Resources for CS1 and CS2

https://faculty.knox.edu/dbunde/teaching/ipdps26/

A traditional CS1 course

- Teaches how to write programs
  - For a single core
  - Without graphics
  - Without other machines or a network
  - Entire programs, starting with a blank file
  - On your own
  - With limited tools (no build tools, no version control, …)
- All using lecture pedagogy

```java
public class HelloWorld {
        public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

CDER (Curriculum Development and Educational Resources)

- Created new versions of the first two courses:
  - In Java (at Knox College): https://drive.google.com/drive/folders/1cMzjVculBfxHtpkEJ88GBusfMWfNbBCW?usp=sharing
  - In C++ (at Tennessee Tech): https://drive.google.com/drive/folders/1zddkyEZt73EB0_StKk1sT2USAzqVHkdu?usp=sharing
- Concepts to bring in:
  - Data parallelism
  - Interaction with server (distributed computing)
  - Event handling

Knox College CS-1 course structure

- 10 week courses, but last week is mostly finals
- Three 70-minute class periods, plus one 70 min lab period
- CS-1 “lab” is first hour of large weekly coding assignment
  - Ensure everyone can install and run assignment
  - Identify and fix problems
  - Help students get started, build momentum
  - Assignment due to LMS before next lab period
- Two in-class written midterms, plus written final
  - Always give students exam from last term as “practice exam”

Pedagogy: “flipped classroom” Instruction

flipped classroom

Textbook/
Online mini-lecture

First Exposure:

QUIZ

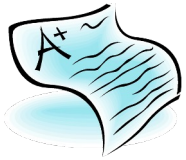
“Lecture”

Learn Hard Stuff:
With teacher and discussion

Homework

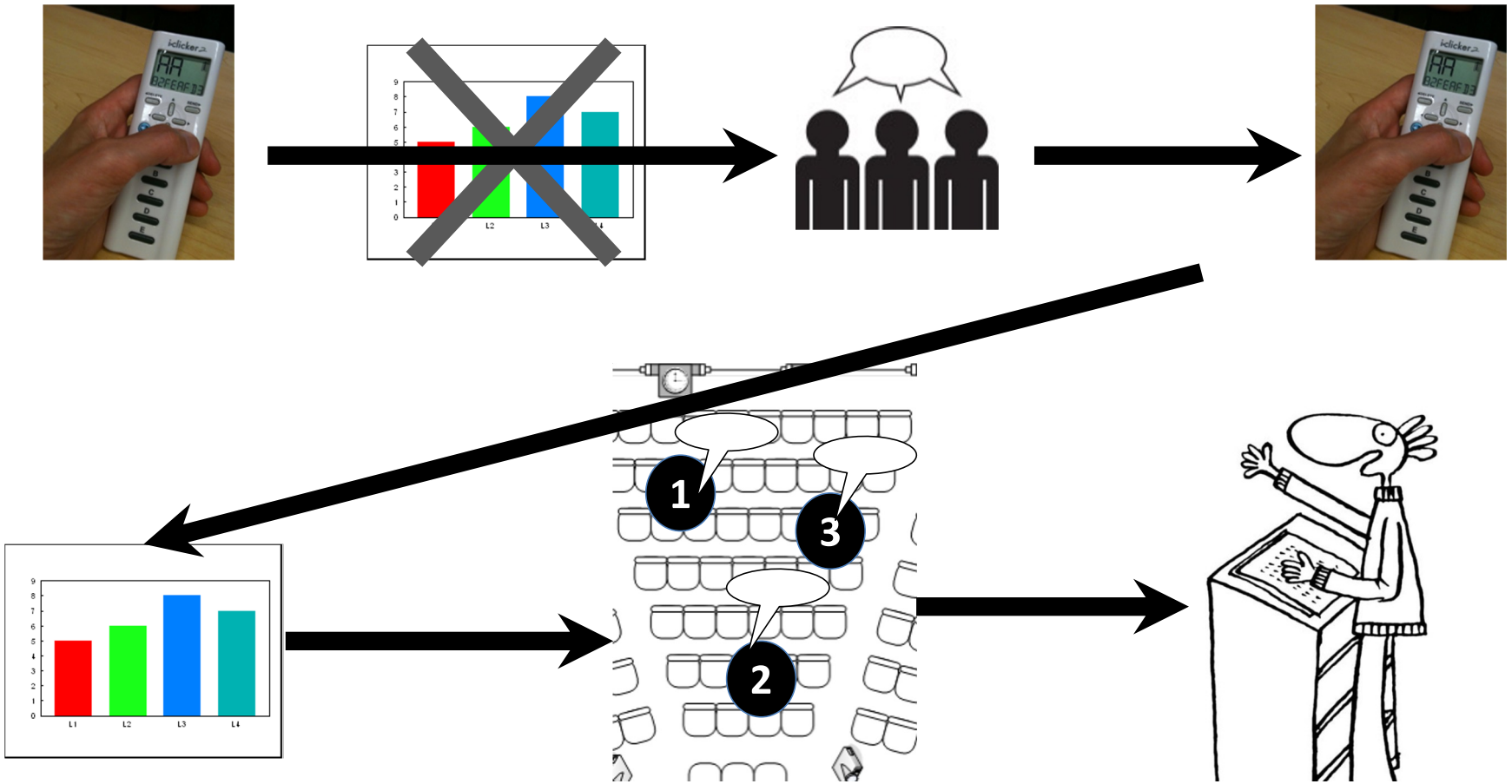
See if You Know Hard Stuff

Exam



Show Knowledge Mastery

Peer Instruction: Clicker Questions



Weekly schedule

Week	Topics	Weekly Coding Assignment	Notes
1	variables, types	Install VS Code	
2	conditionals, loops	Java Turtles (Logo)	"AI honeypot"
3	strings	Flagmaker	Flagmaker, Unplugged Flagmaker
4	functions	Codingbat #1 (short functions)	custom problems
5	arrays	Codingbat #2, Testing	
6	2D arrays, event classes	~~Event Calendar~~ Minecraft or Tic-tac-toe	JSON, Distributed events
7	event driven programming, writing classes	Greenfoot (week 1)	Event driven programming
8	writing classes	Greenfoot (week 2)	Event driven programming
9	Recursion	Recursion	
10	wrap up, finals		

“AI honeypot”

First Java coding assignment in week 2 uses Turtle Graphics, a reimplementation of Logo in Java

If you paste the assignment description into AI, it will generate Swing code, not Turtle Graphics

Can catch people right away, send to honor board, hopefully communicate to students that we care that they do things the right way

Flagmaker

Draw a diagonal line

A

```
for (int i=0; i<g.getNumRows(); i++) {
    g.setColor(i, i, Color.BLUE);
}
```

B

```
for (int i=0; i<g.getNumRows(); i++) {
        g.setColor(0, i, Color.BLUE);
}
```

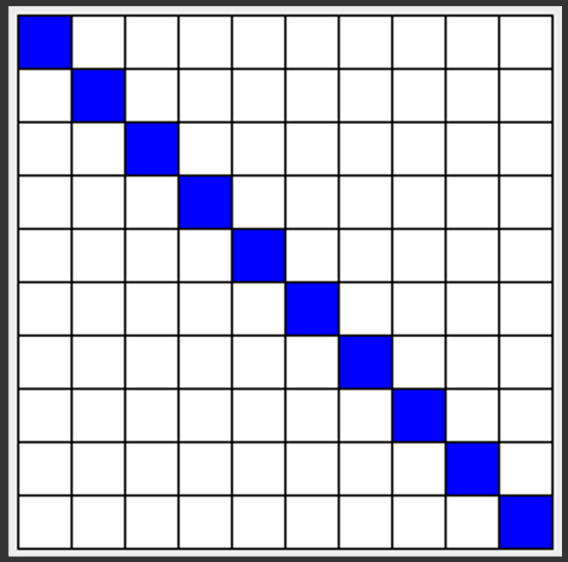
C

```
for (int i=g.getNumRows()-1; i>=0; i--) {
        g.setColor(i, i, Color.BLUE);
}
```

D

```
for (int i=g.getNumRows(); i>0; i--) {
        g.setColor(i, i, Color.BLUE);
}
```

E Not exactly one of the above



Draw a diagonal line

A

```
for (int i=0; i<g.getNumRows(); i++) {
    g.setColor(i, i, Color.BLUE);
}
```

B

```
for (int i=0; i<g.getNumRows(); i++) {
        g.setColor(0, i, Color.BLUE);
}
```

C

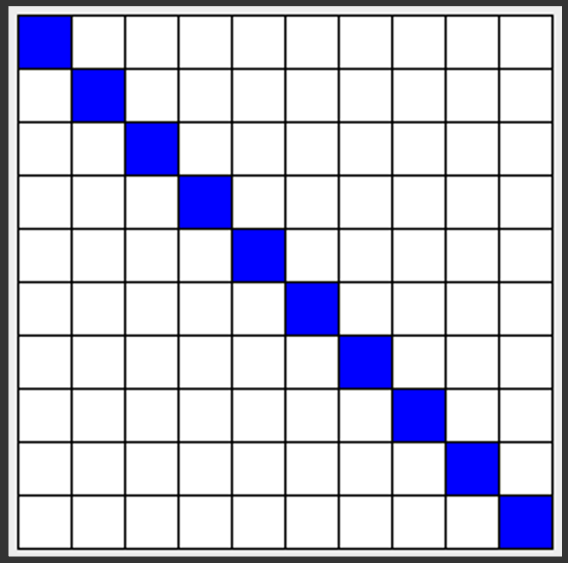
```
for (int i=g.getNumRows()-1; i>=0; i--) {
        g.setColor(i, i, Color.BLUE);
}
```

D

```
for (int i=g.getNumRows(); i>0; i--) {
        g.setColor(i, i, Color.BLUE);
}
```

E Not exactly one of the above

Two correct answers!
Plot twist!

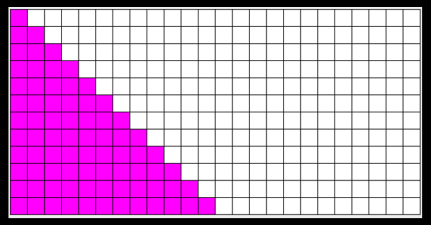


Which is better?

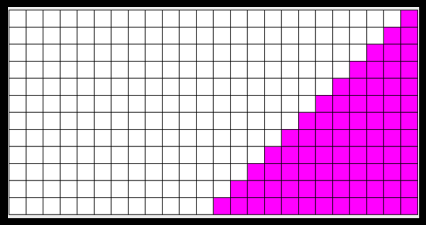
What squares will the code below color?

```
// assume grid is 10x20
for (int i = 0; i<grid.getNumRows(); i++) { //all the rows
    for (int j = 0; j<=(2*i+1); j++) { // from col [0] to col [2*row+1], inclusively
        grid.setColor(i, j, Color.MAGENTA);
    }
}
```

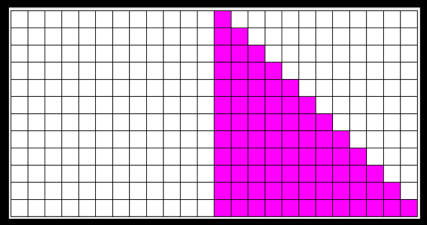
A



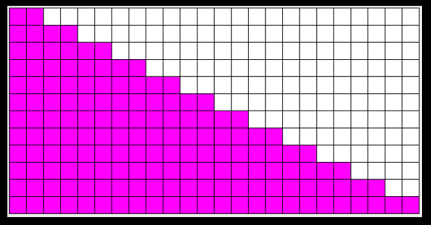
C



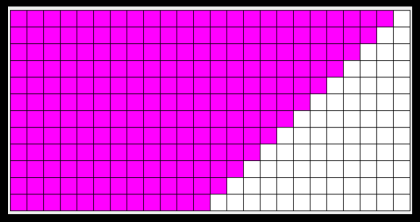
E



B



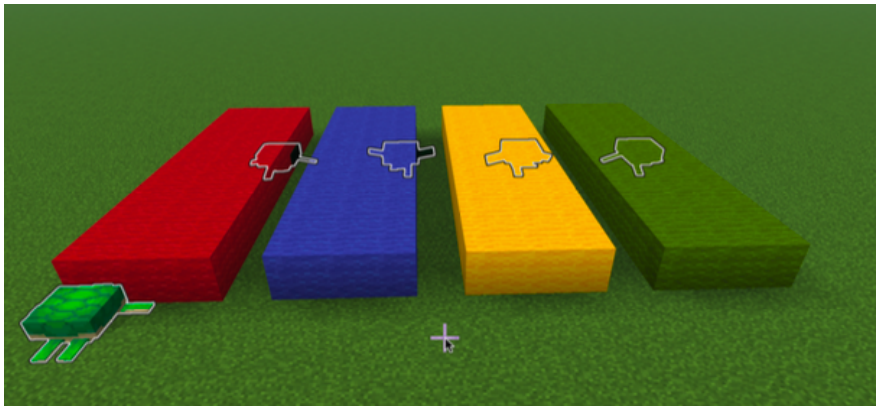
D



3D turtles in Minecraft

(Jurdana Masuma Iqrah, Jaime Spacco, Michael Gerten, EduPar 2026)

- Students write code (Java, Python, et al.) to control a “turtle” in Minecraft
- Drawing happens in a modded Minecraft server, which students can view in unmodified client
- Can create multiple turtles to draw in parallel



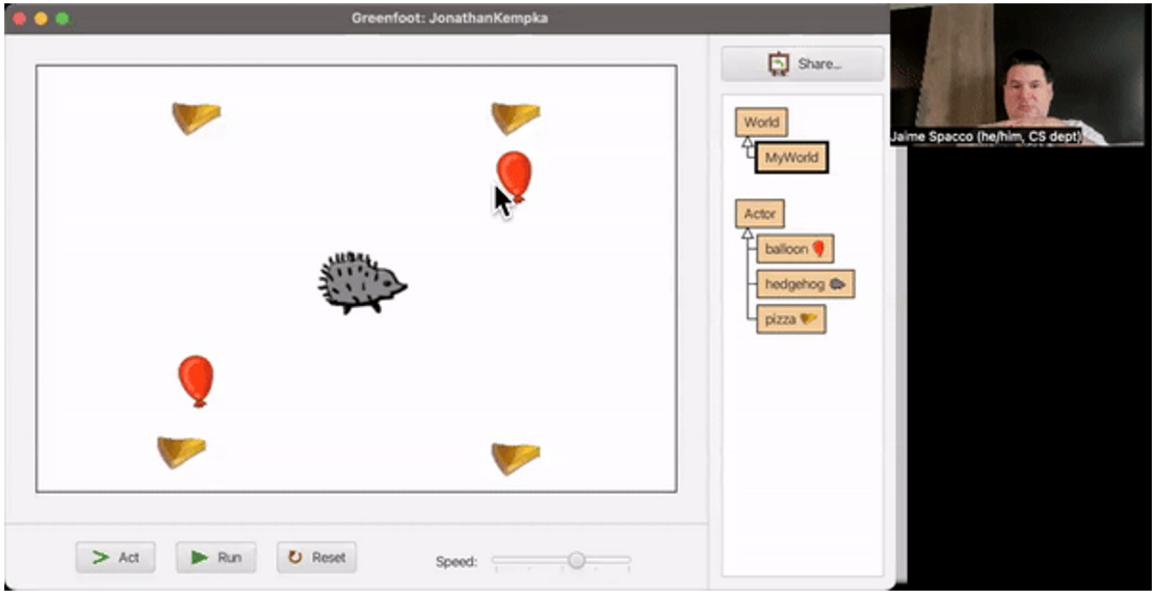
Come see more tomorrow (5/26) at 2:30pm session of EduPar in Salon D

Greenfoot

- Free educational Java IDE: www.greenfoot.org
- Add sprites and code for them
- Really a simple game engine for novice Java programmers

Simple game found online:

Hedgehog needs to eat
Cheese and dodge balloons



```
int speed = 2;
int Lives = 3;
int Points = 0;

public void act()
{
    moveWithKeys();
    showStatus();
    hitPizza();
    hitBalloon();
    testEndGame();
    testWinGame();
}
```

Hedgehog state

Act() method called each frame (e.g. 30 fps)

Code to move Hedgehog

```
public void moveWithKeys()
{
    if (Greenfoot.isKeyDown("right") )
    {
        setLocation(getX()+speed, getY() );
    }
    if (Greenfoot.isKeyDown("Left") )
    {
        setLocation(getX()-speed, getY() );
    }
    if (Greenfoot.isKeyDown("Up") )
    {
        setLocation(getX(), getY()-speed );
    }
    if (Greenfoot.isKeyDown("Down") )
    {
        setLocation(getX(), getY()+speed );
    }
}
```

Status updates and display

```
public void showStatus()
{
    getWorld().showText("Lives"+Lives,50,30);
    getWorld().showText("Points"+Points,50,10);
}

public void addPoints()
{
    Points = Points + 1;
}

public void loseLife()
{
    Lives = Lives - 1;
}
```

Collisions with pizza and balloons

```
public void hitPizza()
{
    if (isTouching (pizza.class) )
    {
        removeTouching (pizza.class);
        addPoints();
    }
}

public void hitBalloon()
{
    if (isTouching (balloon.class) )
    {
        setLocation(300,100);
        loseLife();
    }
}
```

Win and lose conditions

```
public void testEndGame()
{
    if (Lives<1)
    {
        getWorld().showText("YOU LOSE",300,200);
    }
}

public void testWinGame()
{
    if (Points>3)
    {
        getWorld().showText("YOU WIN",300,150);
    }
}
```

Entire balloon code

```
import greenfoot.*;  // (World, Actor, GreenfootImage

/**
 * Write a description of class balloon here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class balloon extends Actor
{
    /**
     * Act - do whatever the balloon wants to do. Thi
     * the 'Act' or 'Run' button gets pressed in the
     */
    public void act()
    {
        move(1);
        if (isAtEdge() )
        {
            turn(180);
            move(1);
        }
    }
}
```

Use in CS1

- Open-ended assignment: Make a game! (w/ >=X sprites and >=Y functions)
  - Tradeoff between students who get excited and students who do the minimum

- Advantages
  - Students love it
  - Easy to install
  - Compact API with useful functionality (like collision detection)

- Disadvantages
  - Default graphics looks middle school, but easy to add
  - Different IDE from whatever you use
  - “Complex” things require scaffolding: animation, moving between screens
  - Lots of online samples and tutorials that students can mindlessly follow (probably disallow snake game, flappy bird, …)

Changes to CS2

- Less writing code for list/stack/queue (still clicker questions)
- More emphasis on iterators (modifying and not just reading)
- Design problems
- Parallelism using fork/join (following Dan Grossman, UWash)

Sample design problem

You're cleaning your dorm room, but every time you start one job you find another that needs your attention first! You try to put away your laundry, but it's all dirty! Then when you try to wash your laundry you find that your laundry basket is filled with junk you've been too lazy to put away, and when you try to put some stuff away you find some nasty old moldy food that you have to clean up, etc. Note that each time you discover a new task, it must be completed before the one that you were previously working on. What ADT would you use to help you keep track of all the subtasks you have to do before your room is finally clean?

(A. List B. Stack C. Queue D. Map or Set E. None of the above)

Labs

1 (day 1): Implementing Array-based Bag

2: Testing PrintQueue

3: Implementing linked lists

4: Timing data structures

5: Exploring a cave (backtracking w/ stacks)

6: Implementing lists w/ recursion

7: Navigation system (comparators and iterators)

8: Building friendship graph

9: Parallelizing heat diffusion

Labs

1 (day 1): Implementing Array-based Bag

2: Testing PrintQueue

~~3: Implementing linked lists~~

4: Timing data structures

5: Exploring a cave (backtracking w/ stacks)

7 ~~6~~: Implementing lists w/ recursion

~~7: Navigation system (comparators and iterators)~~

6 ~~8~~: Building friendship graph

~~9: Parallelizing heat diffusion~~

3: Finding common queries (iterators)

8: Prime counting with fork-join

9: Reductions with fork-join

Fork-join

```
int sum(int[] array, int low, int hi) {
  if(hi==low+1)
    return array[low];
  int left = fork sum(array, low, (hi+low)/2);
  int right = sum(array, (hi+low)/2, hi);
  join;
  return left + right;
}


int result = sum(…);
```

Fork-join

```
int sum(int[] array, int low, int hi) {
  if(hi==low+1)
    return array[low];
  int left = fork sum(array, low, (hi+low)/2);
  int right = sum(array, (hi+low)/2, hi);
  join;
  return left + right;
}


int result = sum(…);
```

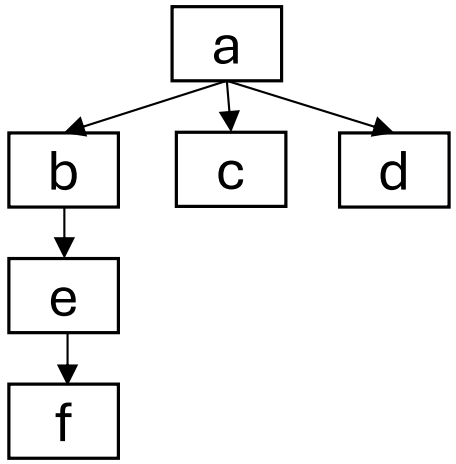
```
class Sum extends RecursiveTask<Integer> {
   //attributes and constructor

   protected Integer compute(){
      if(hi – low < THRESHOLD) …
      Sum left = new Sum(array, low, (hi+low)/2);
      Sum right = new Sum(array, (hi+low)/2, hi);
      left.fork();
      return right.compute() + left.join();
   }
}


sum = ForkJoinPool.commonPool().invoke(
                new Sum(array, 0, array.length));
```

Beyond the code

- Precedence diagrams
  - Drawing based on code, running them



- Work and span metrics

- Without terminology: non-determinism, load balance, speedup