Computer Science Education in the Montessori Classroom

Gary Kacmarcik
Google – Seattle, WA

Sylvie Giral Kacmarcik
Spring Valley Montessori – Federal Way, WA

Preliminaries

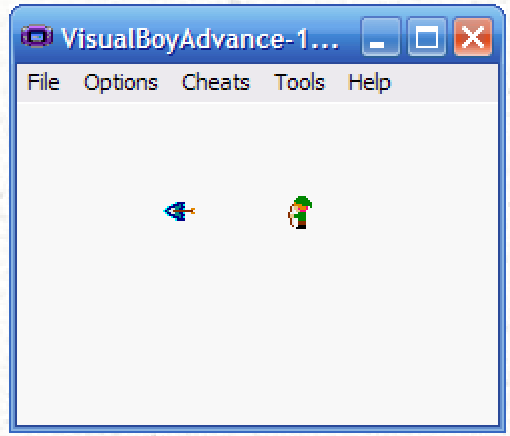
- Slides will be posted at:
  - cse4k12.blogspot.com
- Email contact:
  - garykac @ either “gmail” or “google” .com

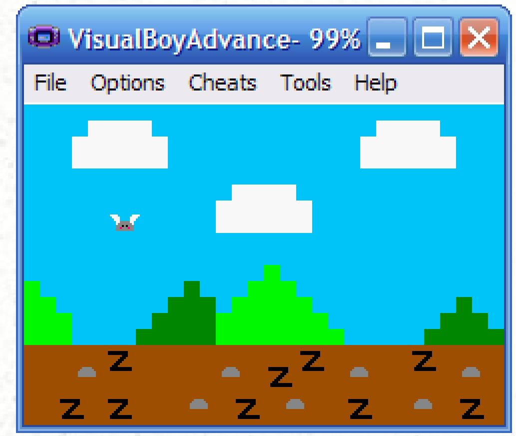
This info will be repeated at end

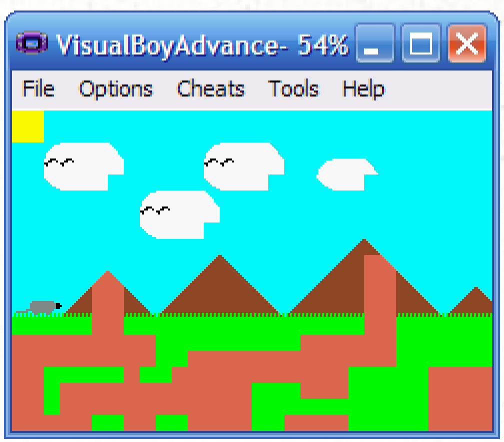
Background

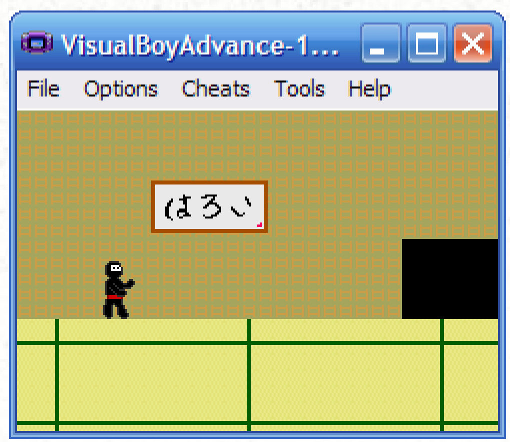
- Desire to teach computer programming
- Taught class:
  - 6th-8th graders
  - ~40 once-a-week 1-hour classes
  - Student project: create a GBA/NDS game
- First few months of class spent teaching basics
  - These could/should have been taught earlier

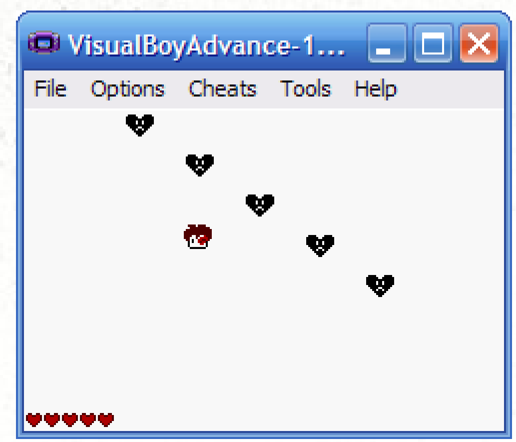
Student projects











Why teach computer science?

- Well, why do we:
  - Teach the Bohr model of the atom?
  - Teach photosynthesis?
  - … other sciences?
- Answer:
  - So students can understand the world around them

Understanding your surroundings

- How does:
  - A toilet work?
  - A phone work?
  - A car (internal combustion engine) work?
- We can explain how these work in general, accessible terms.
  - Concrete physical explanation

How do computers work?

- Many explanations are something like:
  - Mumbling something about 0's and 1's
  - Memory, hard drives, CPU, ALU
- These aren't wrong, but they aren't concrete
  - Students don't walk away with understanding

It's all magic

- “Any sufficiently advanced technology is indistinguishable from magic.”
  - Arthur C. Clark, 1973
- Kids are growing up in this magical world
- How much of the world will they understand in 20, 30 years?
  - Compare this with 100 years ago

But we teach computers, don't we?

- Current state of K-8 computer “science”:
  - Typing skills
  - How to use productivity applications
    - Word processing, spreadsheets, …
- These are necessary, but not sufficient
- We don't teach students how to use a calculator
  - We teach them math

But programming is hard/abstract

- True, so teach programming in 6th-8th
  - Requires abstract thinking
- But teach pre-programming before 6th grade
  - Focus on fundamental principles:
    - Binary
    - Logic
    - Transistors
  - No need for students to use computers for this

Our goals

- Convince you that teaching pre-programming:
  - Is important
  - Is possible
  - Is easy
    - Well, as easy as any other topic you teach

Pre-programming skills

- Binary / Hexadecimal
- Boolean Logic
- Transistors
- How Computers Add
  - Not a skill *per se*
  - But useful to tie everything together

Binary / Hexadecimal
(3rd-4th grade)

Number systems

- How many ways can you represent a number?
  - Tally marks: | || |||
  - Roman numerals: LXVII
  - Writing systems: 一二三四 ... ١٢٣٤...
- Activity:
  - Grab random number of counters and write the # as many different ways as possible
  - Why: To demonstrate that decimal is not the only way to count things

Zero

- The concept of zero is a crucial part of our number system
- Before zero, how do you distinguish between:
  - There were no items
  - Oops, I forgot to fill in this value
- Curriculum link:
  - Compare early societies that invented zero vs. those that didn't

Positional notation

- Decimal is a base-10 positional notation
  - There are 10 digits: 0123456789
  - Value of each digit depends on its position:
    - ..., 1000, 100, 10, 1
    - ..., 10x10x10, 10x10, 10, 1
- Compare 324 and 243
  - Both use same digits
  - Value of each digit is different

Why base-10?

- What about other bases?
  - Some cultures use/used base-5, -12 or -60
- Activity:
  - Where do we see remnants of other bases in modern society?
    - Clocks, Angles, Lat/Long, Eggs
  - Why: Point out that the choice of 10 was somewhat arbitrary

Octal (base-8)

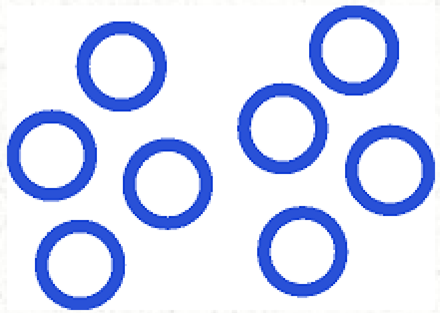
- 8 digits: 01234567
  - Positions: ..., 8x8x8, 8x8, 8, 1
- Why octal?
  - Binary can be confusing at first.
  - Octal is closer to decimal – less confusing
- Activity:
  - Get random number of counters. Group by 10 to get decimal. Using same counters, group by 8 to get octal. Repeat.

Counting in octal

- 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, ..., 77, 100, …
  - 10 comes after 7
  - 20 comes after 17
  - 100 comes after 77
- Compare with decimal
  - 8, 9 are never used

How many is 12?

- Draw 12 circles
  - Ambiguous – base-10 or base-8?
- How many circles?
  - 10 (in octal)
  - 8 (in decimal)



Ambiguity

- “12” means different things in octal/decimal
  - How do we know which one to use?
- Compare:
  - How do you pronounce “wind”?
  - Is “rose” a noun or a verb?
  - “I'll meet you at 8”. Is that AM or PM?
- Context resolves the ambiguity.

Binary (base-2)

- Same as decimal, octal, except:
  - 2 digits: 0, 1
  - Positions: ..., 2x2x2x2, 2x2x2, 2x2, 2, 1

Counting in binary

- 0, 1
  - Oh no, we ran out of digits already
- 10, 11
  - We ran out again
- 100, 101, 110, 111
- 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111
- This is why we start with octal

Binary activity

- Grab a random number of counters (20-30)
  - Group by 2's
    - Combine these into groups of 4's
    - Repeat into groups of 8's, 16's, 32's
  - Place group next to binary position card
  - Binary # is created by:
    - Writing a “1” where you have a group
    - Writing a “0” where you don't

Binary activity (example)

- Example, take 11 tokens: ooooooooooo
  - Group by 2: (oo)(oo)(oo)(oo)(oo)o
  - Group by 4: (oooo)(oooo)(oo)o
  - Group by 8: (oooooooo)(oo)o
- We have: 1 eight, 0 four, 1 two, 1 one
  - Binary number: 1011

Problems with binary

- Decimal is easier to work with
  - 265 versus 100001001
  - What is 11010101101 + 10110100101?
- Non-trivial conversion between base-10

Converting binary/decimal

- Remember, decimal 243 is:
  - 2 x 100 + 4 x 10 + 3 x 1
- Binary 1011001 is:
  - 1x64 + 0x32 + 1x16 + 1x8 + 0x4 + 0x2 + 1x1
  - = 89
  - Ugh! That was work...

Hexadecimal (base-16)

- 16 digits: 0123456789abcdef
  - a=10, b=11, c=12, d=13, e=14, f=15
  - Single digit representation for each value
  - Positions: ..., 16x16x16, 16x16, 16, 1
- Activity: (older students)
  - Where have you seen hexadecimal?
    - Specifying RGB colors: HTML, Photoshop, ...

Counting in hexadecimal

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
20	21	22	...												
...															
e0	e1	e2	e3	e4	e5	e6	e7	e8	e9	ea	eb	ec	ed	ee	ef
f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	fa	fb	fc	fd	fe	ff
100	101	102	...												

Why hexadecimal?

- Compare positions for binary and hexadecimal
- They line up at 1, 16, 256, …
- This makes conversion easy

binary	hexadecimal
1	1
2	
4	
8	
16	16
32	
64	
128	
256	256
512	
1024	
2048	

Converting binary/hexadecimal

- Binary number: 1110100100100101
- Group by 4 digits: (1110)(1001)(0010)(0101)
  - Starting from left side
- Convert each group independently:

binary	hex	binary	hex	binary	hex	binary	hex
0000	0	0100	4	1000	8	1100	c
0001	1	0101	5	1001	9	1101	d
0010	2	0110	6	1010	a	1110	e
0011	3	0111	7	1011	b	1111	f

- Thus, 1110100100100101 becomes E925

Hexadecimal activity

- Activity: Take a set of binary numbers
  - Convert them into ***either*** decimal or hex
  - Why? Get student to recognize hex is easier
- Extra: (older students)
  - “hexadecimal” is a mix of greek/latin roots
  - It should be “sexadecimal”
    - Compare sexagesimal for base-60
  - Can you guess why hex was chosen?

Boolean Logic
(4th-5th grade)

Boolean logic

- Given a set of true/false statements:
  - isRaining – true if it's raining outside
  - hasRaincoat – true if you have a raincoat
- Formal way of combining statements:
  - getsWet = isRaining AND NOT hasRaincoat
- Basis of logical “thought”
- Two values : true/false. Sounds like binary.
  - 0 = false; 1 = true

Boolean logic activity

- Create logical statements about objects/people in the classroom
  - isRed, isFlat, isBiggerThanMyHead, ...
  - isBoy, hasGreenShirt
- Apply the statements to other objects and state whether they are true or false

Boolean operations

- Four basic operations that can be applied to statements:
  - NOT
  - AND
  - OR
  - XOR

NOT

- Changes true to false and *vice versa*
- Given
  - isCat = true
- Then
  - NOT isCat = false
- In English, we say “is not a cat”
  - In logic, we say “not is-a-cat”

AND

- **a** AND **b** is true only if **a** and **b** are both true
- Given statements:
  - isCat
  - hasStripes
- Then
  - isCat AND hasStripes
    - is true for striped cats
    - is false for spotted cats or striped dogs

OR

- **a** OR **b** is true if either **a** or **b** (or both) are true
- Given statements:
  - isCat
  - hasStripes
- Then
  - isCat OR hasStripes
    - is true for striped cats, spotted cats, striped dogs
    - is false for spotted dogs

XOR

- **a** XOR **b** is true if either **a** or **b** (but not both) are true
- Given statements:
  - isCat
  - hasStripes
- Then
  - isCat XOR hasStripes
    - is true for spotted cats, striped dogs
    - is false for striped cats, spotted dogs

Boolean logic activity

- Activity #1:
  - Create more logical statements using NOT, OR, AND, XOR
- Activity #2:
  - Take a collection of objects:
    - Items in room, cards from "Guess Who?", …
  - Choose a few of the objects
  - Create a statement that will be true only for those objects and false for all others

Boolean logic example

- Cockatrice in the game Nethack
  - Petrifies you (turns you to stone) if you touch it
- Logic from game:

```
petrify = (handAttack AND NOT wearGloves)
    OR (kick AND NOT wearBoots
    OR (headbutt AND NOT wearHelmet)
    OR (hug AND NOT (wearGloves AND wearCloak))
    OR bite OR sting OR suckBrain OR swallow
```

  - Note that in the game you can polymorph into monsters (like mind-flayer) that have special attacks (like brain-suck)

Boolean logic activity

- Design a simple game that contains a special object.
  - Create logical expressions that define all the interactions with that object
  - pickUp, drop (on ground, in water), eat, hit, ...
- Trade with a student and see if they can come up with conditions you didn't think of
- Do the same thing for an object in a video game that you've played

Boolean logic – final notes

- Two final notes on boolean logic:
- Truth tables
  - Used to summaries logic statements
  - Useful when solving logic problems
- Logic gates
  - Graphical representation of logical operations

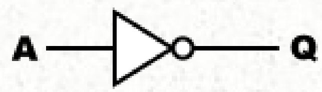
Truth tables

- Way of presenting logical statements
  - Enumerating all possible outcomes

isCat	**hasStripes**		IsCat **AND** hasStripes	IsCat **OR** hasStripes	IsCat **XOR** hasStripes
F	F		F	F	F
F	T		F	T	T
T	F		F	T	T
T	T		T	T	F

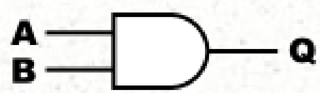
Logic gates

NOT



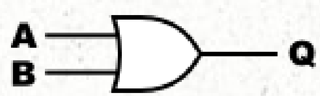
A	Q
0	1
1	0

AND



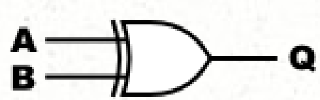
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

OR



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

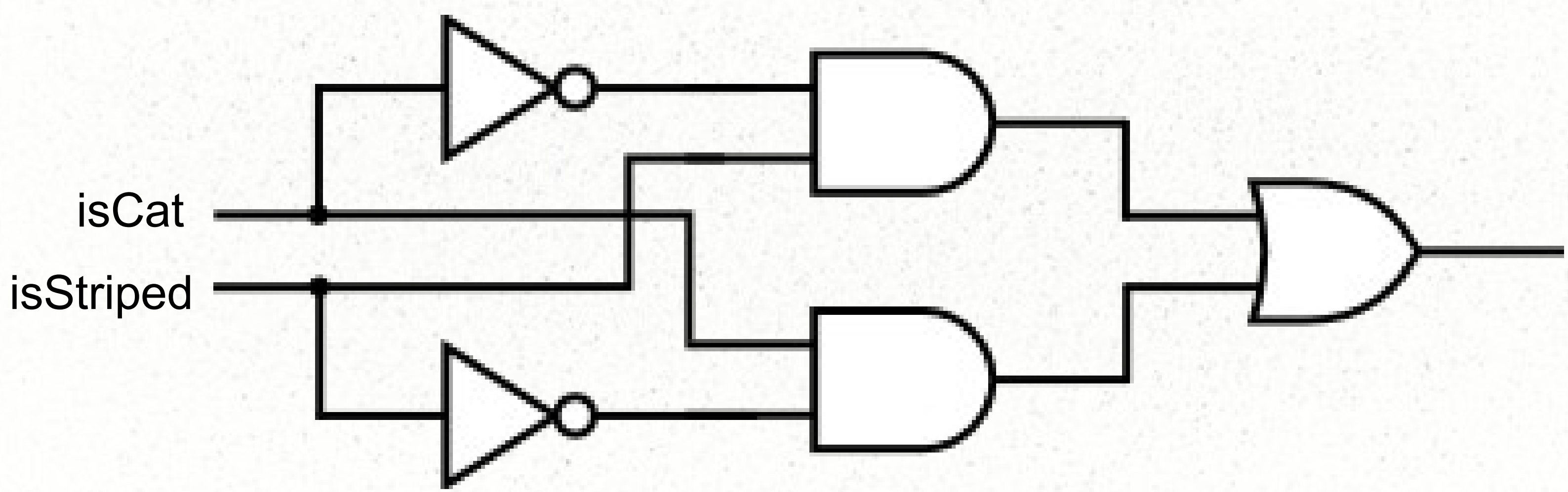
XOR



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Logic gate activity

- Draw one of your logical statements from before as a connected series of gates:



Transistors
(4th-5th grade)

Activity

- Good time to cover/review basic electricity:
  - Light bulbs, switches, batteries
  - Electrical current, electrons
  - Materials: conductors vs. insulators
- No need to cover resistors or anything more complex

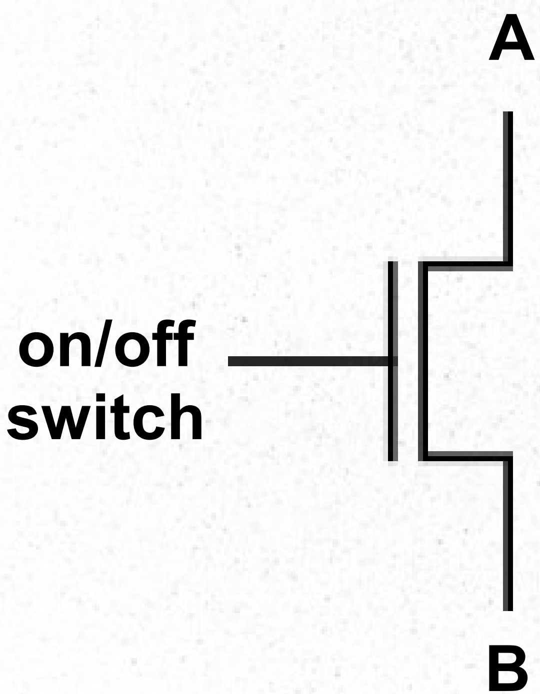
Binary values in electricity

- Electronic devices typically have 2 electrical states:
  - Power (red wire)
  - Ground (black wire)
  - For a battery: (-) Ground (+) Power
- Two states. Sounds like binary:
  - 0 = Ground; 1 = Power

Transistors

- What are transistors?
  - Electrical switches
- Similar to the light switches:
  - Two positions: on / off
    - On: electricity flows to lightbulb
    - Off: electricity does not flow to lightbulb
  - Difference:
    - Switch is controlled electrically
    - So a switch can control another switch

Transistor



If switch is ON
A and B are connected

If switch is OFF
A and B are not connected

Types of transistors

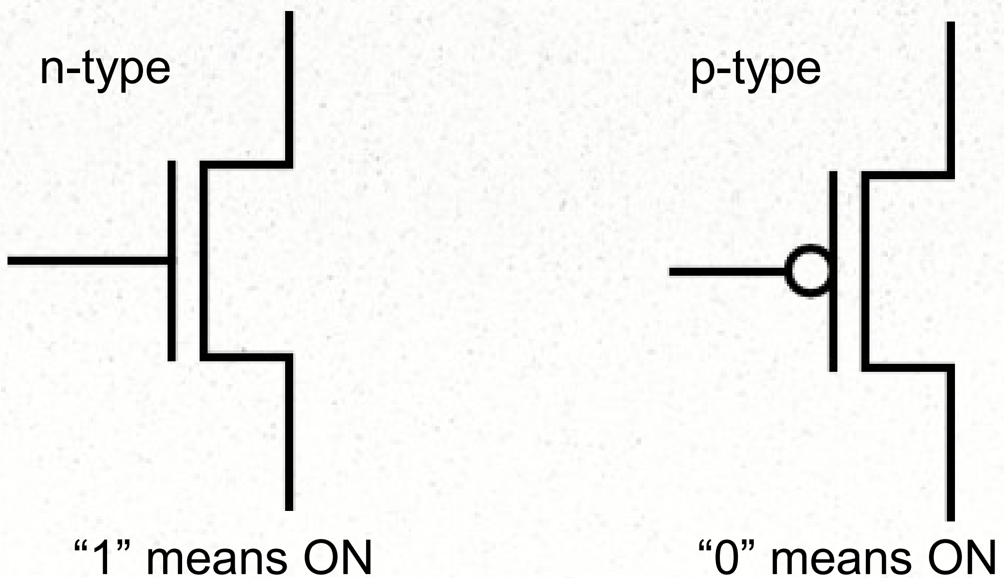
- Lots of different types:
  - FET : field effect transistor
  - BJT : bipolar junction transistor
- But we don't care:
  - Just pick one that is easy to explain:
  - MOSFETs : metal oxide semiconductor FET
  - Also one of the more common types

Semiconductors

- Semiconductors:
  - Not good conductors
  - Not good insulators
- What good are they?
  - With the right tricks, they can switch from one state to the other
  - This is how transistors work

CMOS

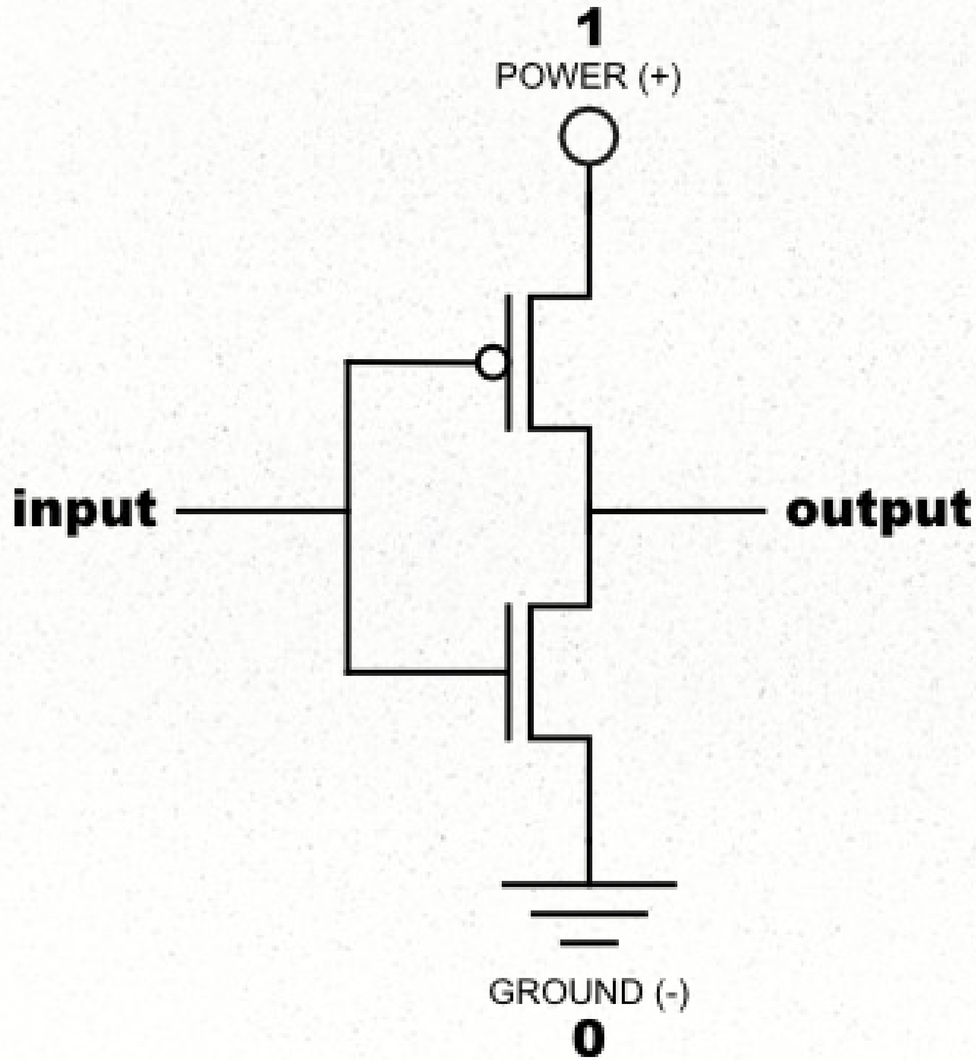
- CMOS (complementary MOS):
  - Uses nMOS and pMOS transistors
  - Arranged in a complementary fashion

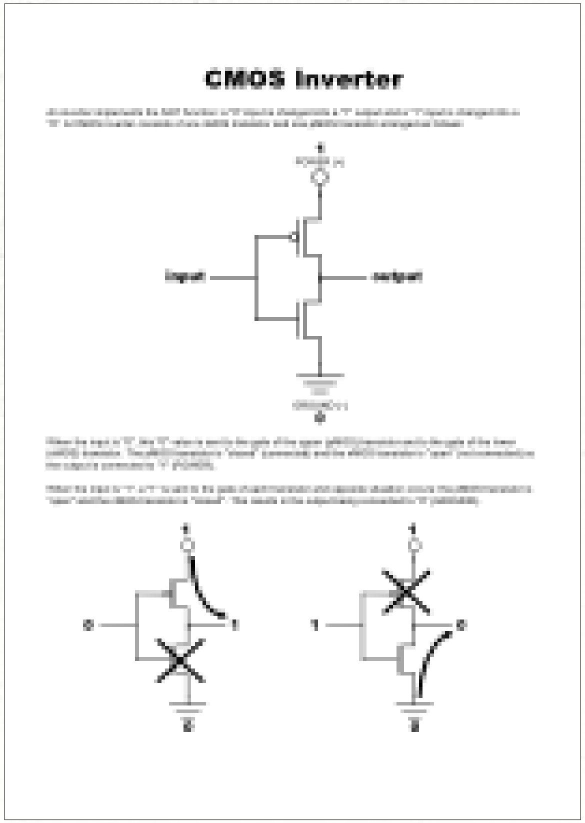


Building logic gates from transistors

- Invertor (boolean NOT) is the simplest
    - Requires 2 transistors: 1 nMOS & 1 pMOS
- Other gates can be built with 2 of each type:
    - NAND = NOT AND
    - NOR = NOT OR
- We can combine gates:
    - AND = NOT NAND
    - OR = NOT NOR

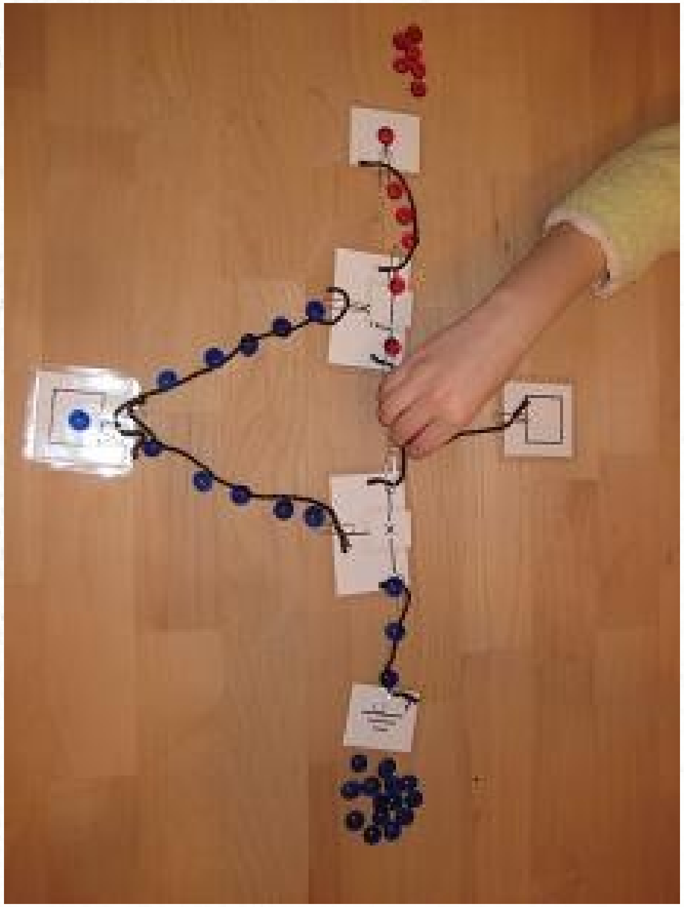
CMOS inverter





Transistor activity - inverter

- Use cards to build logic gates from transistors
- Red/black markers for power/ground
- Students trace flow of “0”s and “1”s



How Computers Add

(5th-6th grade)

How computers add

- Computers add the same way humans add
  - Except in binary

How do you teach addition?

B

A

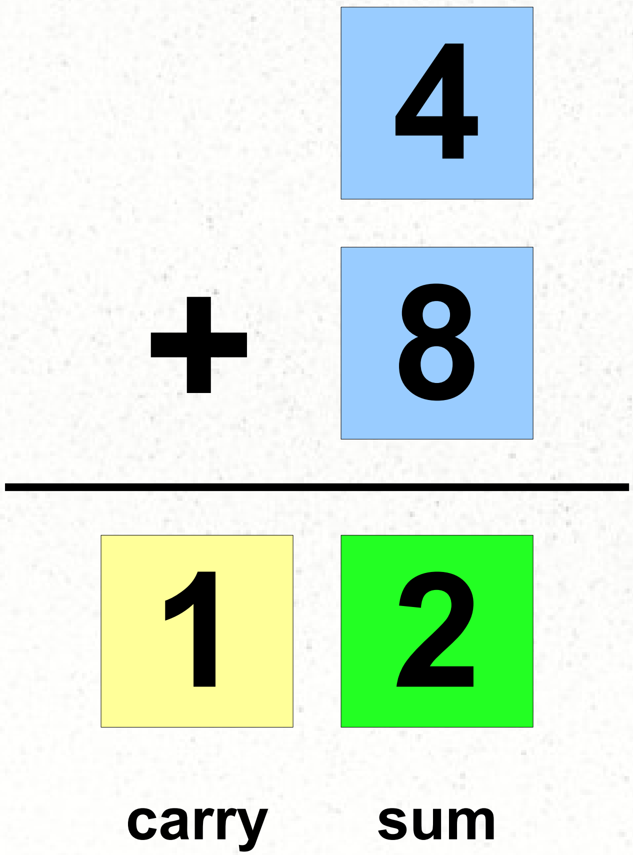
+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

Memorize this table!

Teaching addition (review)

- First teach adding 2 single-digit numbers
  - 1+1=2, 3+5=8, ...
  - Later, multi-digit answers:
    - 7+8 = 15
- Then expand to adding multi-digit numbers:
  - 18 + 23 = 41
  - With carry

Single digit addition



Add the 2 blue numbers

Produce the single-digit sum (green) and the carry (yellow)

Sum table & carry table

B

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

A

Sum table

Carry table

B

+	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1
6	0	0	0	0	1	1	1	1	1	1
7	0	0	0	1	1	1	1	1	1	1
8	0	0	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1

A

Binary addition

- Same general idea as with decimal
- Uses binary addition table
    - Just like the decimal addition table

Binary addition table

B

	+	0	1
A	0	0	1
	1	1	10

That's all

Binary sum and carry tables

- Binary addition table:

B

+	0	1
A 0	0	1
1	1	10

- As before, break into sum and carry tables:

B

+	0	1
A 0	0	0
1	0	1

Carry

B

+	0	1
A 0	0	1
1	1	0

Sum

More logically...

Re-writing these 2 tables more "logically"

Carry

+	B = 0	B = 1
A = 0	0	0
A = 1	0	1

Sum

+	B = 0	B = 1
A = 0	0	1
A = 1	1	0

We get:

A	B		Carry	Sum
0	0		0	0
0	1		0	1
1	0		0	1
1	1		1	0

Adding with logic gates

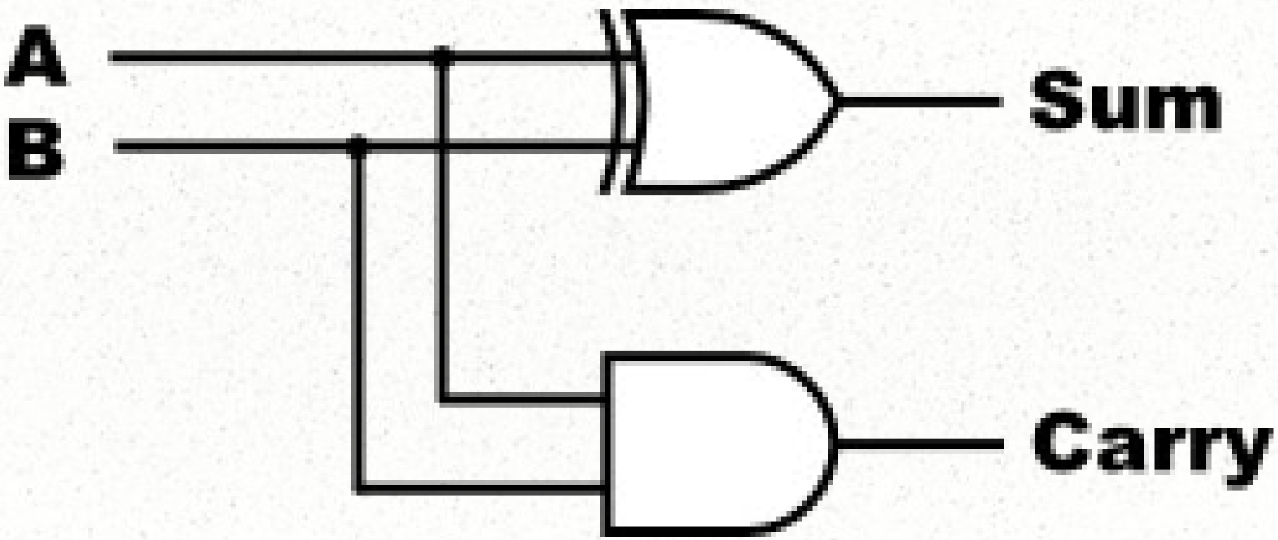
Compare this with our boolean logic gates:

A	B		Carry	Sum		AND	XOR
0	0		0	0		0	0
0	1		0	1		0	1
1	0		0	1		0	1
1	1		1	0		1	0

Carry is AND

Sum is XOR

The half-adder



Half-adder activity

- Use cards like with the transistor
  - AND, XOR gate cards + yarn for “wire”
- Build a half-adder
- Verify that it produces the binary addition table

Adding multiple digits

- A half-adder only adds 2 single digit numbers
  - It only does "half" the job
- We need to be able to handle both carries
  - Carry coming in from previous digit
  - Carry going out to next digit

Multi-digit addition (decimal)

	0	0	1	0
		2	7	6
+			1	8
		2	9	4

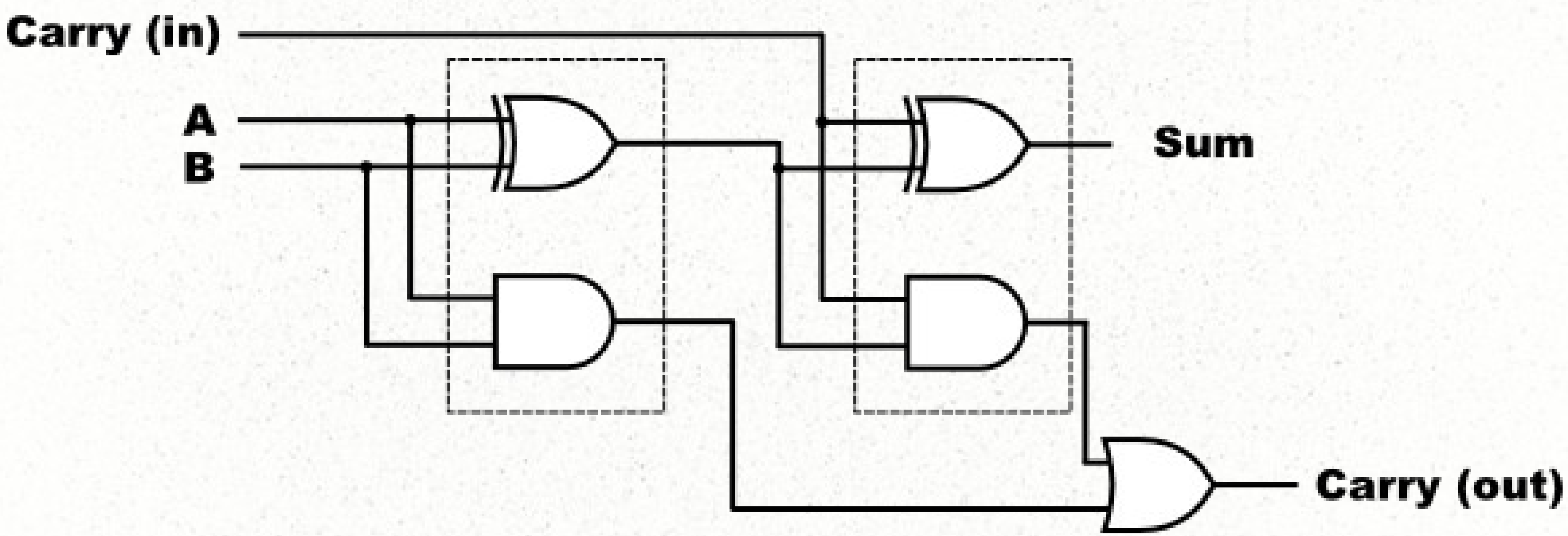
Add the 3 blue numbers

Produce the 2 green numbers

Supporting carries with a full-adder

- A full-adder extends the half-adder by adding support for the carry
- Two steps:
  - Add the two numbers (as before)
  - Add the carry to the result
- Built from 2 half-adders
  - Plus an OR gate to combine the carries

Full-adder



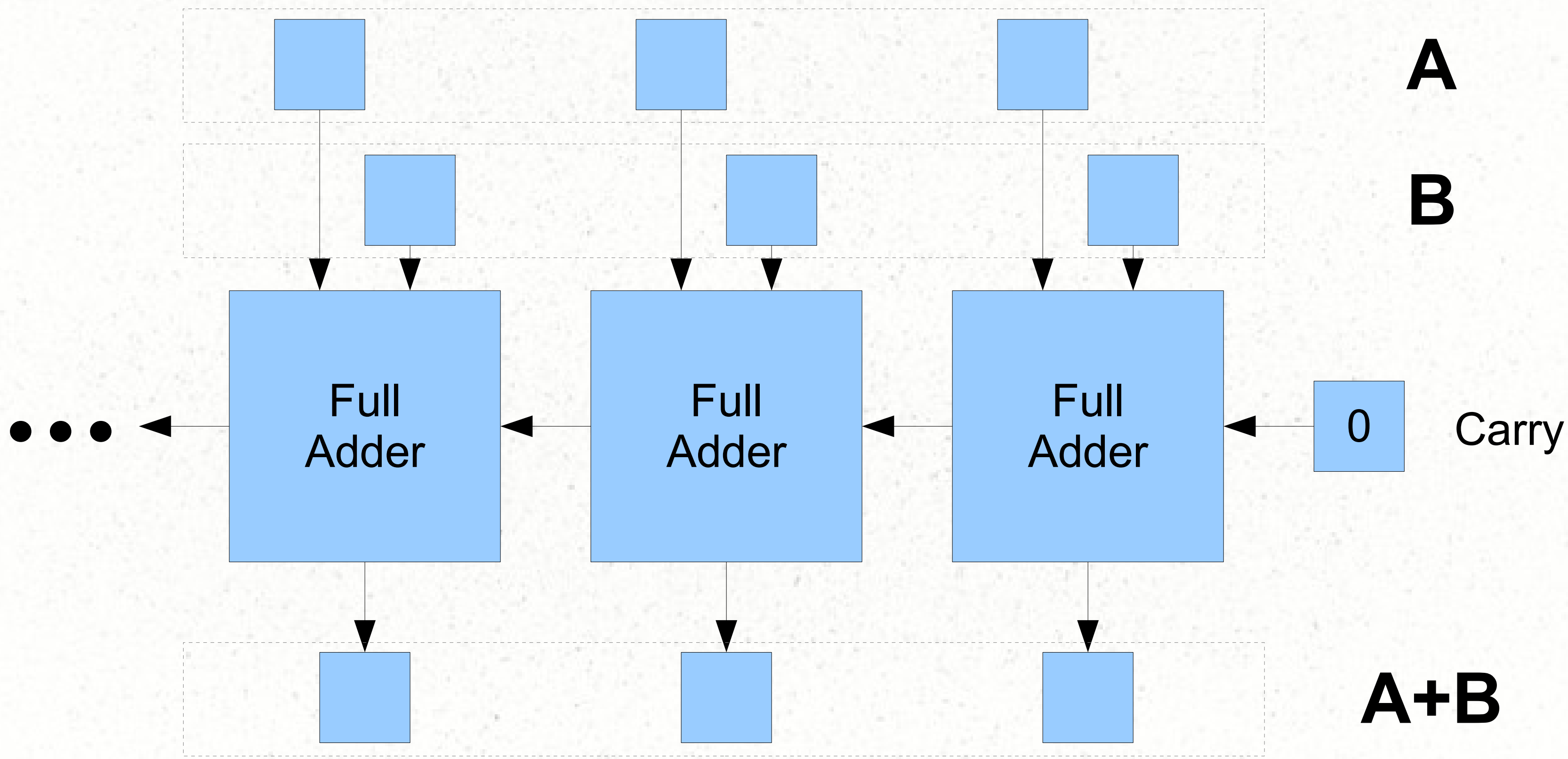
Full-adder activity

- Using cards, construct a full-adder and verify it works
- Investigate why we can OR the carries together
  - Each half-adder produces a carry
  - The full-adder just ORs the 2 carries together
  - Is it possible for both carries to be set at the same time?

Ripple carry adder

- Full adders connected together
  - Propagate the carry from one digit to the next
  - Just like we teach students to do for decimal
- Any number of full adders can be connected
  - 4 full adders supports adding two 4-digit numbers

Ripple carry adder



Ripple carry adder activity

- With N students, give each one a full-adder to construct.
  - Connect them end-to-end to build a ripple carry adder
  - Add 2 N-digit binary numbers

Resources

- Slides will be posted at:
  - cse4k12.blogspot.com
  - Updates will also be posted there
- We are in the process of converting these activities into worksheets/activity sheets.
- Email contact:
  - garykac @ either “gmail” or “google” .com