COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

Hello!

James (Jim) Young
young@cs.umanitoba.ca
jimyoung.ca

office hours T / Th: 17:00 – 18:00
EITC-E2-582
(or by appointment, arrange by email)

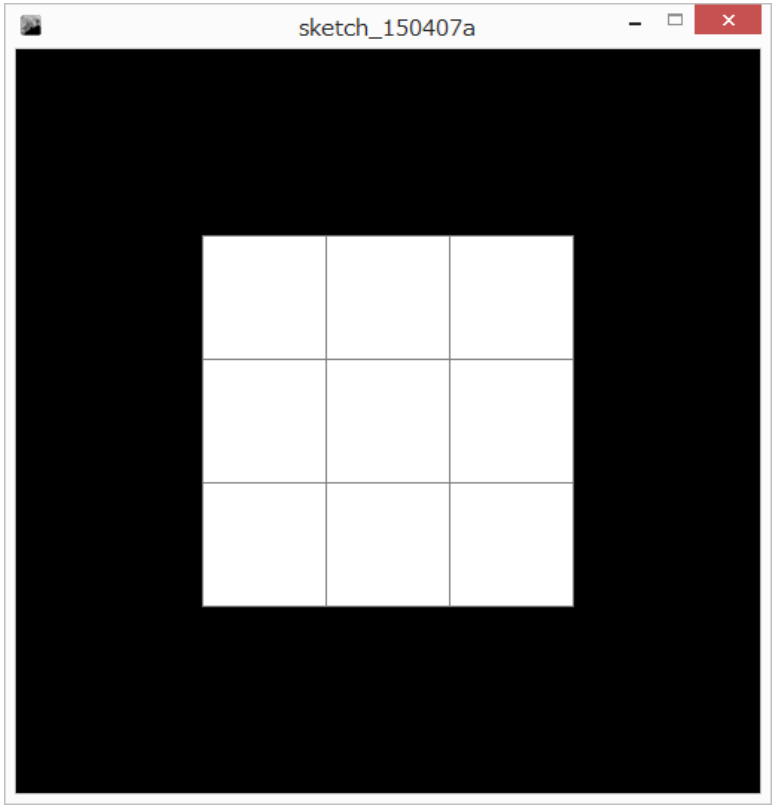
AS2

- Typo - reverse y not x when ball hits paddle
- Ball bounce off edge of paddle – easy bugs

Example: basic tic-tac-toe board

Setup variables

- board grid
- board size
- tile size
- boardCenterX,Y
- Board Left / Top



Setup the for loops

Iterate over tiles i (width), j (height)

Calculate left and top of each tile

Place a rectangle at the tile location

Is the mouse inside any of the tiles?

Update the for loop – while drawing, check to see if the mouse is inside

Add helper variables: right, bottom

Basic logic:

if mouse is to the right of left wall

to the left of the right wall

below the top wall

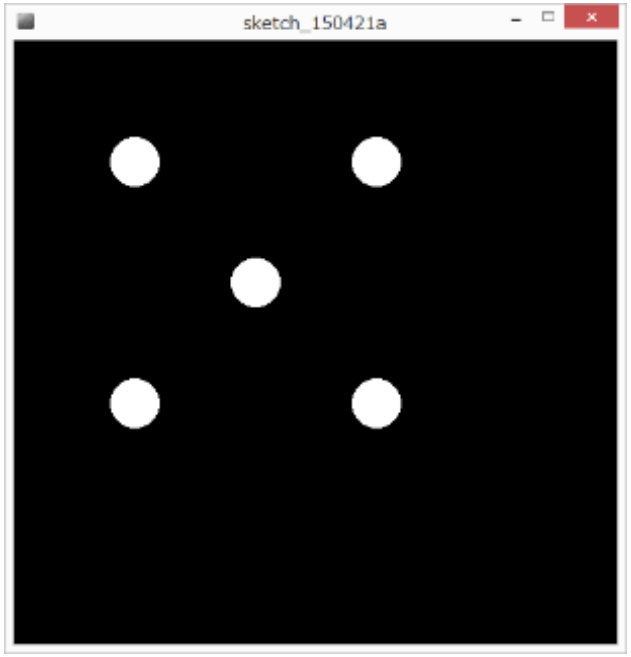
above the bottom wall

Change the color

exercise

Try drawing X and O instead of changing the color

For loop and boolean exercises



Draw a dice face

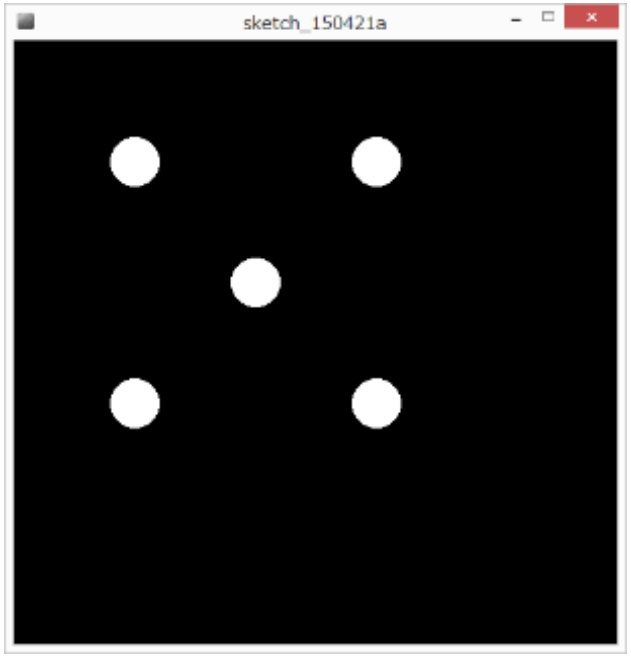
First, setup globals

- diceGrid
- diceSize
- diceSpacing
- left
- top
- dotSize

For loop for dice

Setup the nested for loop iterating over i,j and draw a dot grid.

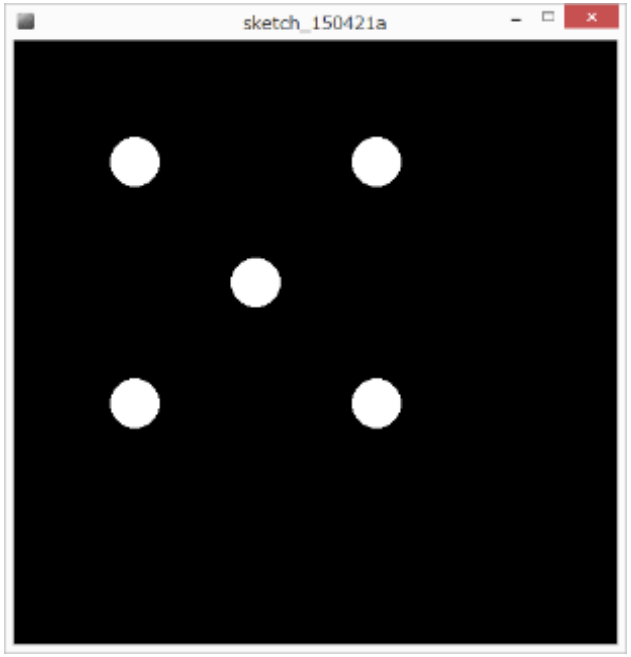
Use boolean logic to do one diagonal – a three!



Two diagonals for a 5

Other diagonal?

i (x)	j (y)
0	2
1	1
2	0



Combine with an OR

exercise

Do the other common dice faces

Coding style and standards

Coding style is VERY important!

1. commenting!
2. indentation!
3. Use meaningful variable names

   int a; // bad. Too short. Not meaningful.

   int a2; // even worse!

3. Use named constants! (in a second)
4. More…

Variable names..

Descriptive
Self-commenting
e.g.,
float t; // tax rate
float taxRate;

Standards – be aware of them

```
int _data;
boolean isHit;
```

“readable” code

what does this mean?

```
int resultA = 100*5*26;
int resultB = 52*5*26;
int resultC = 88*5*26;
```

in this case, a summer cottage industry calculating season costs:

the first number is the cost of a service per day, the second is the number of days a week open, and the third number (26) is how many weeks the business is open a year

what are two problems with my above example?

1) hard to read
2) what if the season or week length changes?
   I need to make a bunch of changes

named constants!!

constant – a value or piece of information which we guarantee will not change while the program is running. e.g., length of a business season, or sales tax, etc.

```
int resultA = 100*5*26
int resultB = 52*5*26;
int resultC = 88*5*26;
```

```
int resultA = 100*DAYS_PER_WEEK*WEEKS_PER_YEAR;
int resultB = 52*DAYS_PER_WEEK*WEEKS_PER_YEAR;
int resultC = 88*DAYS_PER_WEEK*WEEKS_PER_YEAR;
```

```
int hydroCost = hydroDaily *DAYS_PER_WEEK*WEEKS_PER_YEAR;
int resultB = 52*5*26;
```

naming conventions

note: naming conventions are not processing rules, but accepted standards that help improve readability:

note: named constants are usually ALL_CAPS_WITH_UNDERSCORES_FOR_SPACES

regular changing variables are usually smallFirstWordAndCapitalizeEveryOtherWord.

```
int hydroCost = hydroDaily *DAYS_PER_WEEK*WEEKS_PER_YEAR;
```

named constants in Processing are done with the “final” keyword.

final type variableName;

final variables can only be set ONCE and never change:

final int WEEKS_PER_YEAR = 26;

or

final int WEEKS_PER_YEAR;
WEEKS_PER_YEAR = 26;
WEEKS_PER_YEAR = 0; // ← illegal because already set

for your assignments…

- reasonable variable names
- consistent and good indentation
- reasonable comments (err on the side of too many)

will be stressed more as we go through the course

Data types and memory

bits and bytes and nibbles… (don't memorize)

A computer stores everything as switches (**bits**)
represent 0 (off) and 1 (on).

A group of 8 **bits** (switches) makes a **byte**
00110110 ← one byte of data

A group of 4 **bits** (half a bite) make a **nibble**
(I kid you not!) 1101 is a **nibble** of data

1024 **bytes** (2^{10}) make a **kilobyte**

1024 **kilobytes** (2^{20} bytes) make a **megabyte**

1024 **megabytes** (2^{30} bytes) make a **gigabyte**

1024 **gigabytes** (2^{40} bytes) make a **terabyte**

A **terabyte** has 1,099,511,627,776 bytes or
8,796,093,022,208 switches
using a 7cm standard light switch… 615 million KM
4 times the distance to the sun!!!!!!!!!!!!

(aside: new standard units are moving to even powers of ten where 1 terabte = 1,000,000,000,000 bytes)

Counting with bits!!!!! (not covered in class, not testable)

0 -> 0

1 -> 1

How to represent “2”?

we need another bit. Put it in front

Start over..

00 // right column is 2^0 place

01

10 -> 2 // left column is 2^1 place

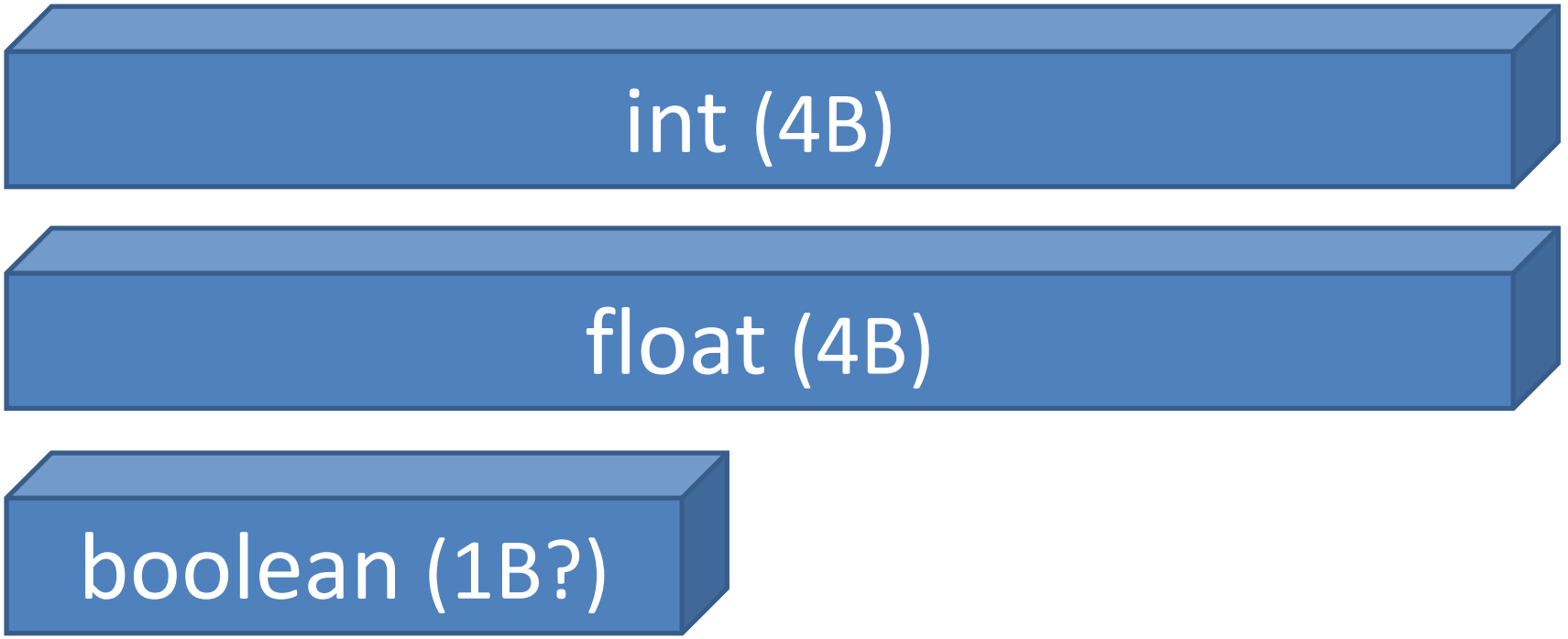
11 -> 3

100 -> 4 // left column is 2^2 place

What is 6?

110

data types so far

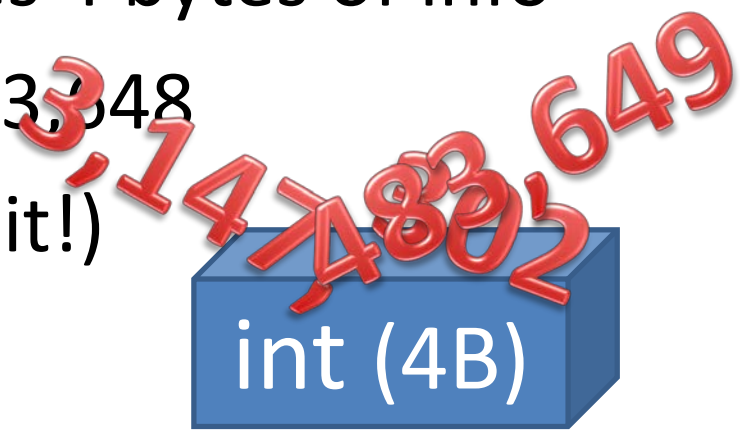


integer overflows

"int" type in Processing stores 4 bytes of info

smallest number is -2,147,483,648

largest is 2,147,483,647 (try it!)



what happens when you go past these numbers accidentally?

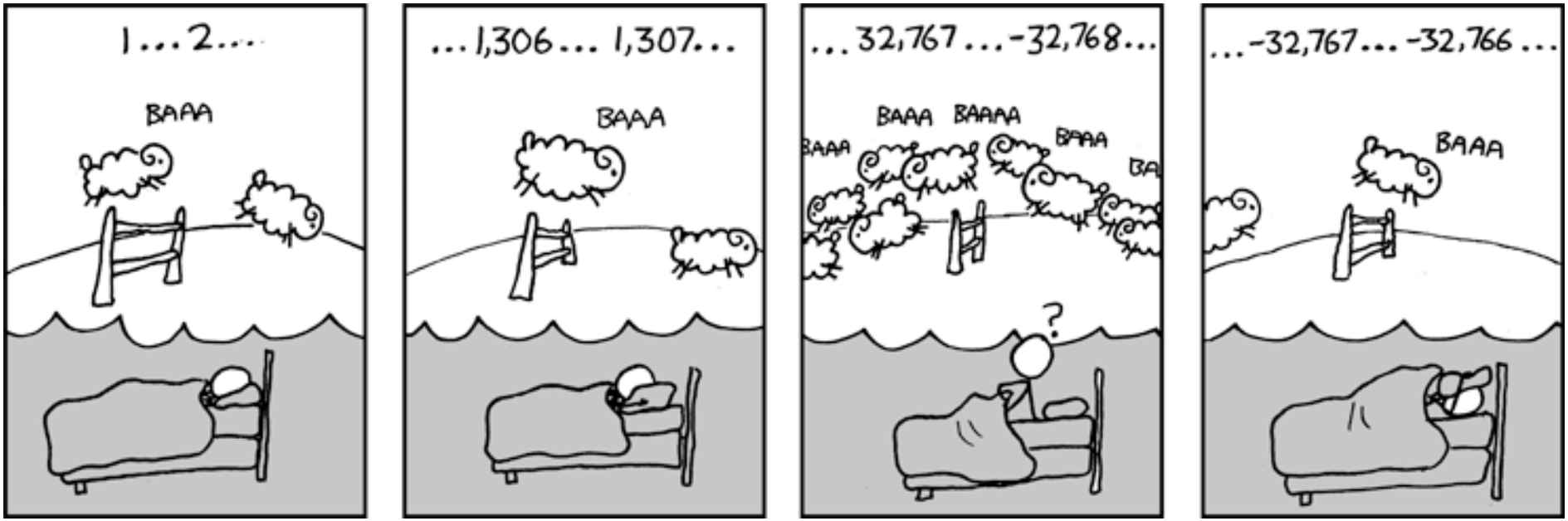
variable overflow

note: when you go over the specified maximum value of an integer variable, the value *wraps around* to the smallest value.

when you go below the specified minimum, it wraps in the other direction.

quick comic

a data type that we do not use in this course, called a **short** (a short integer), is 2 bytes (int is 4) and can hold the range -32,768…32,767



XKCD.com

exercise: infinite loop?

```
for (int i = 1; i > 0; i++) // infinite loop?
{
   ; // do nothing
}
background(255);
line(0,0,mouseX,mouseY);
```

is this an infinite loop?

lets test

!(infinite loop):

```
for (int i = 1; i > 0; i++) // infinite loop?
{
   ; // do nothing
}
```

this is not an infinite loop because i cannot get infinitely large. It is limited by the memory of the int data type. Once it hits the largest limit, adding one will make it “roll over” to the smallest value, making it less than 0.

two ways to avoid overflow:

1) use a different data type

2) be clever with your calculations to avoid large numbers

Primitive data types: integers

type	size	minimum	maximum

All used like int

Integer math

Primitive data types: floating point

float – 4 bytes

double – 8 bytes

More memory is more precision, not more range

e.g.,

float - 0.6666667

double - 0.6666666666666666

primitives

boolean – true, false

char – store one character (later!)