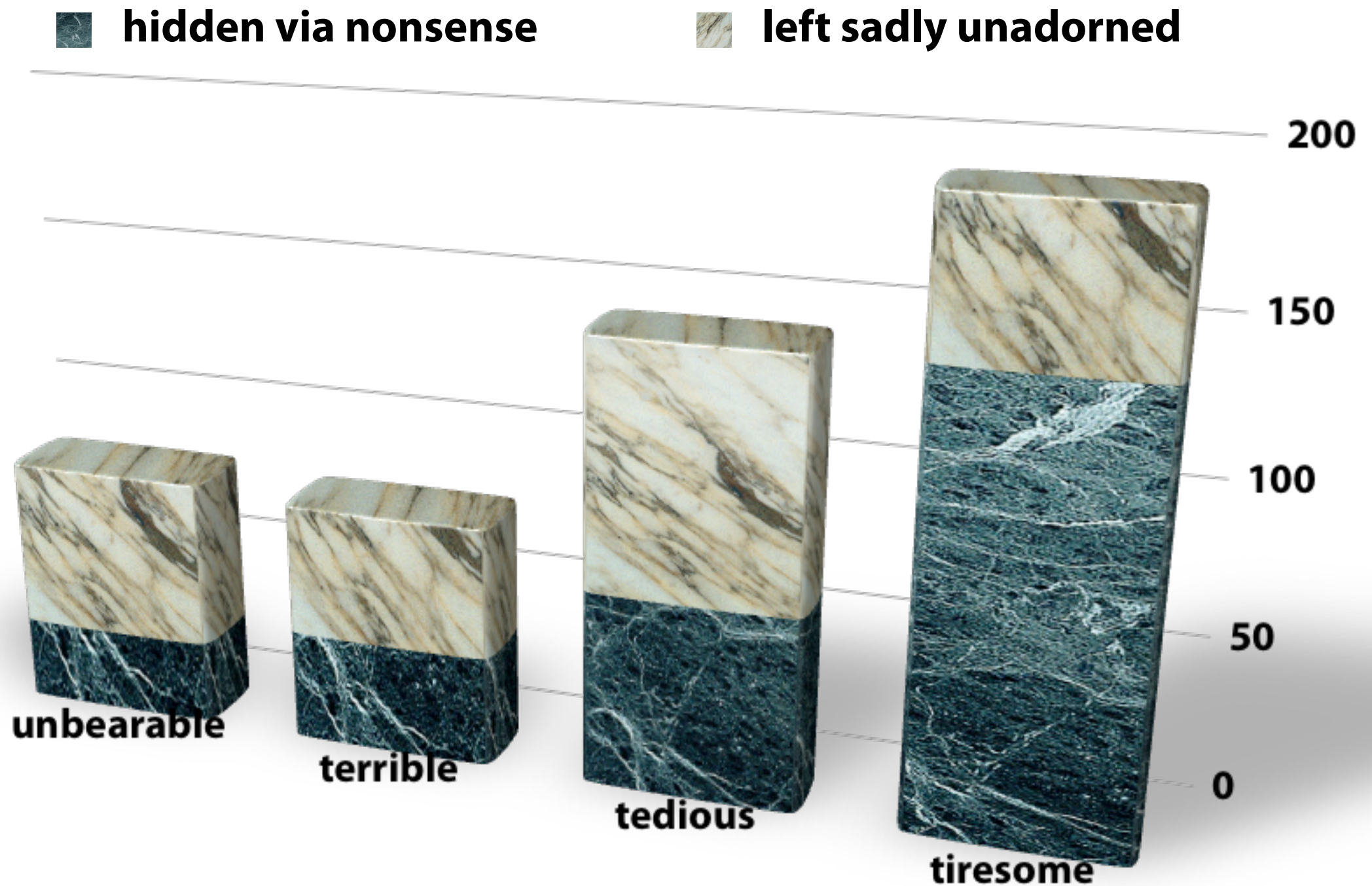
Other kinds of bad slides?

- **Sure!**
 - **Many ways**
 - **Too terse perhaps as bad as too verbose**
- **Kind of too bad that “verbose” is on a line by itself up there; is it lonely?**

Other kinds of bad slides? (ctd.)

- **Does this help my presentation?**
 - **Not really.**
- **Inconsistent punctuation is sort of fun**
- **What information do these slides convey?**
- **Why aren't these bullets in my notes?**

Trying to hide bad slides (2007)





Yikes!

Bad slides are ubiquitous


Engineering and Work Practice Controls (con't)

- The employer must:
 - Train employees to use new devices and/or procedures
 - Document in ECP



Results

■ Attitudes	Agree (%)
- People choosing to live near forests accept risks	91
- Homeowners should follow gov't guidelines to manage for wildfire risk	80



PROTECTING AMERICA'S WORKFORCE
THE FY06 BUDGET

2004 Highlights

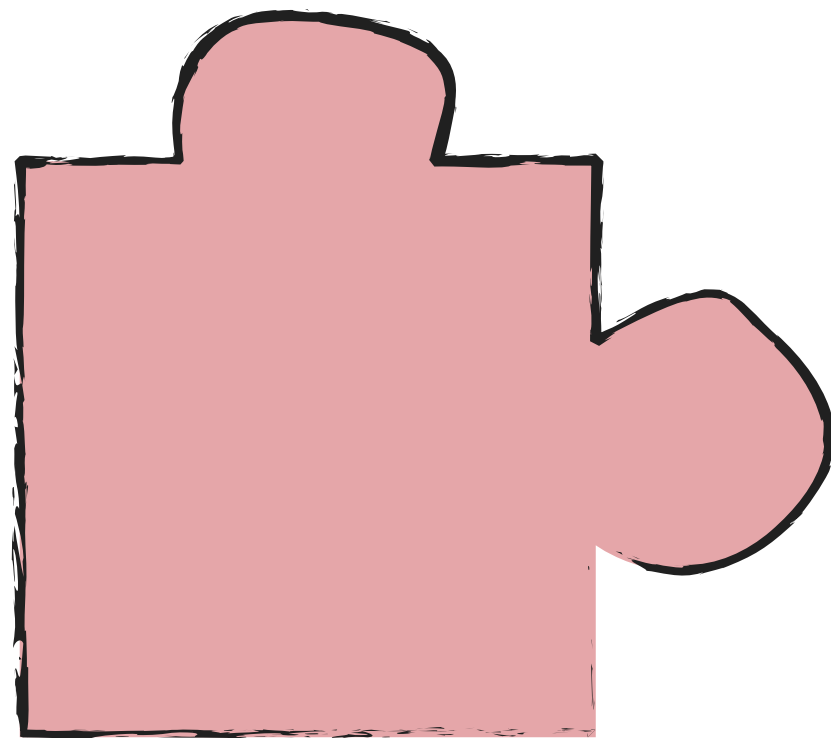
- Recovered nearly \$200 million in back wages for 265,000 workers
- Secured monetary results of over \$3 billion for workers' pension and health plans
- Implemented innovative training programs to prepare workers

3

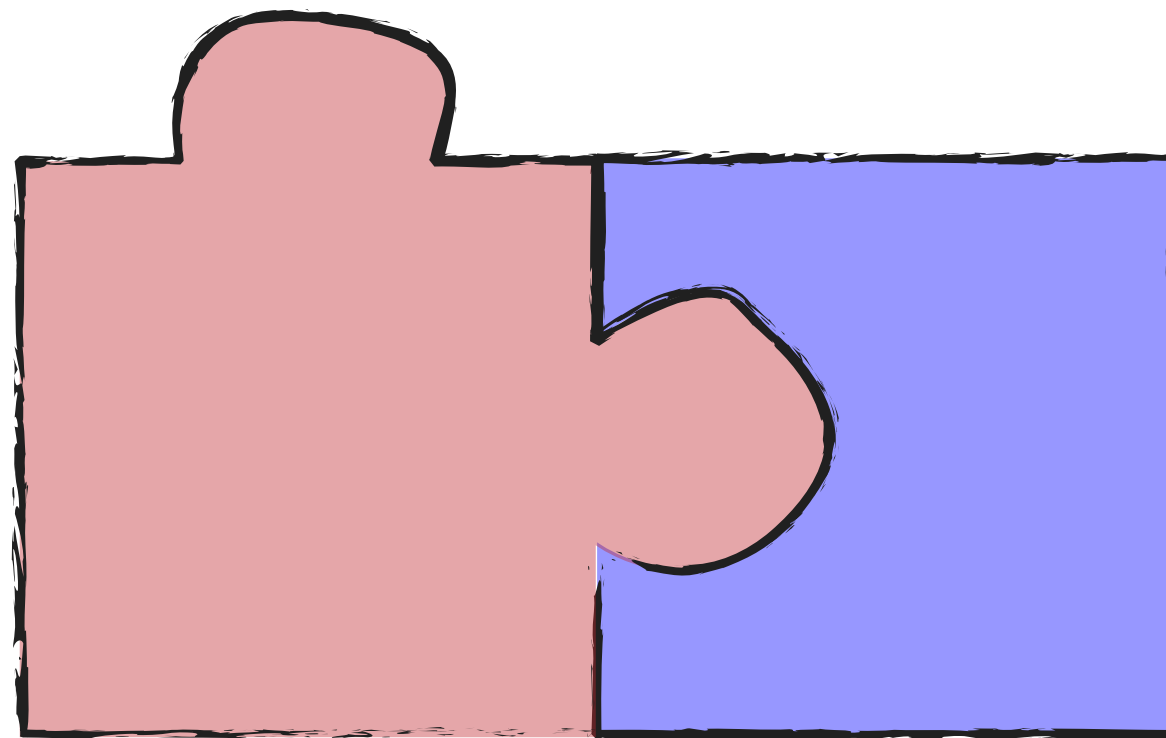
("credits": OSHA, USDA, DOL)

A better model

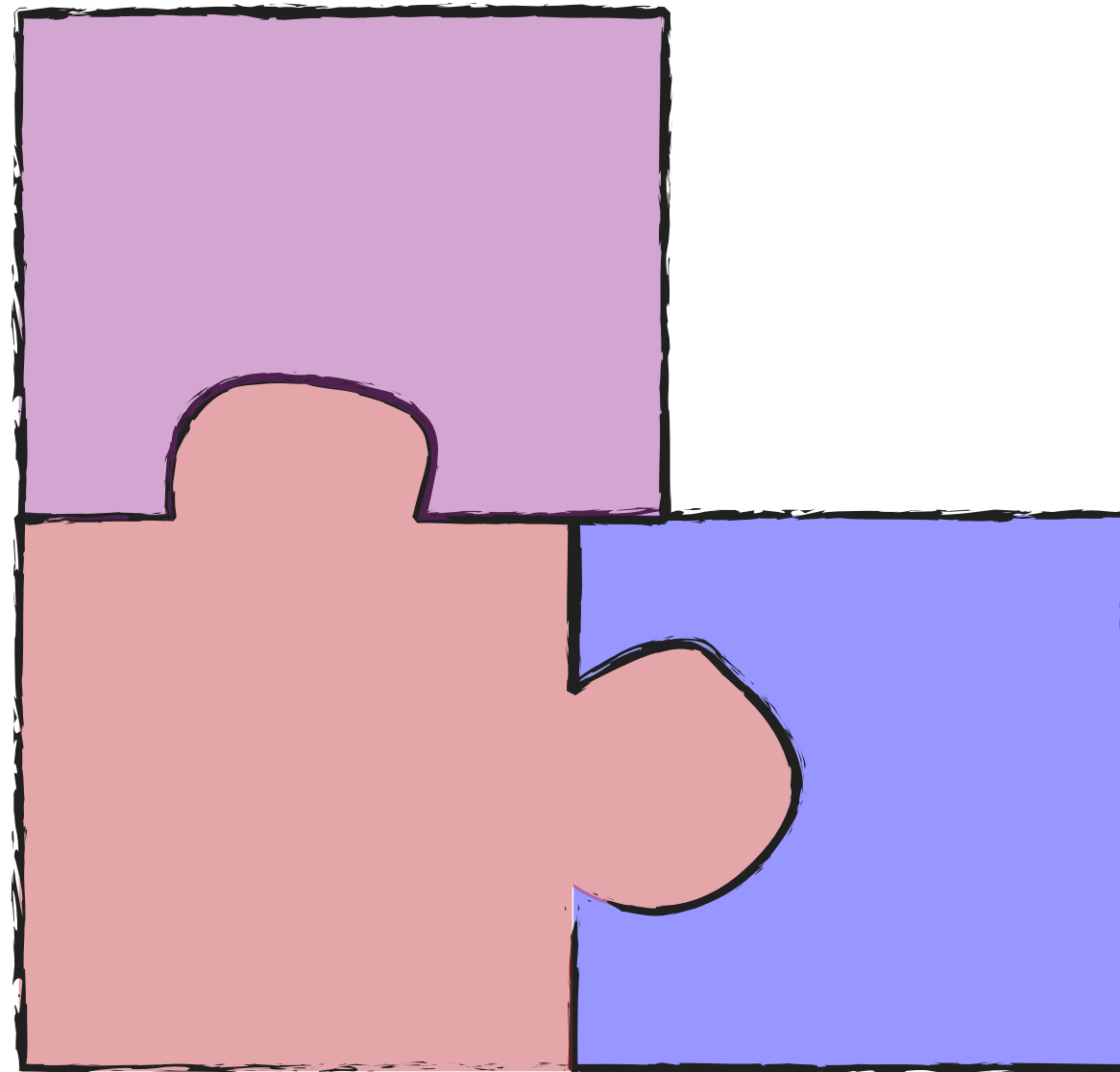
A better model



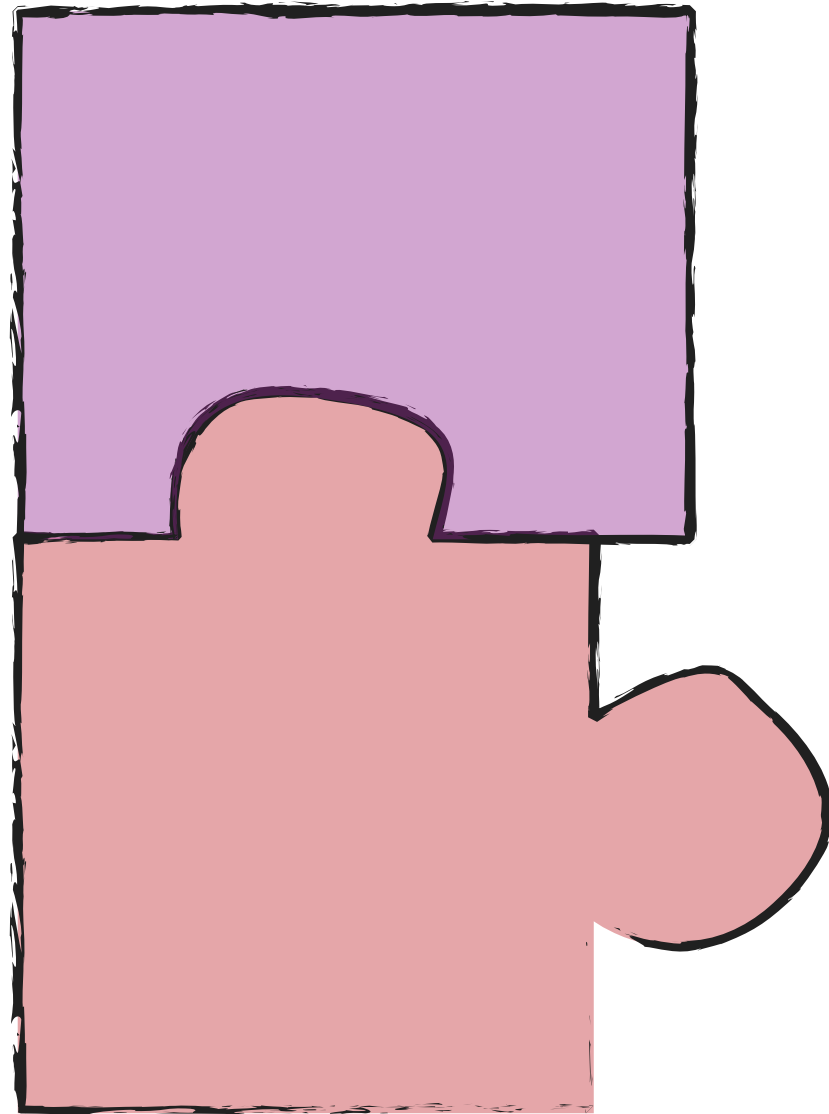
A better model



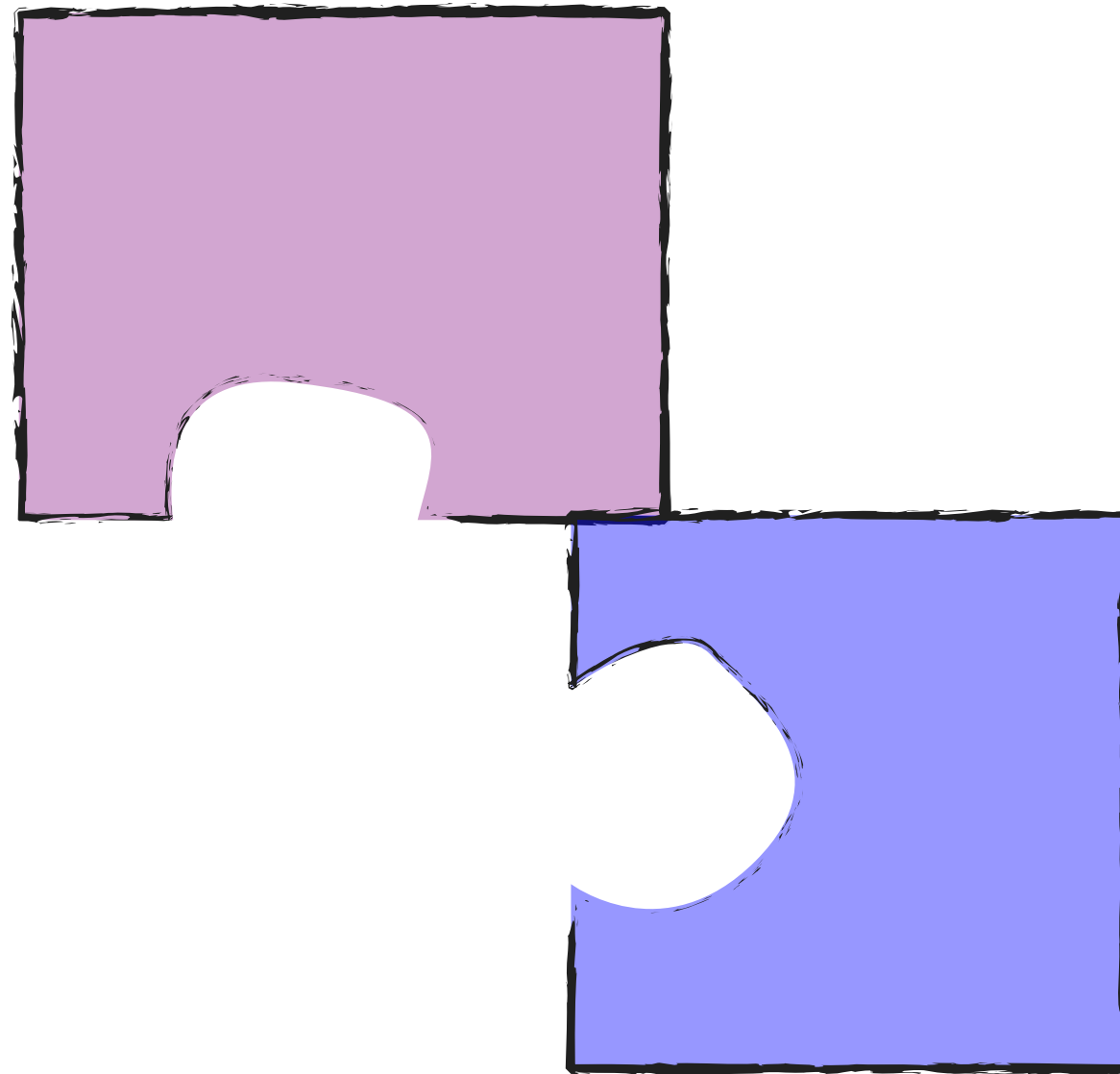
A better model



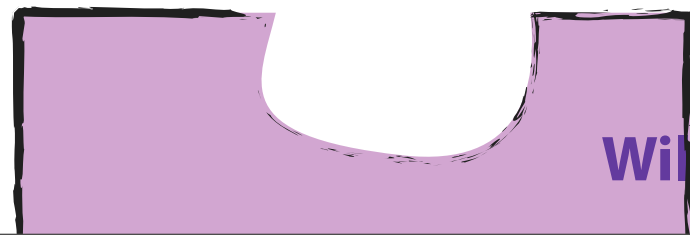
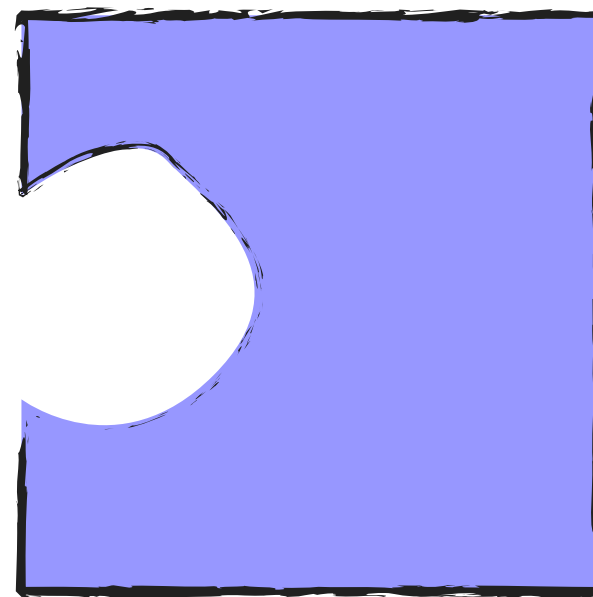
A better model



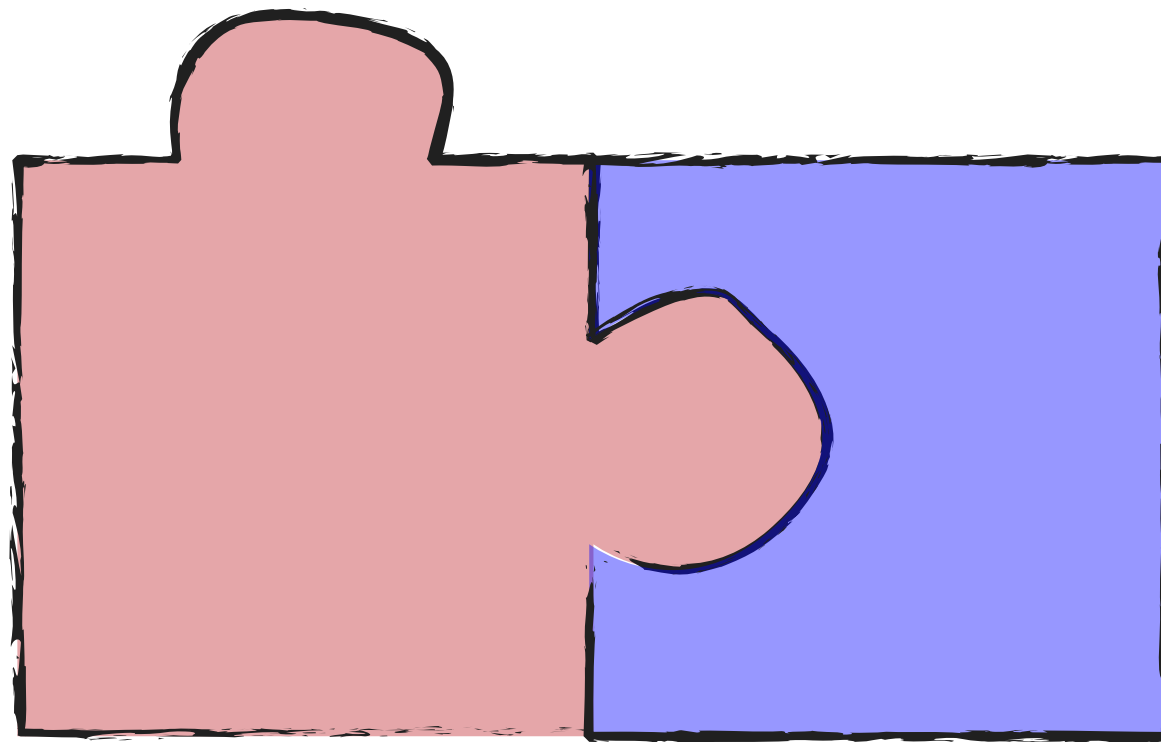
A better model



A better model



What should slides do?



Case study: figures

Allegro con brio.

Flauti.

Oboi.

Clarinetti in B.

Fagotti.

Corni in Es.

Corni in Es.

Timpani in C.G.

Violino I.

Violino II.

Viola.

Violoncello.

Basso.

ff

p

DA-DA-DA
DAAAH!

Flauti.

Oboi.

Clarineti in B.

ff

Fagotti.

p

Corni in Es.

Corni in Es.

Timpani in C.G.

Violino I.

ff

p

Violino II.

ff

p

Viola.

ff

p

Violoncello.

ff

p

Basso.

ff

DA-DA-DA
DAAAH!

BASSOONISTS
WAKE UP HERE

DUH-DUH-DUH
DUUUH!

Flauti.

Oboi.

Clarineti in B.

ff

Fagotti.

p

Corni in Es.

Corni in Es.

Timpani in C.G.

Violino I.

ff

p

Violino II.

ff

p

Viola.

ff

p

Violoncello.

ff

p

Basso.

ff

Detailed description: This is a musical score for a symphony, likely by William C. Benton, in 2/4 time with a key signature of two flats (B-flat and E-flat). The score is for a full orchestra, including Flauti, Oboi, Clarineti in B, Fagotti, Corni in Es, Timpani in C.G., Violino I, Violino II, Viola, Violoncello, and Basso. The score is divided into two systems. The first system shows the initial entry of the strings and woodwinds. The second system shows the bassoonists entering with a new melody. Handwritten annotations in speech bubbles provide a humorous commentary on the music. The first bubble, 'DA-DA-DA DAAAH!', points to the initial notes of the Clarineti in B. The second bubble, 'BASSOONISTS WAKE UP HERE', points to the entry of the Fagotti. The third bubble, 'DUH-DUH-DUH DUUUH!', points to the entry of the Corni in Es. The score includes dynamic markings such as *ff* (fortissimo) and *p* (piano). The woodwinds and strings are playing a rhythmic pattern of eighth notes, while the brass instruments are playing a more melodic line.

Case study: animated processes

```

public class Foo {
    private Foo f;
    private int k;

    public Foo(int k) {
        this.k = k;
        this.f = this;
    }

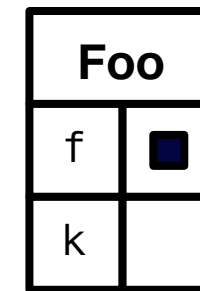
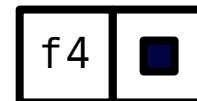
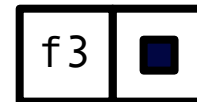
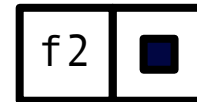
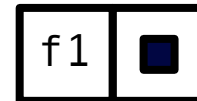
    public void sF(Foo f) {
        this.f = f;
    }

    public static void main(String args) {
        Foo f1, f2, f3, f4;
        f1 = new Foo(1);
        f2 = new Foo(2);
        f3 = new Foo(3);
        f4 = new Foo(4);

        f2.sF(f4);

        f4 = f1;
        // BANG
    }
}

```



```

public class Foo {
    private Foo f;
    private int k;

    public Foo(int k) {
        this.k = k;
        this.f = this;
    }

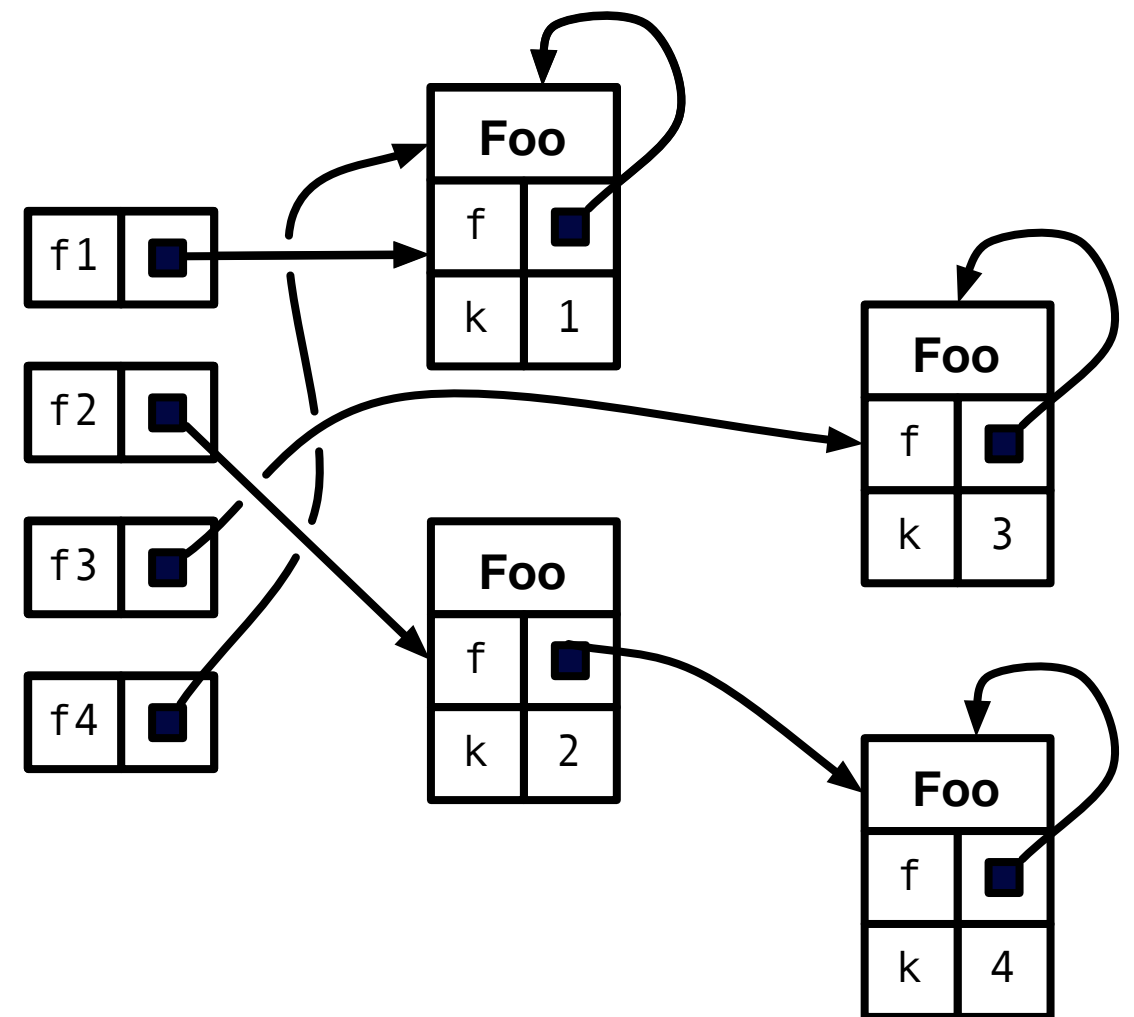
    public void sF(Foo f) {
        this.f = f;
    }

    public static void main(String args) {
        Foo f1, f2, f3, f4;
        f1 = new Foo(1);
        f2 = new Foo(2);
        f3 = new Foo(3);
        f4 = new Foo(4);

        f2.sF(f4);

        f4 = f1;
        // BANG
    }
}

```

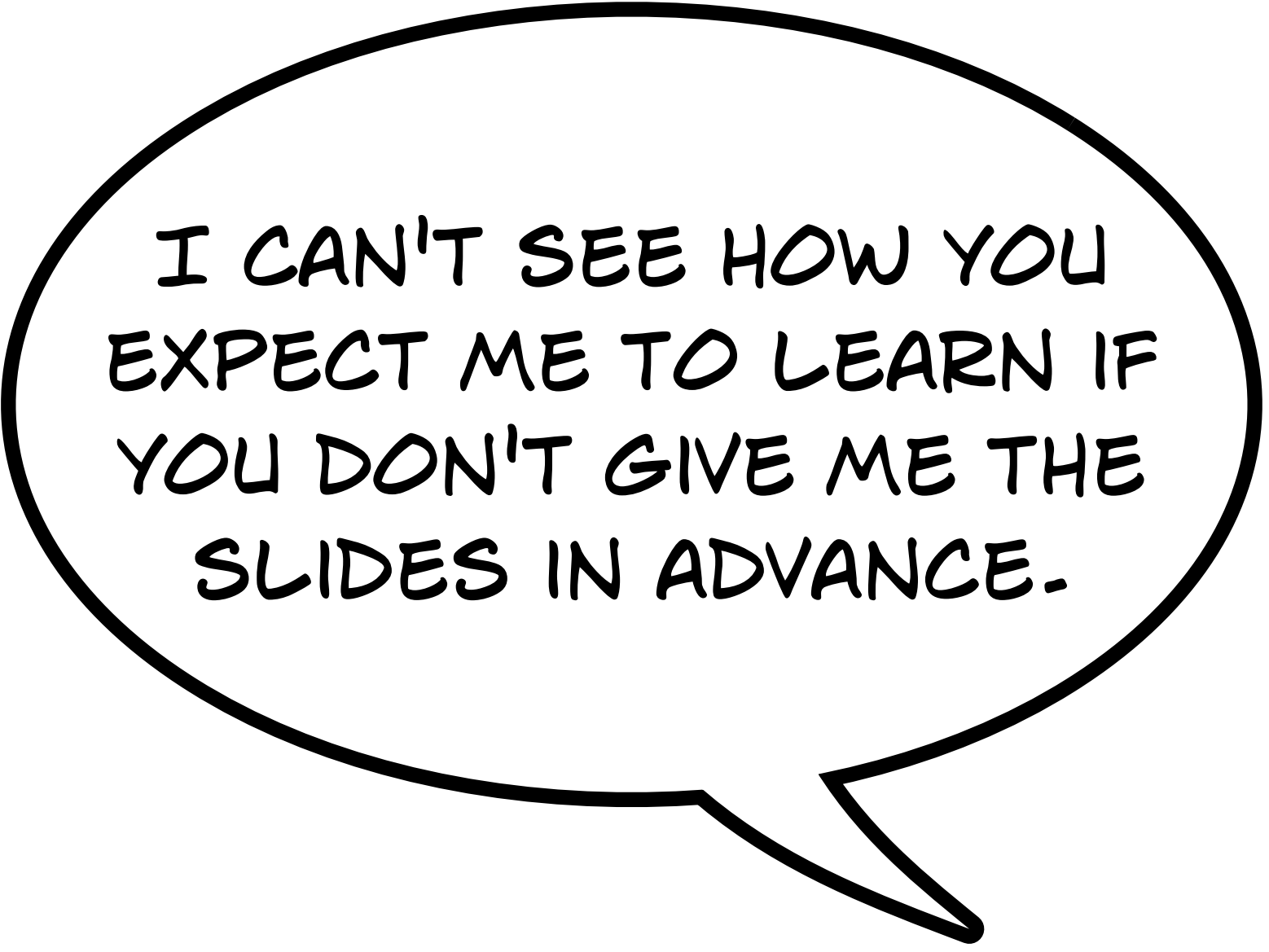


What else are slides good for?

Forecast

- When and how to use slides
- Why and how to make handouts
- How to make your life easier
- Teaching the Facebook generation

One problem with slides



I CAN'T SEE HOW YOU
EXPECT ME TO LEARN IF
YOU DON'T GIVE ME THE
SLIDES IN ADVANCE.

(Yes, this actually happened.)

The dirty little secret...

...is that I don't want to distribute my slides at all!

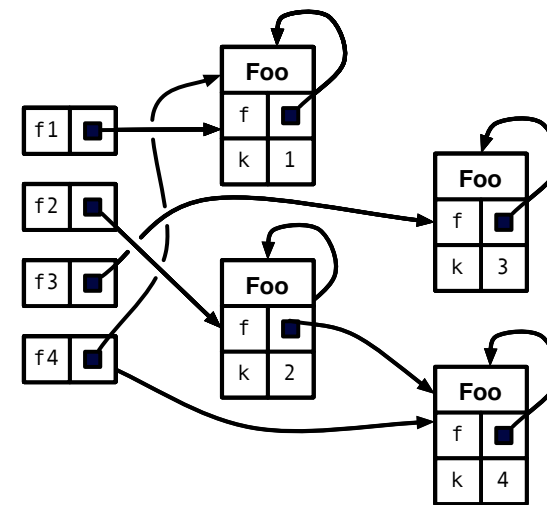
The dirty little secret...

...is that I don't want to distribute my slides at all!

The dirty little secret...

...is that I don't want to distribute my slides at all!

```
public class Foo {  
    private Foo f;  
    private int k;  
  
    public Foo(int k) {  
        this.k = k;  
        this.f = this;  
    }  
  
    public void sF(Foo f) {  
        this.f = f;  
    }  
  
    public static void main(String args) {  
        Foo f1, f2, f3, f4;  
        f1 = new Foo(1);  
        f2 = new Foo(2);  
        f3 = new Foo(3);  
        f4 = new Foo(4);  
  
        f2.sF(f4);  
  
        f4 = f1;  
        // BANG  
    }  
}
```



The dirty little secret...

...is that I don't want to distribute my slides at all!

The quandary

Handouts are the solution

Handouts are the solution

Engineering and Work Practice Controls (con't)

- The employer must:
 - Train employees to use new devices and/or procedures
 - Document in ECP



Handouts are the solution

Engineering and Work Practice Controls (con't)

- The employer must:
 - Train employees to use new devices and/or procedures
 - Document in ECP

Why integer overflow "wraps around"

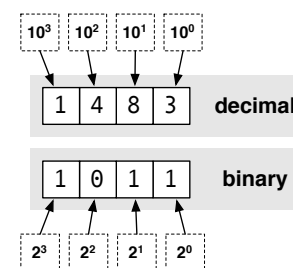
handout for CS 302 by Will Benton (willb@cs)

The whole-number types in Java (e.g. `int`, `long`, etc.) have limited ranges. Of course, any type representable in a finite computer has a limited range, but you're likely to actually run into the limits of the Java primitive types before you have to deal with a number that you can't represent on a computer.

Manipulating a variable so that its value would exceed the range of its type results in *integer overflow*. When this happens, the value of the variable "wraps around" to the opposite end of the range, just as a classic video game character might wrap around to the other side of the screen upon crossing an edge.

This handout will explain why this happens. To do so, we'll first need to consider how computers represent numbers. (If you find this interesting, you'll enjoy a computer organization class!)

Computers represent numbers in *binary*, or base-2. Humans typically deal with numbers in *decimal*, or base-10. Both kinds of numbers have "places," in which a digit denotes some quantity of a power of the base. Let's examine two different four-digit numbers to see the difference:



The decimal number has the value 1483: $1 * 10^3 + 4 * 10^2 + 8 * 10^1 + 3 * 10^0$. The binary number has the value 11: $1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$.

Note that with n bits, you can represent a number up to $b^n - 1$, where b is the base.

WHAT DOES THIS BINARY STUFF HAVE TO DO WITH OVERFLOW?

GLAD YOU ASKED!

CONSIDER HOW YOU'D ADD TWO NUMBERS BY HAND: YOU'D LINE UP EACH PAIR OF DIGITS, AND THEN ADD A PAIR AT A TIME, CARRYING WHEN NECESSARY.

IN THIS EXAMPLE, WE NEEDED AN EXTRA PLACE! REMEMBER, THOUGH, THAT IN AN N-BIT INTEGER, YOU ONLY HAVE N PLACES!

9999 + 1 = 10000

(*ADDITION BASICALLY WORKS THE SAME WAY IN BINARY.)

9999 + 1 = 0000

AS YOU CAN SEE, HAVING A LIMITED RANGE FOR NUMBERS MEANS YOU'RE GOING TO RISK WRAPPING AROUND TO THE OTHER END OF THE RANGE.

OK, I SEE THAT MUCH. BUT WHY DO JAVA INTEGERS WRAP AROUND AND BECOME NEGATIVE? SHOULDN'T THEY BECOME 0 INSTEAD?

WELL, RECALL THAT THE RANGE OF INTEGER TYPES IN JAVA INCLUDES NEGATIVE NUMBERS.

WELL, LET'S TALK ABOUT HOW JAVA REPRESENTS NEGATIVE NUMBERS IN BINARY; THAT SHOULD CLEAR THINGS UP!

CAN YOU GIVE ME AN EXAMPLE?

SURE!

7₁₀ == 0111₂
-7₁₀ == 1001₂
-8₁₀ == 1000₂

(*NOTE THAT THIS ALLOWS AN N-BIT NUMBER TO ASSUME A RANGE OF VALUES FROM -2^(N-1) TO +2^(N-1)-1.)

SO IF WE'RE USING 4-BIT NUMBERS, WHAT HAPPENS WHEN WE ADD 1 TO 7?

WE REPRESENT NEGATIVE NUMBERS IN DECIMAL BY PREFIXING THEM WITH A - SIGN. HOWEVER, THERE'S NO PLACE FOR SUCH A SIGN IN A BINARY NUMBER!

OVERFLOW!

Copyright © 2007 Will C. Benton

why-integer-overflow-cl-graffiti: Created on Fri Feb 18 2007; modified on Sun Feb 18 2007; page 1 of 1

Handouts are the solution

Handouts are the solution

Handouts are the solution

Handouts are the solution



25

Why integer overflow "wraps around"
handout for CS 302 by Will Benton (willb@cs)

This whole-number type in Java (e.g., `int`, `long`, etc.) has a limited range. Of course, any type representable in a finite computer has a limited range, but you're likely to actually run into the limits of the Java primitive types before you have to deal with a number that you can't represent on a computer.

Manipulating a variable so that its value would exceed the range of its type results in *integer overflow*. When this happens, the value of the variable "wraps around" to the opposite end of the range, just as a classic video game character might wrap around to the other side of the screen upon crossing an edge.

This handout will explain why this happens. To do so, we'll first need to consider how computers represent numbers. (If you find this interesting, you'll enjoy a computer organization class!)

Computers represent numbers in *binary*, or base-2. Humans typically deal with numbers in *decimal*, or base-10. Both kinds of numbers have "places," in which a digit denotes some quantity of a power of the base. Let's examine two different four-digit numbers to see the difference:

10 ³	10 ²	10 ¹	10 ⁰
1	4	8	3
decimal			
1	0	1	1
2 ³	2 ²	2 ¹	2 ⁰
binary			

The decimal number has the value 1483: $1 \cdot 10^3 + 4 \cdot 10^2 + 8 \cdot 10^1 + 3 \cdot 10^0$. The binary number has the value 11: $1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$.

Note that with n bits, you can represent a number up to $2^n - 1$, where n is the base.

25

Why integer overflow "wraps around"

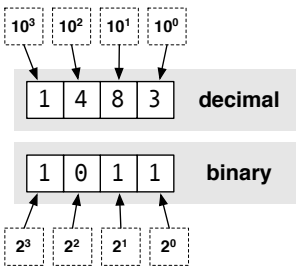
handout for CS 302 by Will Benton (willb@cs)

The whole-number types in Java (e.g., `int`, `long`, etc.) have limited ranges. Of course, any type representable in a finite computer has a limited range, but you're likely to actually run into the limits of the Java primitive types before you have to deal with a number that you can't represent on a computer.

Manipulating a variable so that its value would exceed the range of its type results in *integer overflow*. When this happens, the value of the variable "wraps around" to the opposite end of the range, just as a classic video game character might wrap around to the other side of the screen upon crossing an edge.

This handout will explain why this happens. To do so, we'll first need to consider how computers represent numbers. (If you find this interesting, you'll enjoy a computer organization class!)

Computers represent numbers in *binary*, or base-2. Humans typically deal with numbers in *decimal*, or base-10. Both kinds of numbers have "places," in which a digit denotes some quantity of a power of the base. Let's examine two different four-digit numbers to see the difference:



The decimal number has the value 1483: $1 \cdot 10^3 + 4 \cdot 10^2 + 8 \cdot 10^1 + 3 \cdot 10^0$. The binary number has the value 11: $1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$.

Note that with n bits, you can represent a number up to $2^n - 1$, where n is the base.

Copyright © 2007 Will C. Benton

What does this binary stuff have to do with overflow?

GLAD YOU ASKED!

CONSIDER HOW YOU'D ADD TWO NUMBERS BY HAND: YOU'D LINE UP EACH PAIR OF DIGITS, AND THEN ADD A PAIR AT A TIME, CARRYING WHEN NECESSARY.

9999
+ 1

10000

(ADDITION BASICALLY WORKS THE SAME WAY IN BINARY.)

9999
+ 1

0000

IN THIS EXAMPLE, WE NEEDED AN EXTRA "PLACE". REMEMBER, THOUGH, THAT IN AN N -BIT INTEGER, YOU ONLY HAVE N PLACES!

AS YOU CAN SEE, HAVING A LIMITED RANGE FOR NUMBERS MEANS YOU'RE GOING TO RISK WRAPPING AROUND TO THE OTHER END OF THE RANGE.

OK, I SEE THAT MUCH. BUT WHY DO JAVA INTEGERS WRAP AROUND AND BECOME NEGATIVE? SHOULDN'T THEY BECOME 0 INSTEAD?

WELL, RECALL THAT THE RANGE OF INTEGER TYPES IN JAVA INCLUDES NEGATIVE NUMBERS.

YEAH, SURE, BUT I STILL DON'T GET IT.

WELL, LET'S TALK ABOUT HOW JAVA REPRESENTS NEGATIVE NUMBERS IN BINARY; THAT SHOULD CLEAR THINGS UP!

WE REPRESENT NEGATIVE NUMBERS IN DECIMAL BY PREFIXING THEM WITH A - SIGN. HOWEVER, THERE'S NO PLACE FOR SUCH A SIGN IN A BINARY NUMBER!

THE WAY THAT JAVA REPRESENTS NEGATIVE NUMBERS IS CALLED TWO'S COMPLEMENT.

IN TWO'S COMPLEMENT, YOU NEGATE A NUMBER BY COMPLEMENTING EACH DIGIT AND ADDING ONE...

CAN YOU GIVE ME AN EXAMPLE?

SURE!

$7_{10} == 0111_2$
 $-7_{10} == 1001_2$
 $-8_{10} == 1000_2$

(NOTE THAT THIS ALLOWS AN N -BIT NUMBER TO ASSUME A RANGE OF VALUES FROM -2^{N-1} TO $2^{N-1}-1$.)

SO IF WE'RE USING 4-BIT NUMBERS, WHAT HAPPENS WHEN WE ADD 1 TO 7?

0111
+ 0001

1000 OVERFLOW!

(LIKE A VIDEO GAME)

why-integer-overflow-cl-graffite: Created on Fri Feb 16 2007; modified on Sun Feb 18 2007; page 1 of 1

Short but descriptive title

your contact information



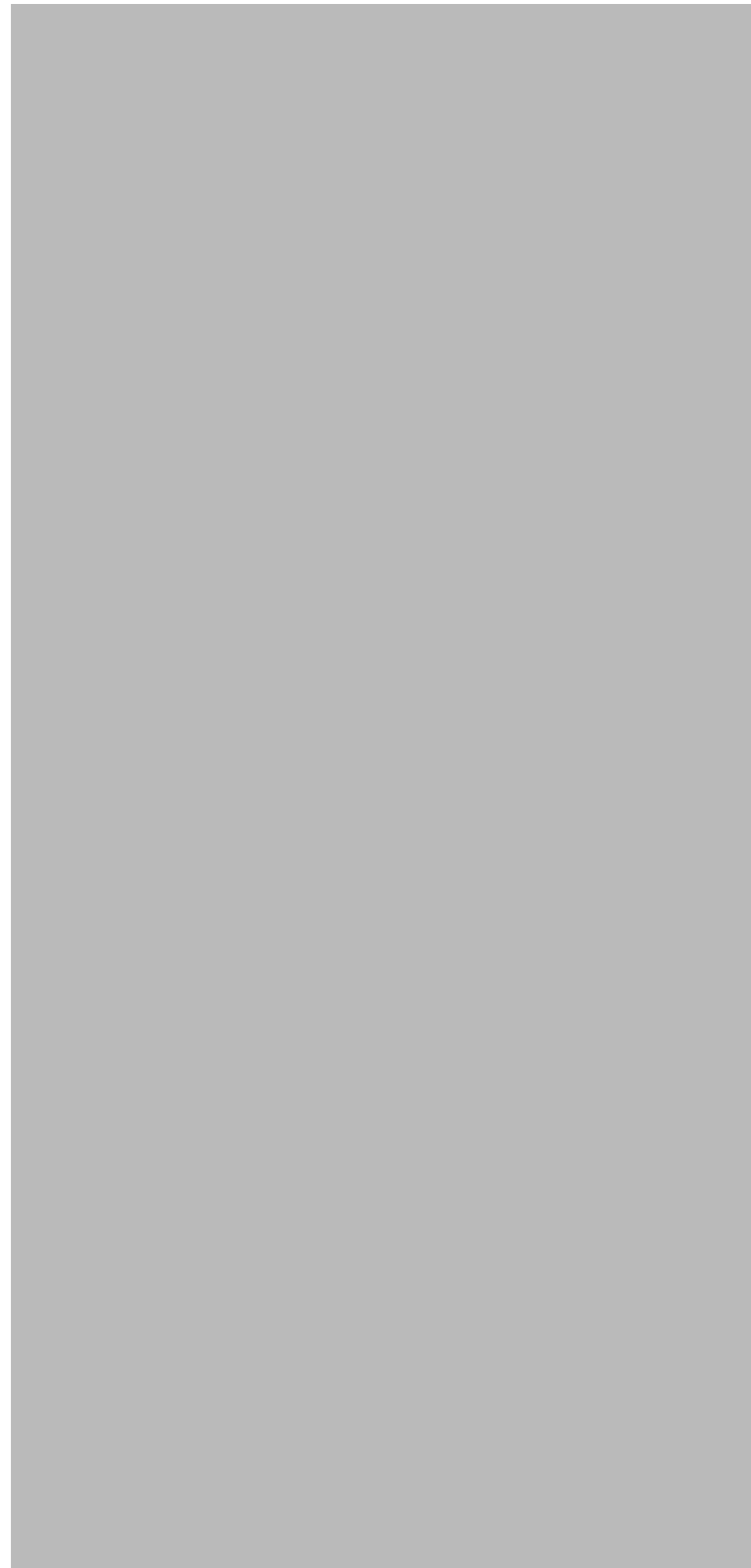
copyright notice



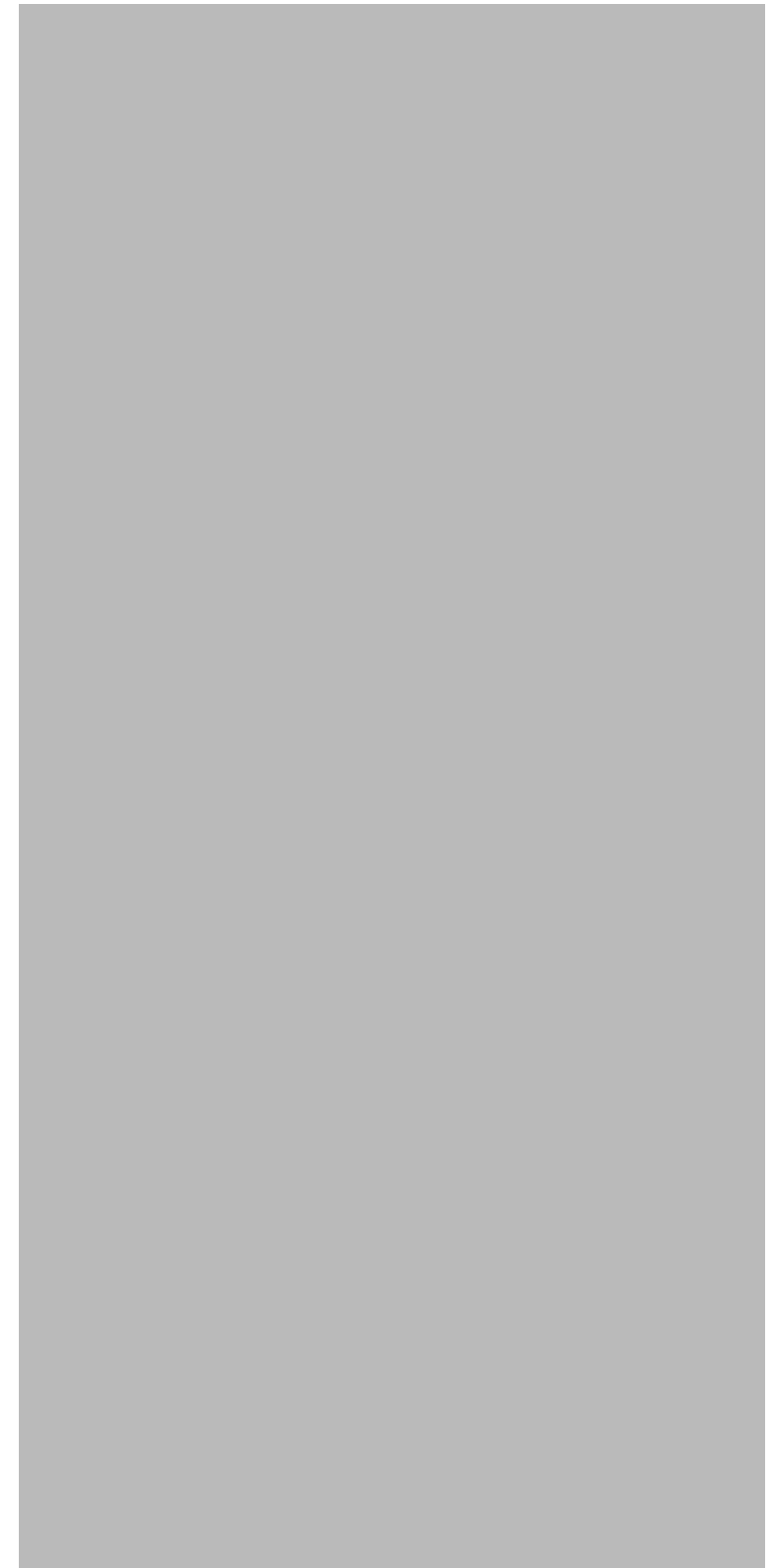
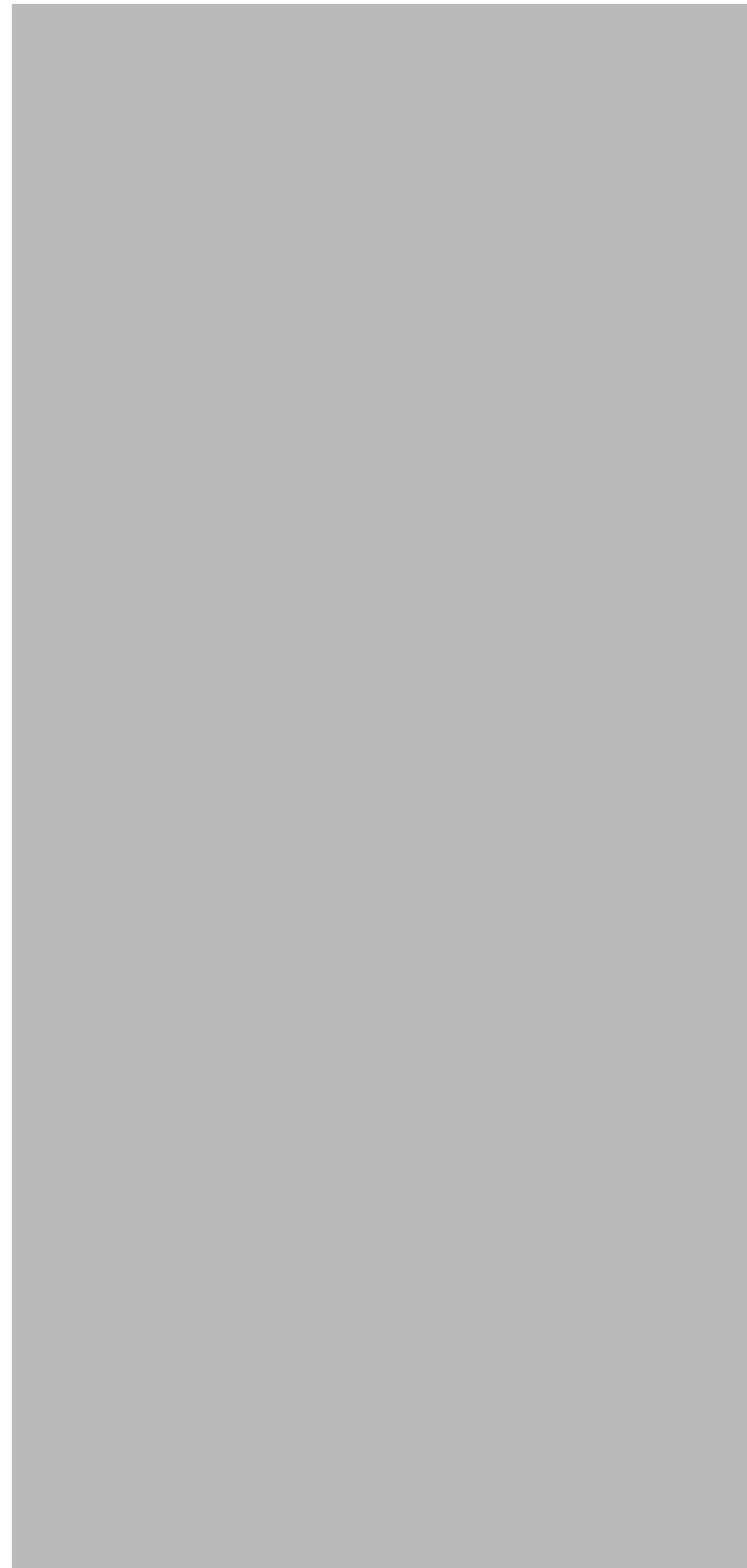
identifying information for this file

Short but descriptive title

your contact information



copyright notice



identifying information for this file

Short but descriptive title

your contact information



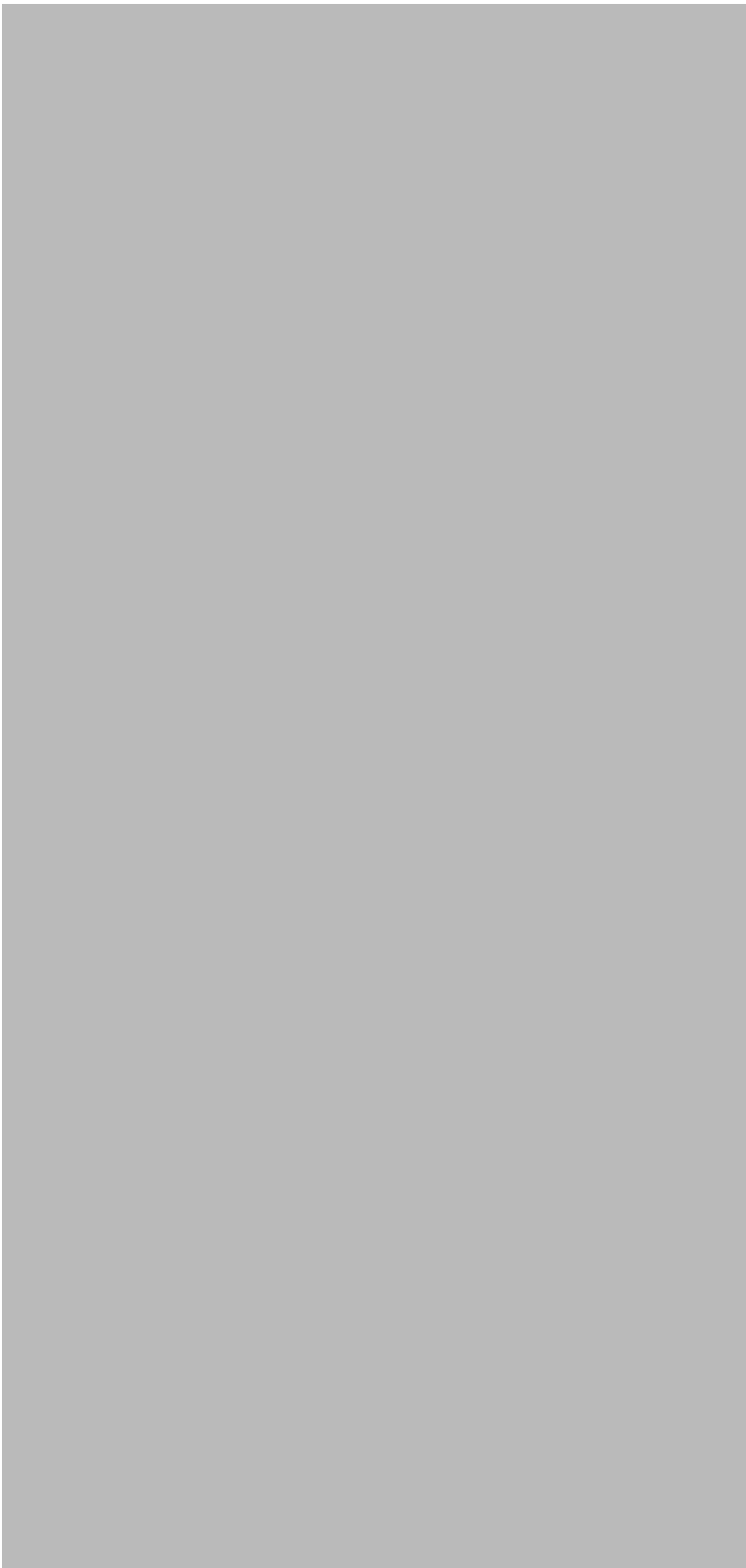
copyright notice



identifying information for this file

Short but descriptive title

your contact information



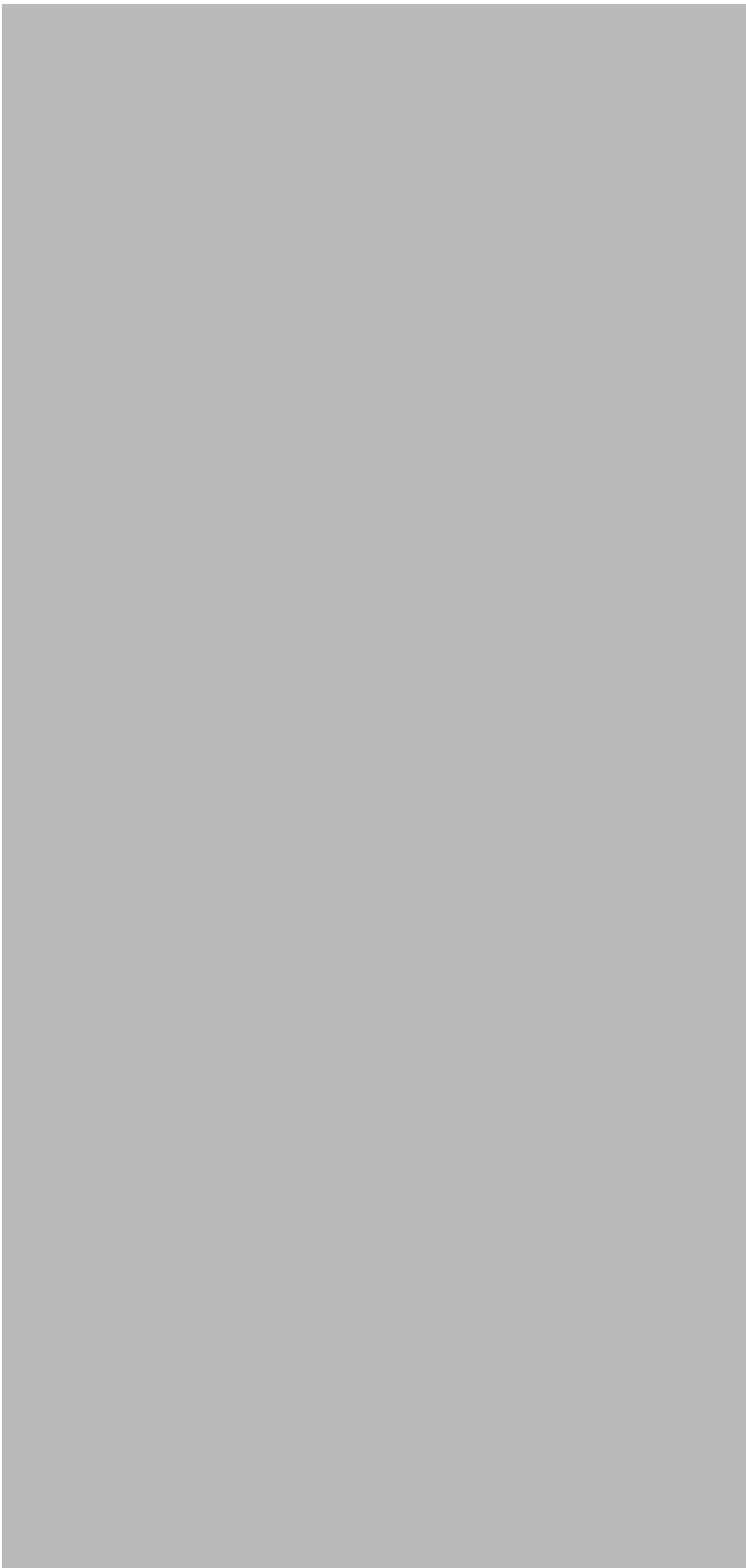
copyright notice



identifying information for this file

Short but descriptive title

your contact information



copyright notice



identifying information for this file

Short but descriptive title

your contact information



See *The anatomy of a simple handout* for more

copyright notice

identifying information for this file

“I’m already overworked.”



OK, THAT HANDOUT IS
FINE. NOW CAN I HAVE
THE POWERPOINTS?

Forecast

- When and how to use slides
- Why and how to make handouts
- How to make your life easier
- Teaching the Facebook generation

Learn the software

Build your own templates

Use “variables”

Your favorite tip here....

Forecast

- When and how to use slides
- Why and how to make handouts
- How to make your life easier
- Teaching the Facebook generation

Technology and experience