Introduction to Distributed Computing

(with hands-on Squeak demo for K-12)

Supercomputing 2005

Randy Heiland
Scientific Data Analysis Lab
heiland@indiana.edu

pervasivetechnologylabs
AT INDIANA UNIVERSITY

What is Distributed Computing?

- **Wikipedia: “the coordinated use of physically distributed computers”**

We’re going to assume you know what a computer is, so we’ll try to explain what is meant by “physically distributed” and “coordinated use”.

“physically distributed” computers

Q: What distinguishes two (or more) coordinated computers being:

- **within a room?**
- **within a building?**
- **within a State | continent | planet?**

pervasivetechnologylabs
AT INDIANA UNIVERSITY

Answer: The Network!

- “coordinated” computers implies some sort of communication – electronic, optical, or telepathic ☺
- Communication between two computers separated by an ocean will (probably) take longer than two computers within a room
- The underlying application – the computation – will determine how important communication needs to be

Communication models

- Tightly-coupled
  - processors need to exchange data frequently
- Loosely-coupled
  - processors exchange data infrequently
- Uncoupled
  - processors work independently

Decomposing a problem

- How do we decompose (partition, break-up) a problem so that we can take advantage of distributed computing?
- Two basic strategies:
  - Data decomposition
    - Have each computer perform the same calculation, but just on its particular chunk of data
  - Task decomposition
    - Have each computer perform different tasks/ calculations (perhaps on very same data)

Parallel vs. Distributed vs. Grid

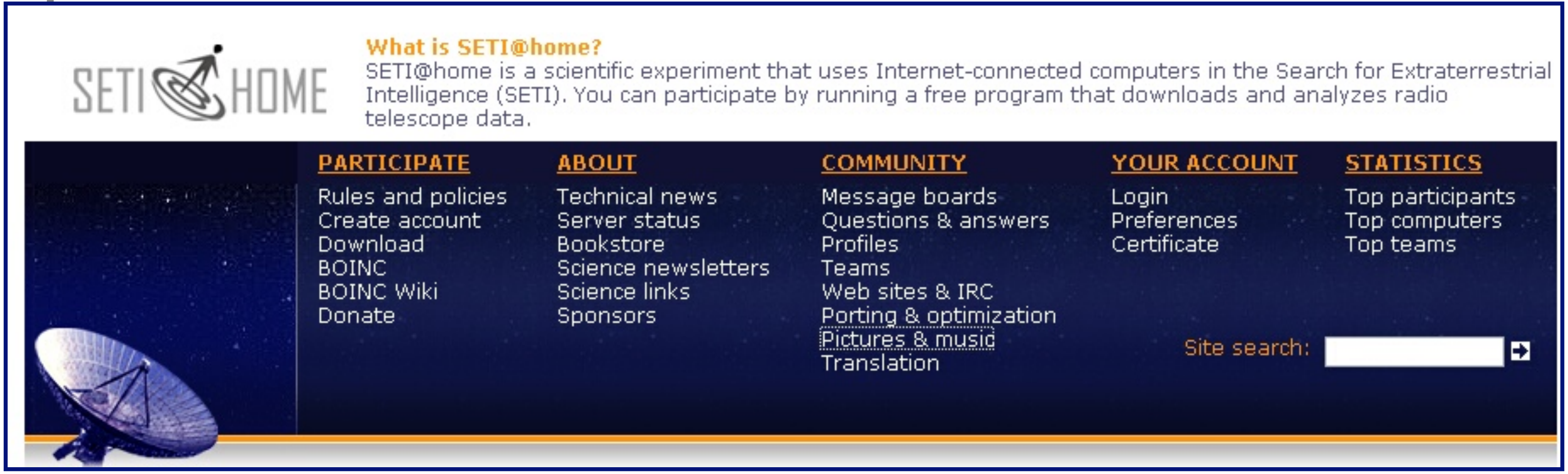
Although an oversimplification, let’s describe these 3 computing paradigms as follows:

- Parallel computing – in single machine room; very fast network; homogeneous computers
- Distributed computing – “spatially” larger than Parallel; distributed memory; heterogeneous computers
- Grid computing – “spatially” larger than Dist’d; may involve dist’d parallel computers

Let’ s look at some “successful” examples of distributed computing projects…

From the very large – SETI@home

- Search for Extraterrestrial Intelligence
- Started 1999

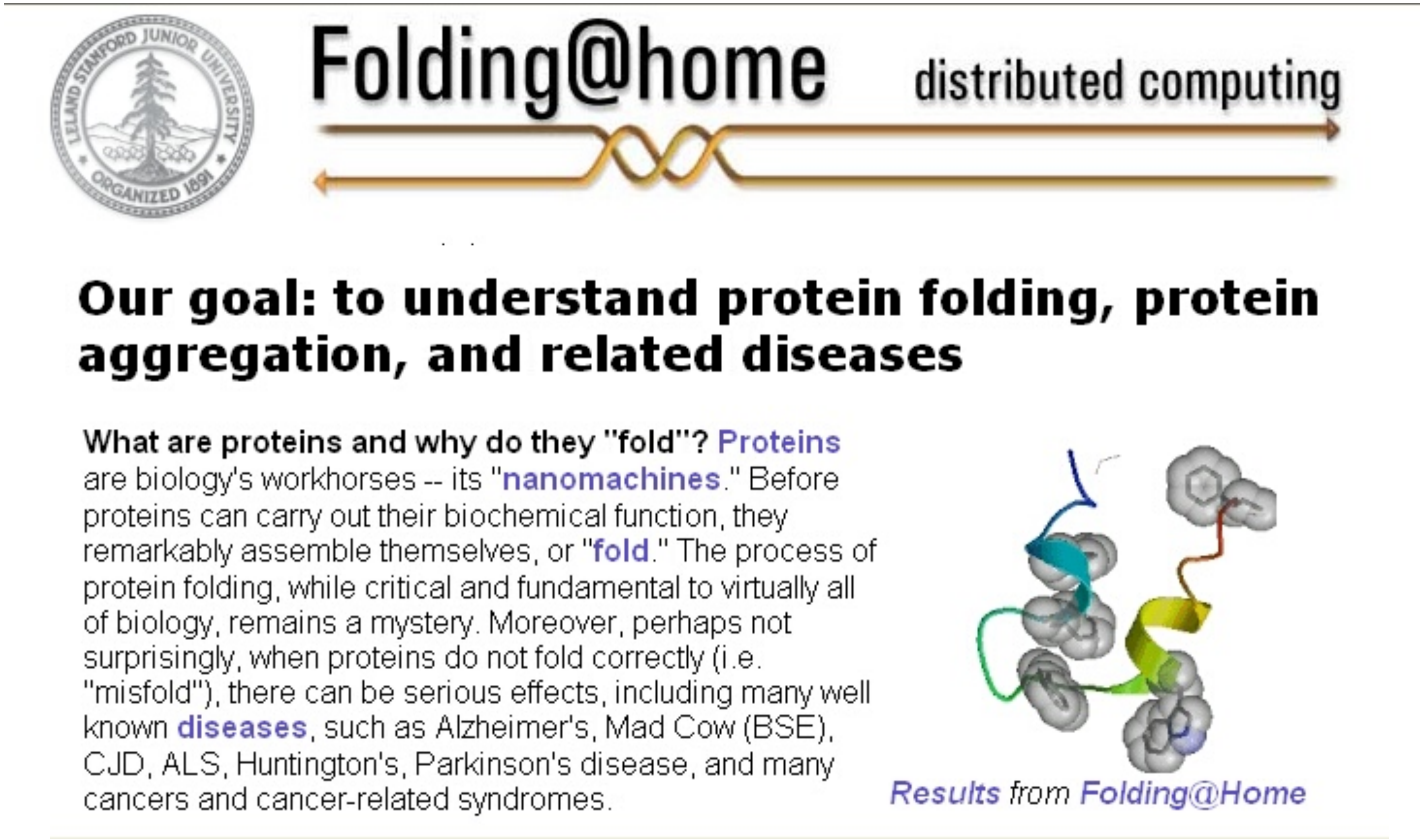


What is SETI@home?

SETI@home is a scientific experiment that uses Internet-connected computers in the Search for Extraterrestrial Intelligence (SETI). You can participate by running a free program that downloads and analyzes radio telescope data.

pervasivetechnologylabs
AT INDIANA UNIVERSITY

To the very small



Folding@home distributed computing

Our goal: to understand protein folding, protein aggregation, and related diseases

What are proteins and why do they "fold"? **Proteins** are biology's workhorses -- its "**nanomachines**." Before proteins can carry out their biochemical function, they remarkably assemble themselves, or "**fold**." The process of protein folding, while critical and fundamental to virtually all of biology, remains a mystery. Moreover, perhaps not surprisingly, when proteins do not fold correctly (i.e. "misfold"), there can be serious effects, including many well known **diseases**, such as Alzheimer's, Mad Cow (BSE), CJD, ALS, Huntington's, Parkinson's disease, and many cancers and cancer-related syndromes.

Results from Folding@Home

For the mathematically-inclined

http://www.mersenne.org/

GIMPS
The Great Internet Mersenne Prime Search
Finding 8 World Record Primes!
2^P-1
MAY BE PRIME!

Previous 3 successful examples

- Not to trivialize them, but they have been successful, in part, because each has a communication model which is basically uncoupled, i.e. each processor is able to independently work on a chunk of data
- Life isn’t so easy with a tightly-coupled application

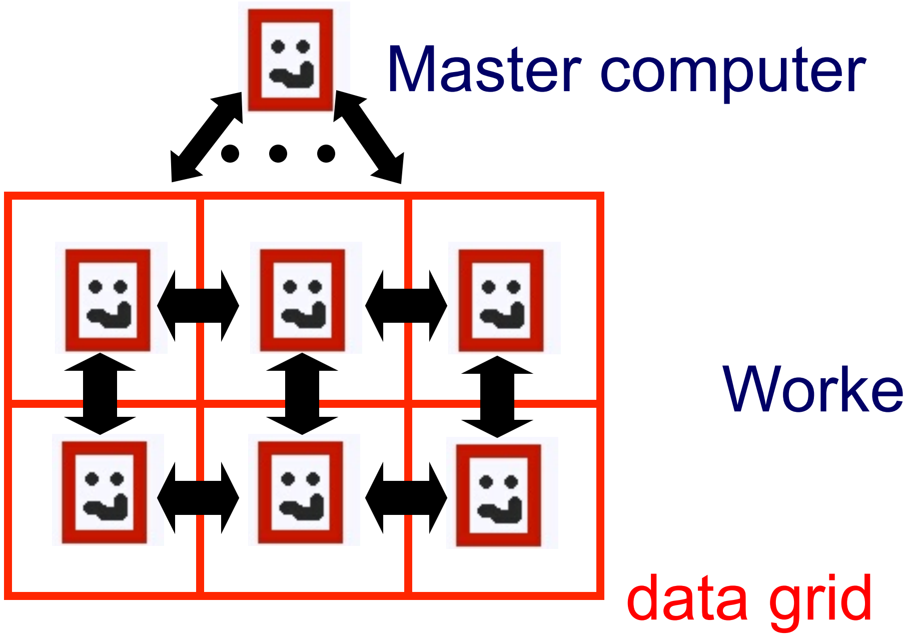
HOW do processors communicate?

- Message-Passing Interface (MPI) – de facto standard for computational science applications.

Two widely used, open source implementations:

  - MPICH
  - LAM/MPI

Communication patterns



One common pattern is found in applications involving data decomposition. Here each processor handles its section of data, communicating with its neighbors, as well as the “master”. There are many other patterns.

Advanced Topics

- Distributed Algorithms, Nancy A. Lynch

  MIT OpenCourseWare → Electrical Engineering and Computer Science

Enough lecture – let’s play!

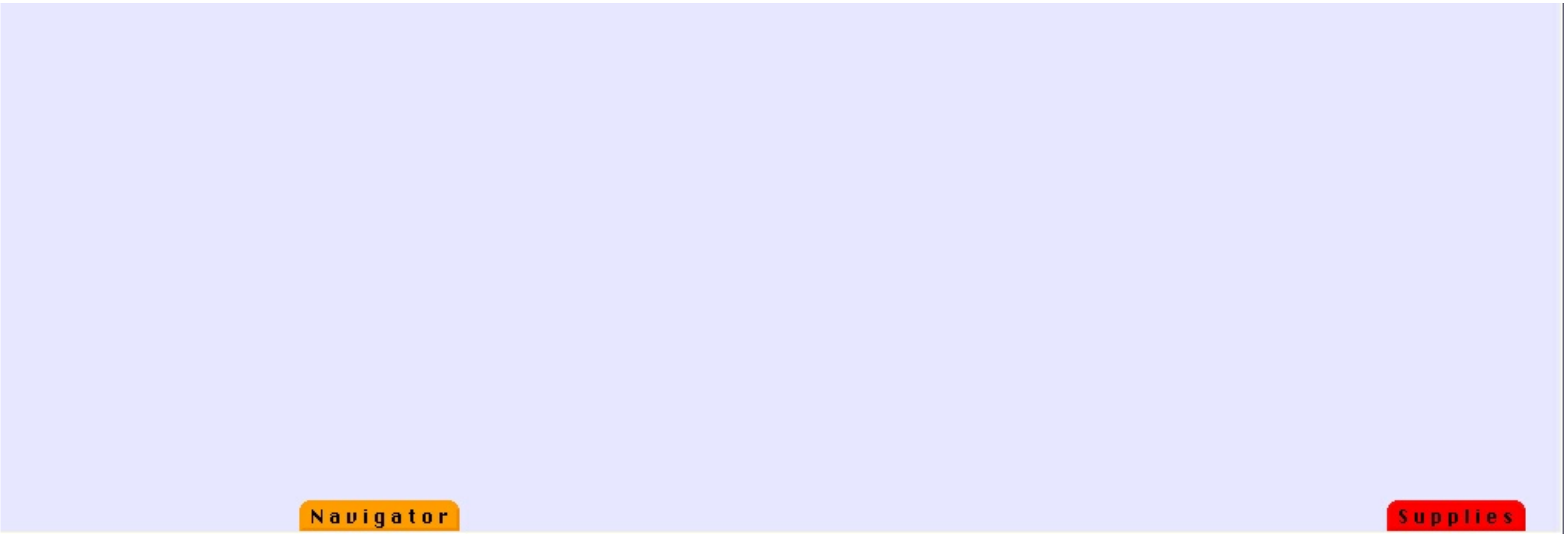
- Let’s explore some simulated parallel/ distributed computing concepts (and have some fun) using a freely available software package called Squeak.

Targeting K-12: squeakland.org

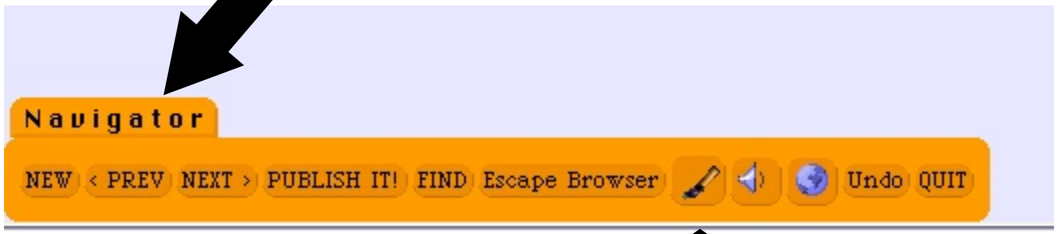
- Free/open source
- “Media authoring” tool
- Fun, interactive
- Math & science concepts through simulations
- Alan Kay – “father of the personal computer”

If you don’t have Squeak installed, you can get it from squeakland.org

Startup screen:

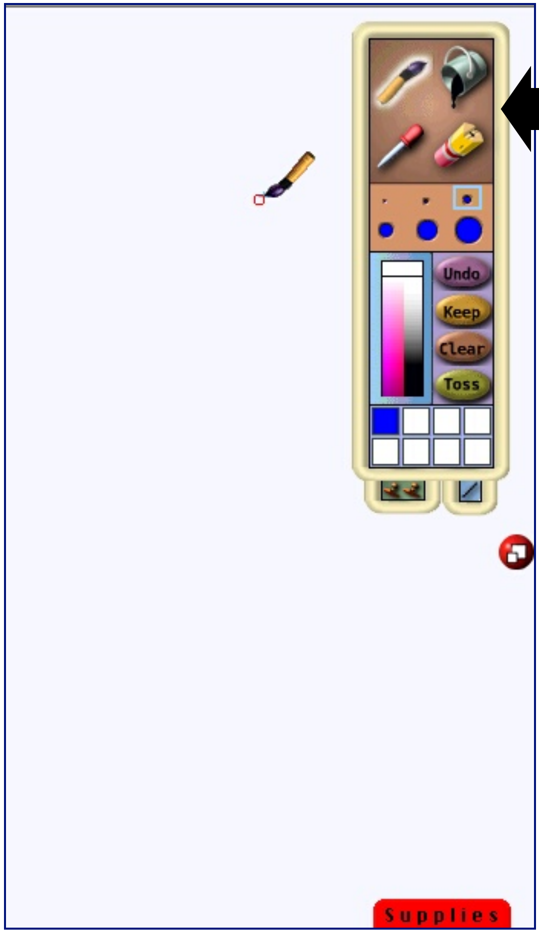


Click on Navigator tab to open it



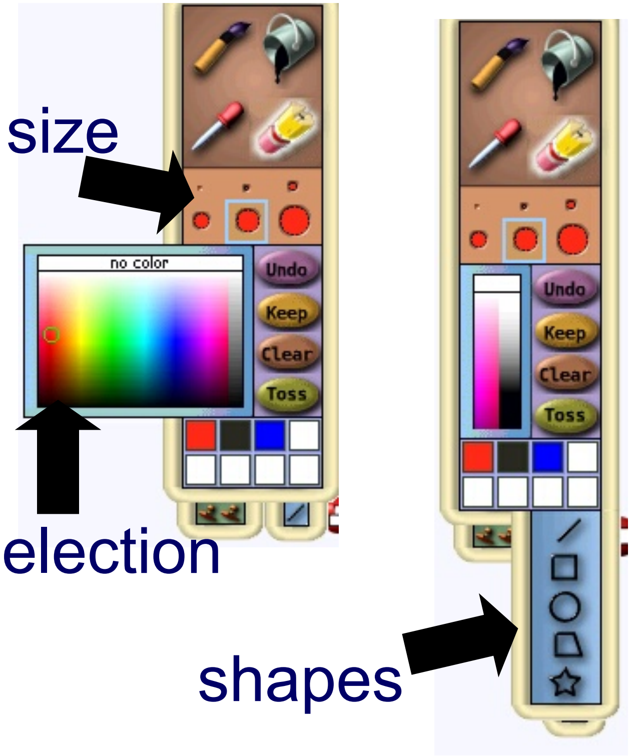
then click on the paintbrush icon

Painting/drawing palette



Paint brush, fill bucket,

eye-dropper(color selector), eraser



Brush size

color selection

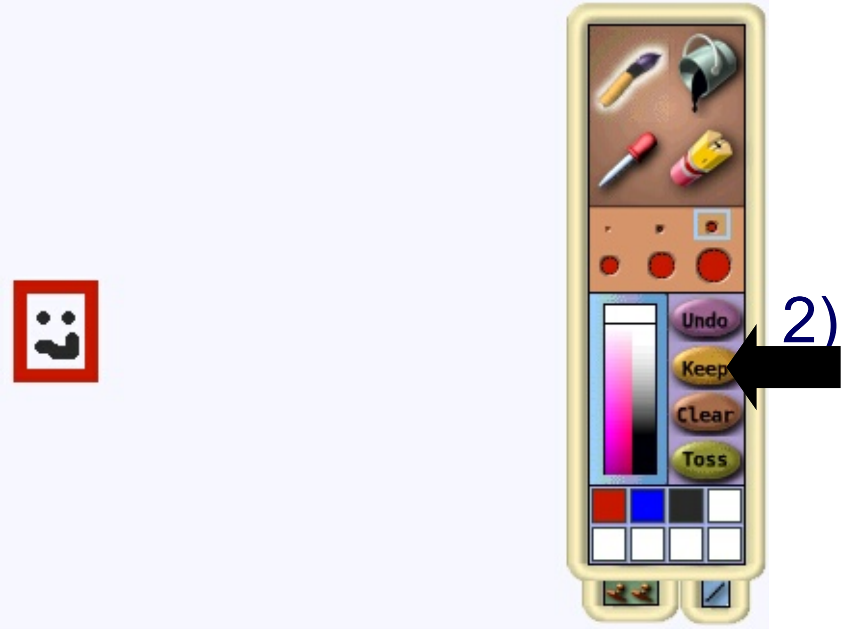
shapes

pervasivetechnologylabs
AT INDIANA UNIVERSITY

Simulated distributed computing

- Create a ‘computer’
- Make copies of it
- Have our ‘distributed computer system’ do something interesting

Create a ‘computer’ :

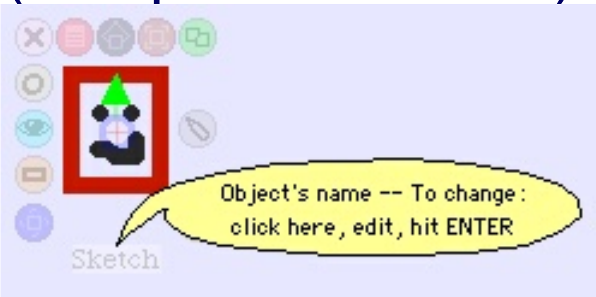


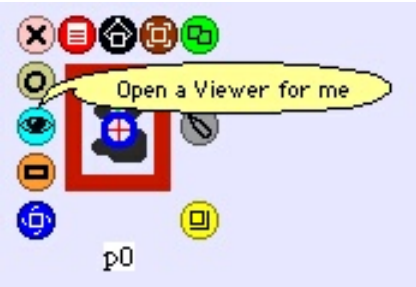
1) Draw a picture of your ‘computer’

“mousing over” a painted object should reveal a set of “halos”. Otherwise, on Windows, Alt-click the object. On Linux, Ctl-click. On Macs, Cmd-click.

2) Keep it

3) Rename it: “p0” (for “processor #0”)





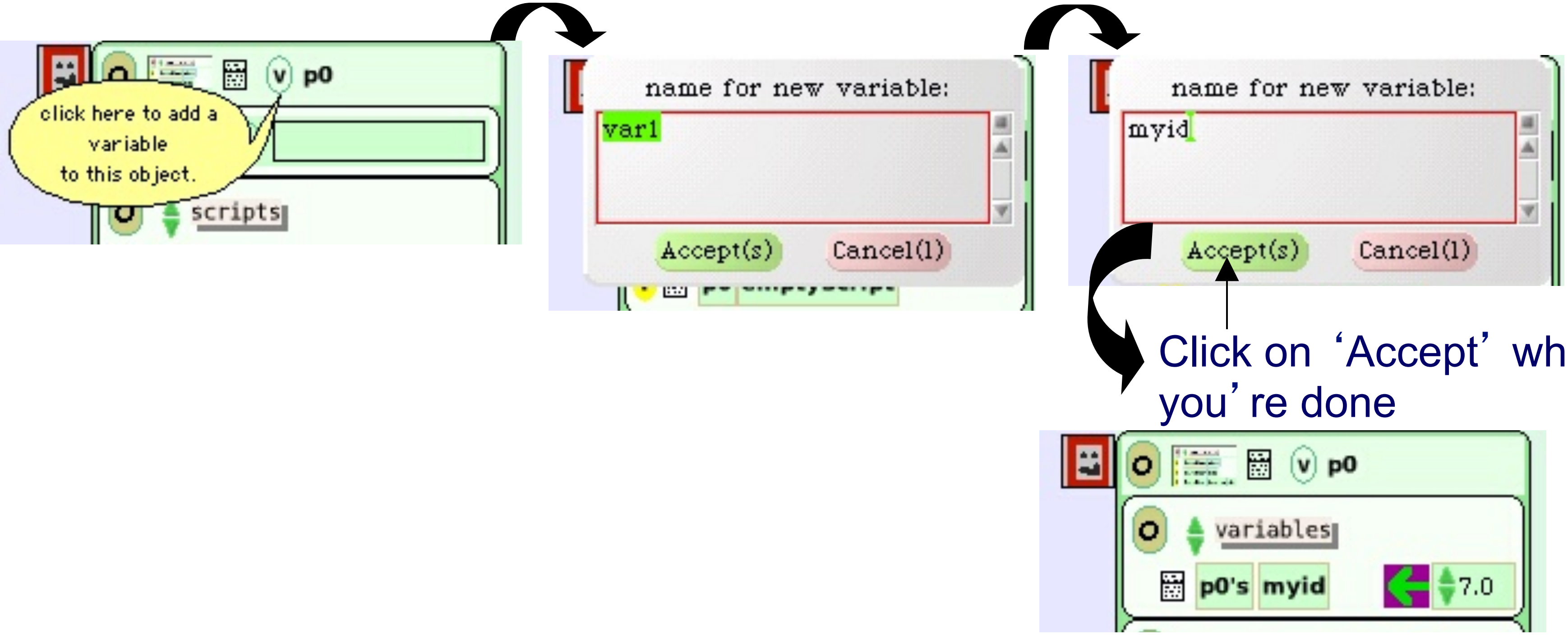
4) Open its Viewer

(click the “eyeball” halo)

pervasivetechnologylabs
AT INDIANA UNIVERSITY

Object-oriented programming

An object’s Viewer lets us ‘program’ the object.
A Viewer contains categories of draggable ‘tiles’.

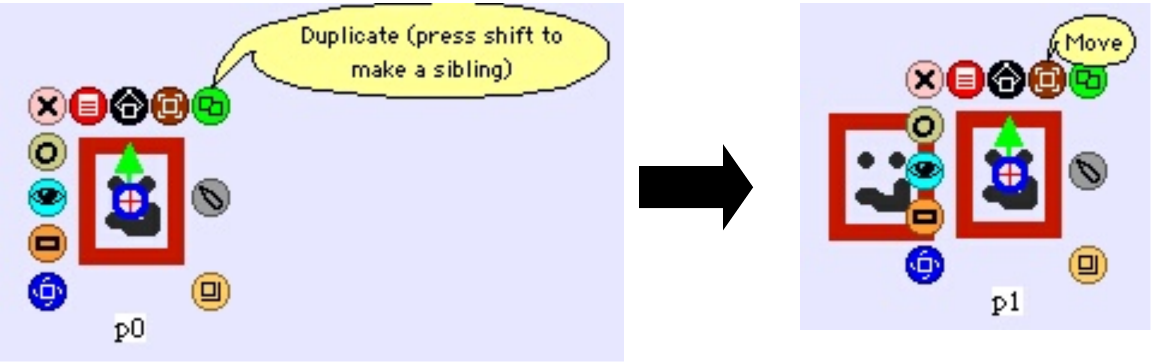


Click on ‘Accept’ when you’re done

Create a variable ‘myid’ for the object ‘p0’

(myid is a variable used in MPI programs that identifies each computer)

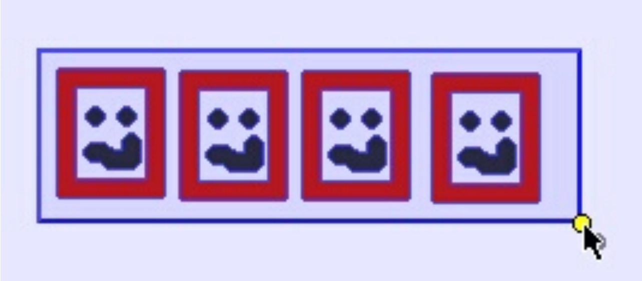
Clone p0 to make p1,p2,p3



Clone p0 by making a sibling (hold ‘Shift’ while pressing the Duplicate halo). Notice the name of the new sibling is ‘p1’.

Make another sibling of p0 (NOT of p1!) → p2

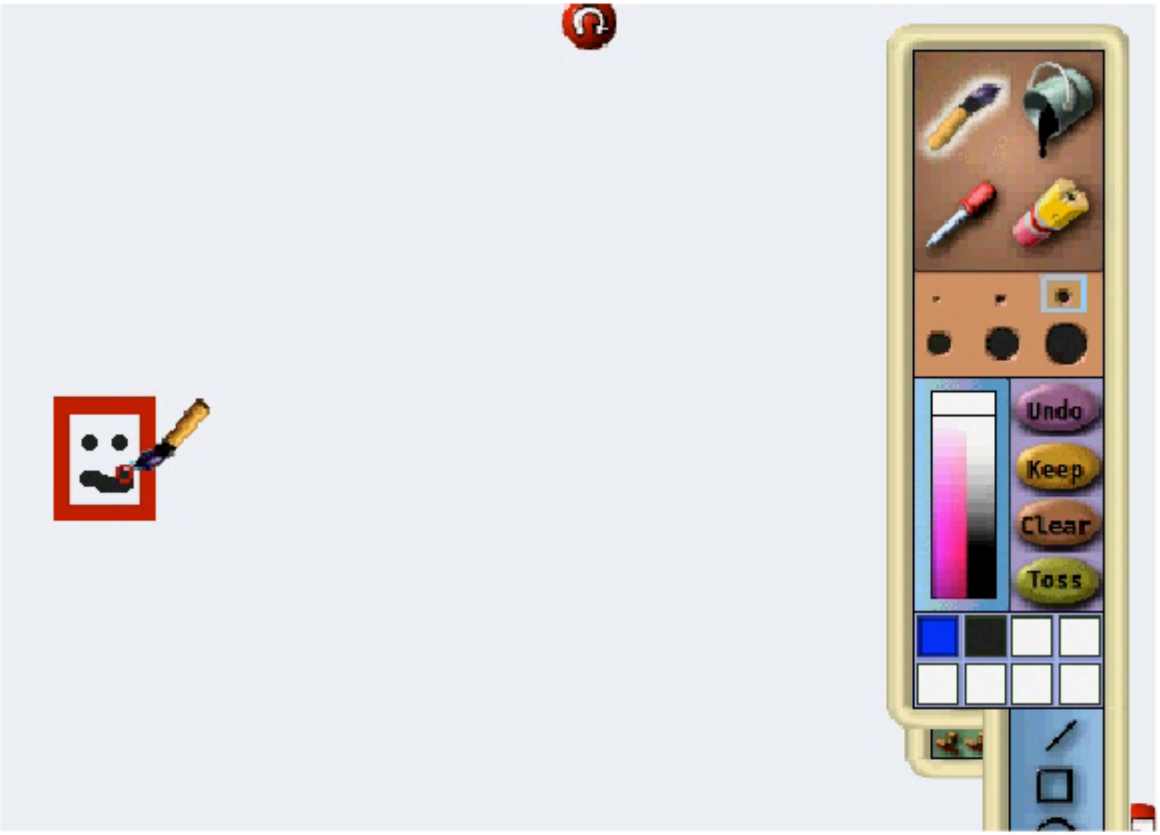
Make a third sibling of p0 → p3



If you need to move p0-p3, you can ‘Shift’ rubberband them to select the group.

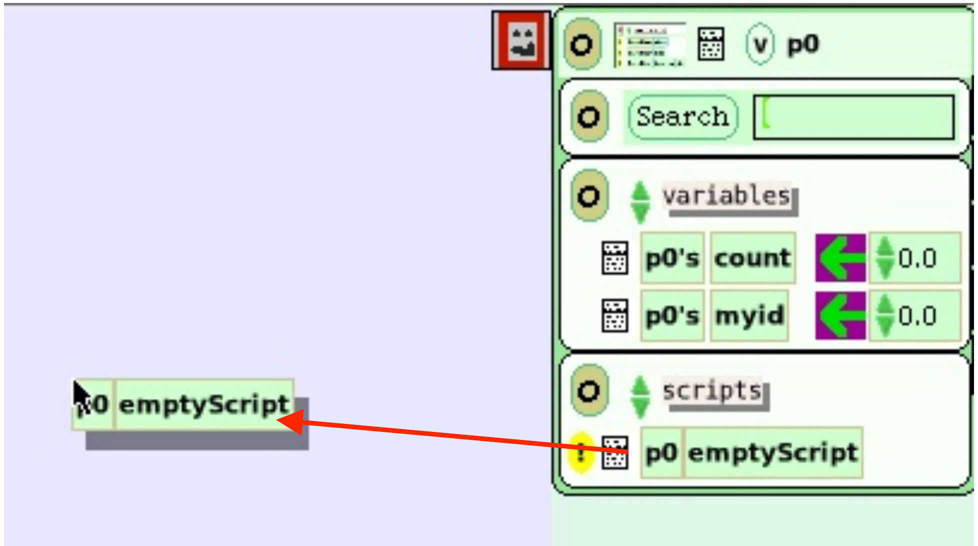
pervasivetechnologylabs
AT INDIANA UNIVERSITY

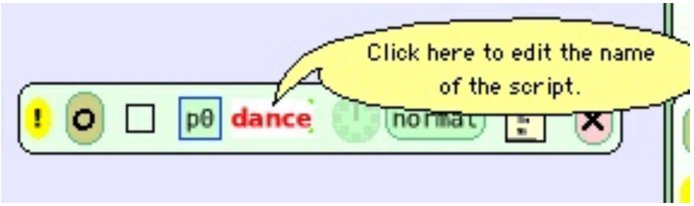
Flash movie: creating p0-p3



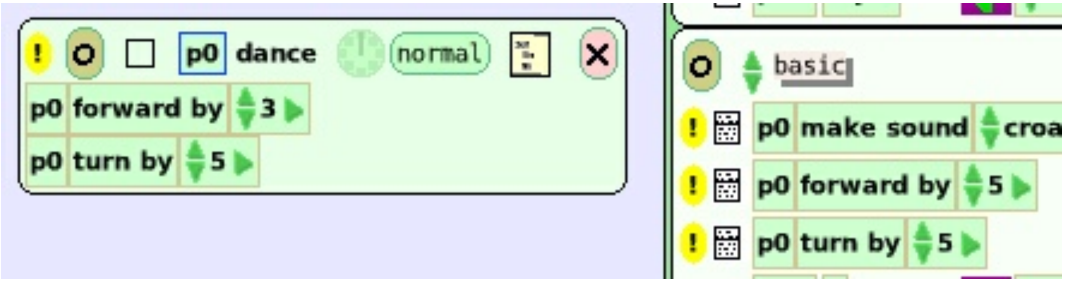
When making siblings of p0, be sure to press ‘Shift’ key while selecting the Duplicate halo.

Silly example: ‘dancing’ computers





Click here to edit the name of the script.



p0 forward by 3
p0 turn by 5

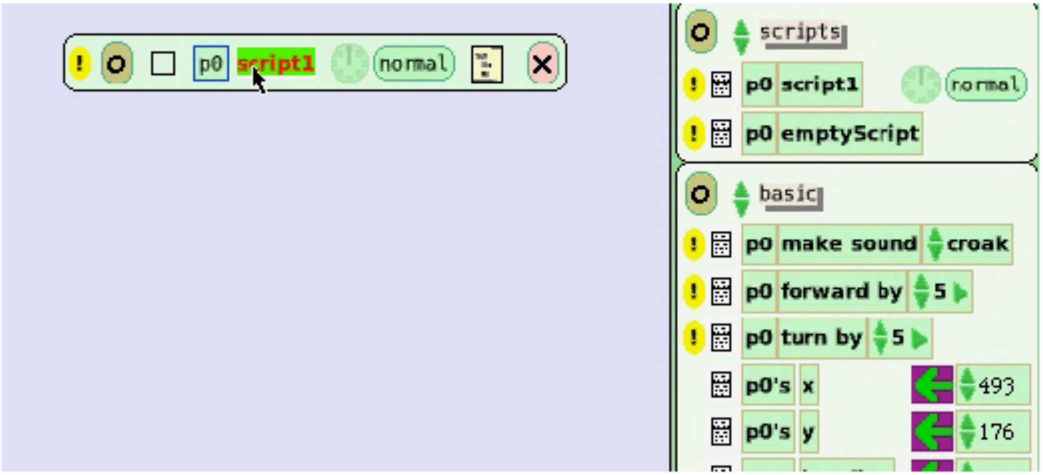
p0 make sound croa
p0 forward by 5
p0 turn by 5

Create a ‘dance’ script for p0:

- move forward
- turn

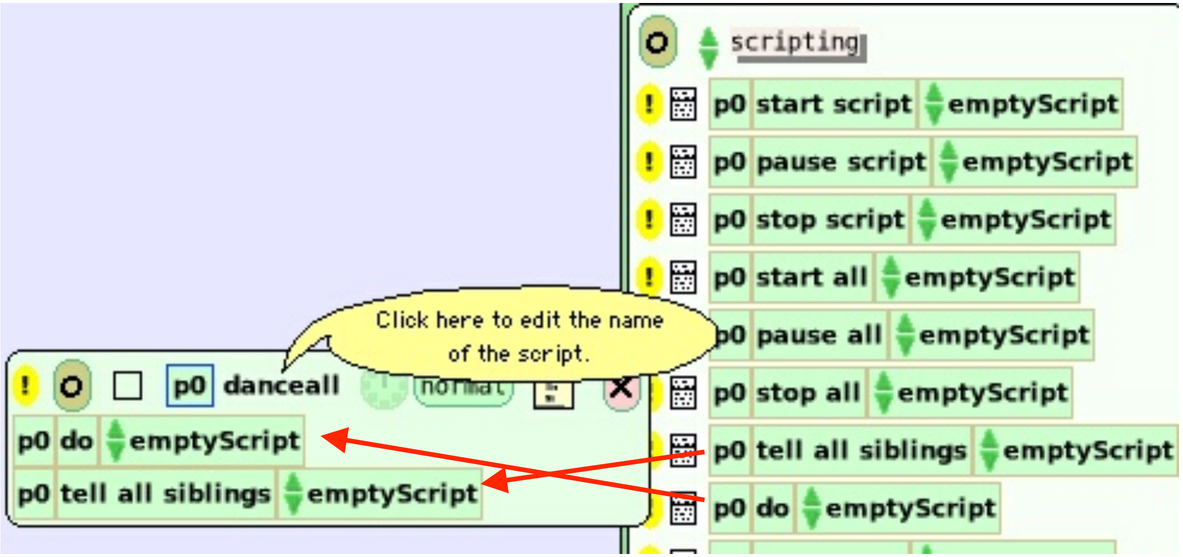
pervasivetechnologylabs
AT INDIANA UNIVERSITY

Flash movie: create a script



Create a script (a program) by dragging/ dropping tiles (i.e. hold/release mouse button)

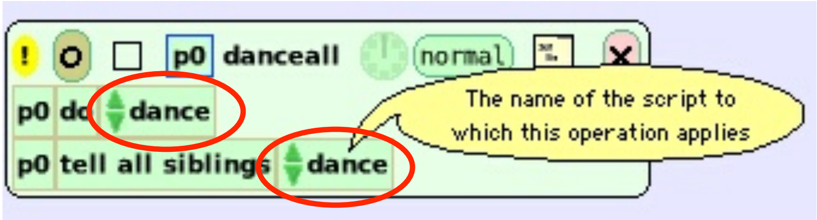
Make all computers dance



Click here to edit the name of the script.

```
p0 danceall normal
p0 do emptyScript
p0 tell all siblings emptyScript
```

scripting
p0 start script emptyScript
p0 pause script emptyScript
p0 stop script emptyScript
p0 start all emptyScript
p0 pause all emptyScript
p0 stop all emptyScript
p0 tell all siblings emptyScript
p0 do emptyScript



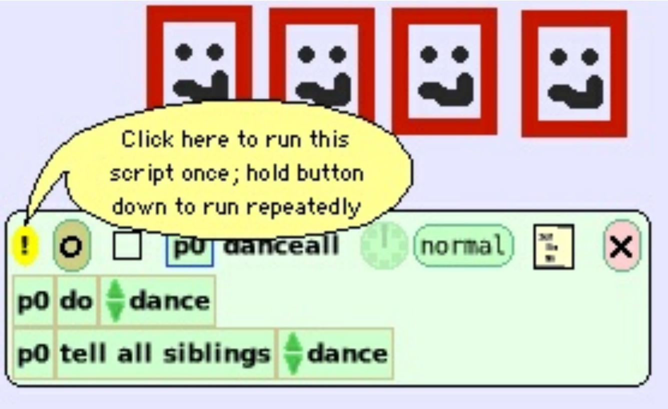
```
p0 danceall normal
p0 do dance
p0 tell all siblings dance
```

The name of the script to which this operation applies

Create another script, name it ‘danceall’ and have it invoke the ‘dance’ script for p0 and all its siblings.

pervasivetechnologylabs
AT INDIANA UNIVERSITY

Sync’ d line dancing

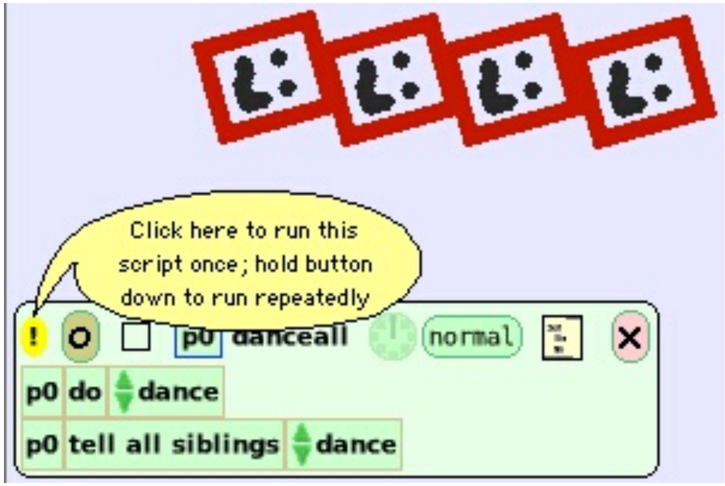


Click here to run this script once; hold button down to run repeatedly

p0 danceall

p0 do dance

p0 tell all siblings dance



Click here to run this script once; hold button down to run repeatedly

p0 danceall

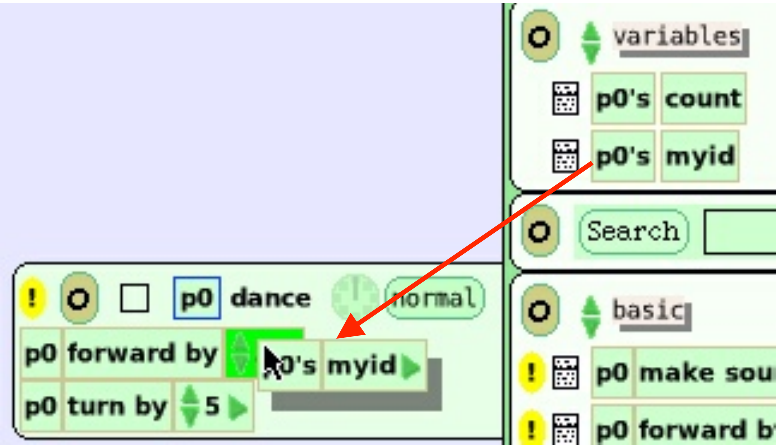
p0 do dance

p0 tell all siblings dance

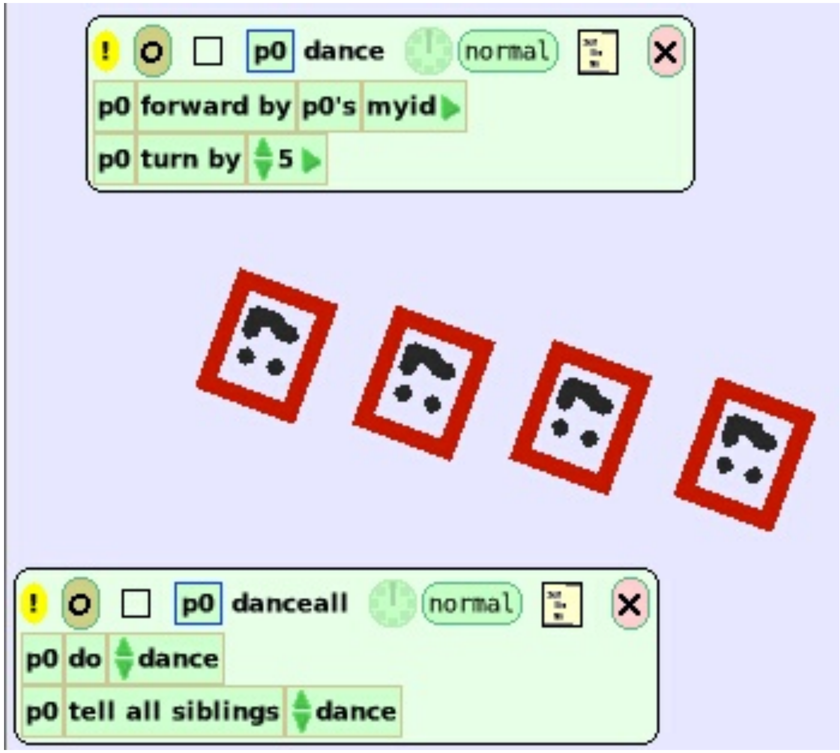
Running the ‘danceall’ script will cause all computers to dance the same circle dance.

Asynchronous dancing

Replacing the “forward by” fixed value with the variable ‘myid’ will make each computer have a unique dance.



```
p0 dance  normal
p0 forward by  p0's myid
p0 turn by 5
```



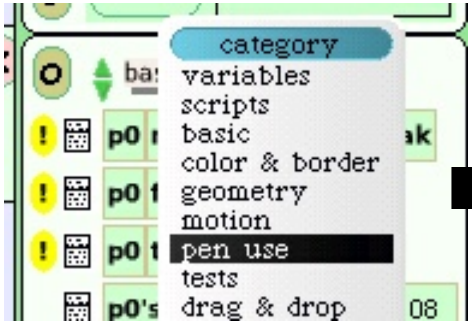
```
p0 dance  normal
p0 forward by p0's myid
p0 turn by 5
```

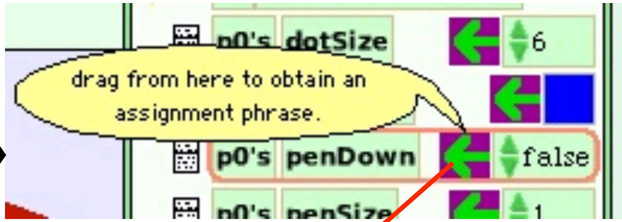
```
p0 danceall  normal
p0 do dance
p0 tell all siblings dance
```

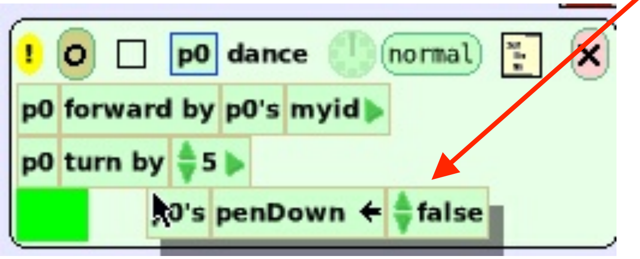
pervasivetechnologylabs
AT INDIANA UNIVERSITY

Draw my dance

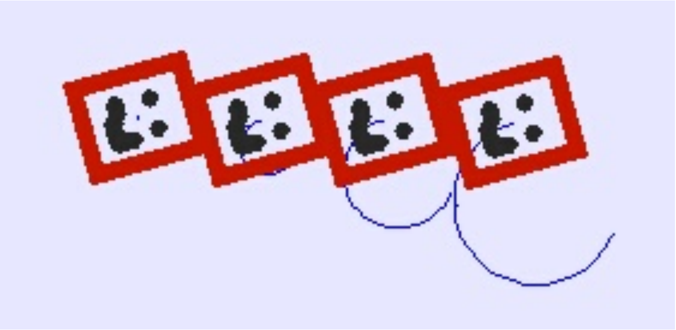
In the ‘pen use’ category, drag the ‘penDown’ tile into our script to draw our paths.

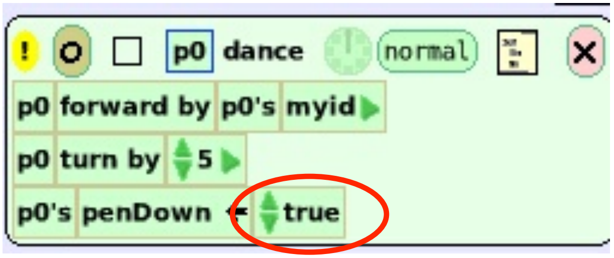






```
p0 dance normal
p0 forward by p0's myid
p0 turn by 5
p0's penDown ← false
```

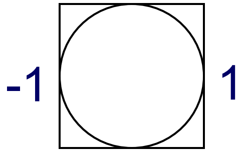




```
p0 dance normal
p0 forward by p0's myid
p0 turn by 5
p0's penDown ← true
```

Calculating the value of pi (π)

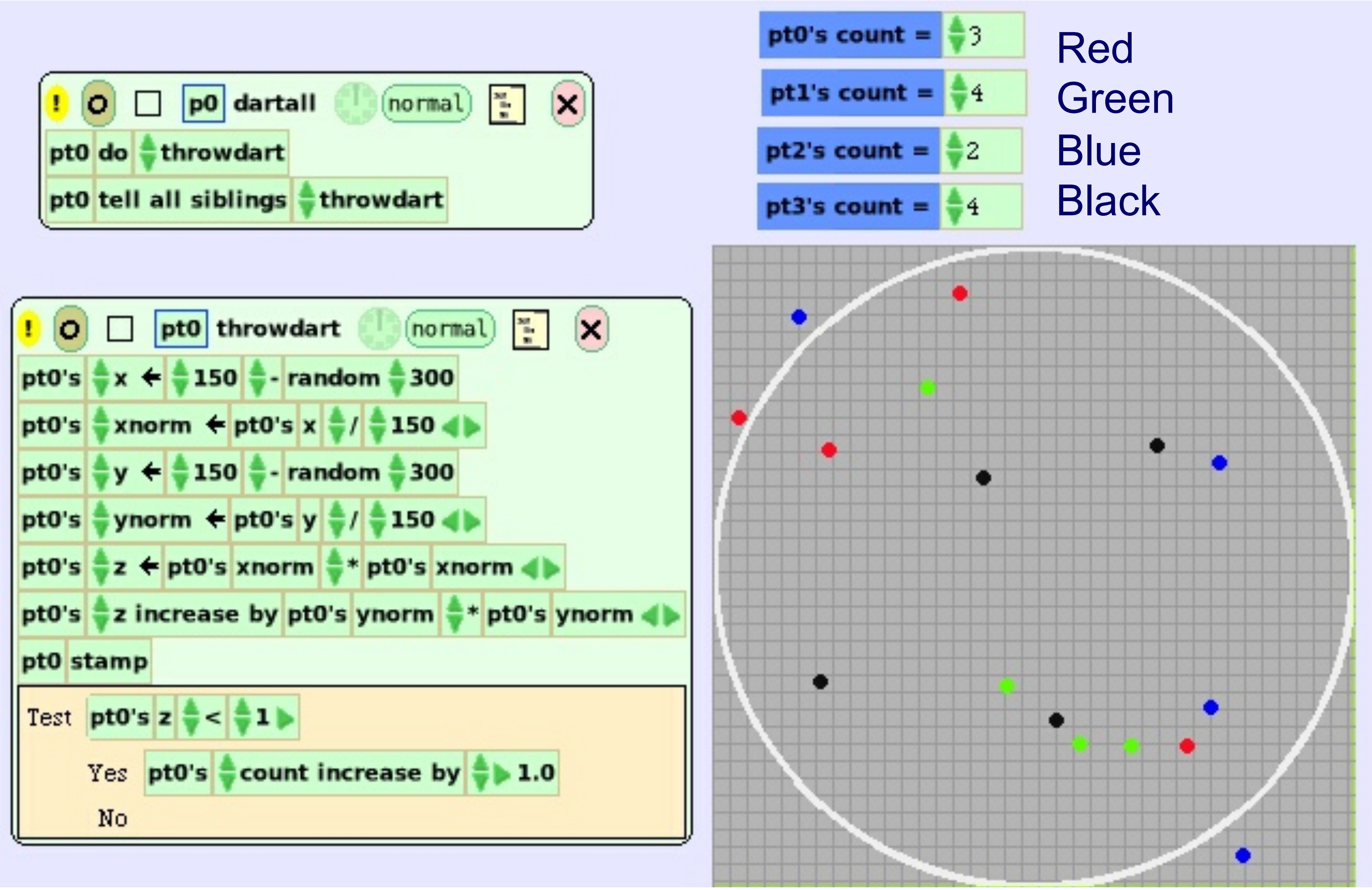
An introductory MPI program is to compute the value of pi using a Monte-Carlo approach.
Each processor:



- "throws darts" at a square board of unit radius
- if a dart lands inside the unit circle, increment a counter

Since the area of the square board is =4 (2x2) and the area inside the unit circle is $\pi r^2 = \pi$
Then we know that π/4~(#darts in circle)/(total #)
and we can approximate π

Each of 4 processors has a different colored dart – results after a few throws



```
pt0's count = 3
pt1's count = 4
pt2's count = 2
pt3's count = 4
```

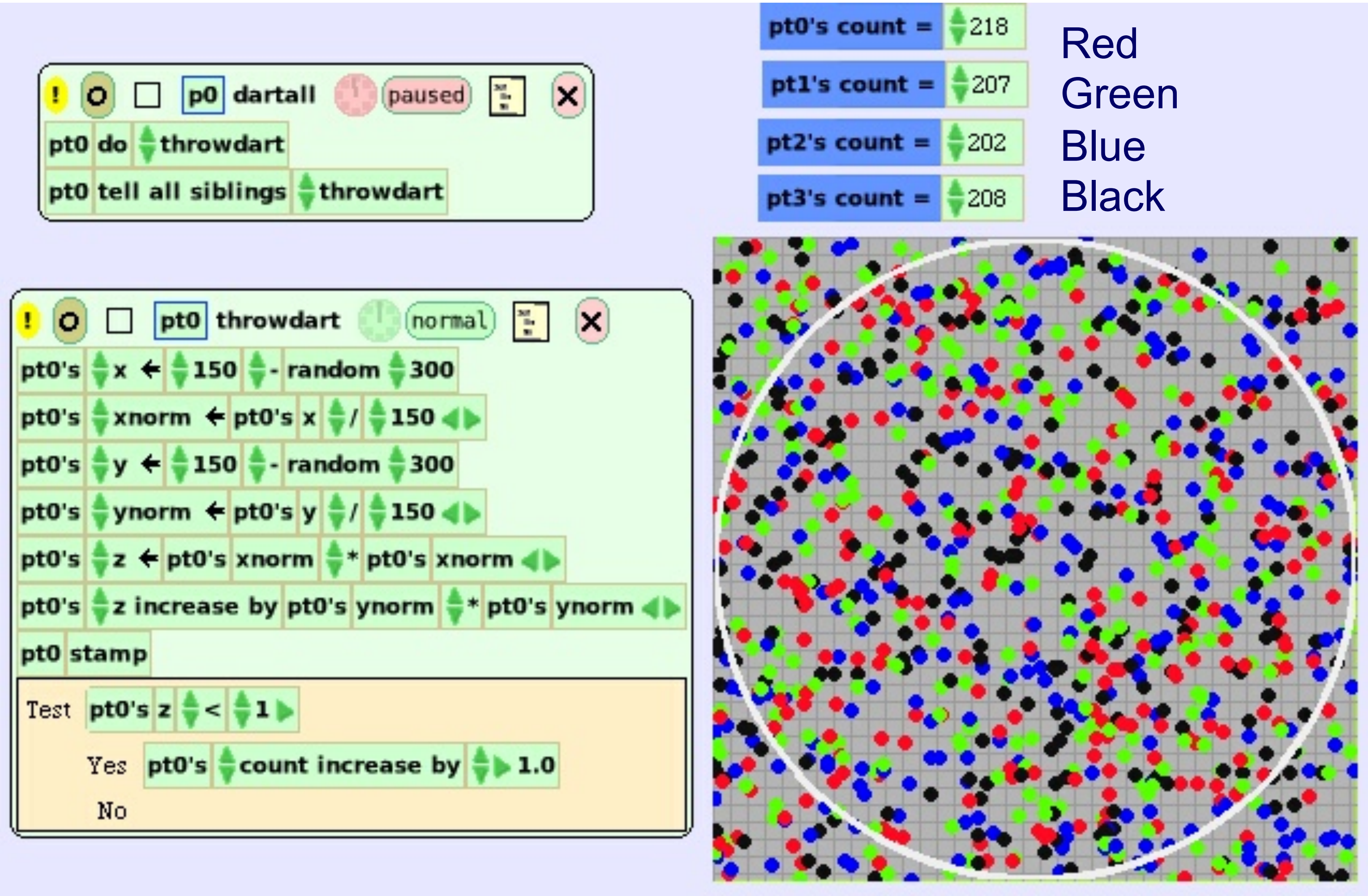
Red
Green
Blue
Black

```
! ○ □ p0 dartall normal ✕
pt0 do throwdart
pt0 tell all siblings throwdart
```

```
! ○ □ pt0 throwdart normal ✕
pt0's x ← 150 - random 300
pt0's xnorm ← pt0's x / 150
pt0's y ← 150 - random 300
pt0's ynorm ← pt0's y / 150
pt0's z ← pt0's xnorm * pt0's xnorm
pt0's z increase by pt0's ynorm * pt0's ynorm
pt0 stamp
Test pt0's z < 1
    Yes pt0's count increase by 1.0
    No
```

The pixel size of our dart board is 300x300; we normalize it to be [-1,1]

Results after several throws



```
! ○ □ p0 dartall paused ✕
pt0 do throwdart
pt0 tell all siblings throwdart
```

```
! ○ □ pt0 throwdart normal ✕
pt0's x ← 150 - random 300
pt0's xnorm ← pt0's x / 150
pt0's y ← 150 - random 300
pt0's ynorm ← pt0's y / 150
pt0's z ← pt0's xnorm * pt0's xnorm
pt0's z increase by pt0's ynorm * pt0's ynorm
pt0 stamp
Test pt0's z < 1
  Yes pt0's count increase by 1.0
  No
```

pt0's count =	218	Red
pt1's count =	207	Green
pt2's count =	202	Blue
pt3's count =	208	Black

The calculation of pi is left as an exercise…
(get the Squeak project from http://sda.iu.edu/K-12)

pervasivetechnologylabs
AT INDIANA UNIVERSITY

Thanks!

We gratefully acknowledge the developers of Squeak and the squeakland.org community of users.