

Parallel Programming with Pictures

Annette FENG, Yasmine BELGHITH, and Emma MANCHESTER

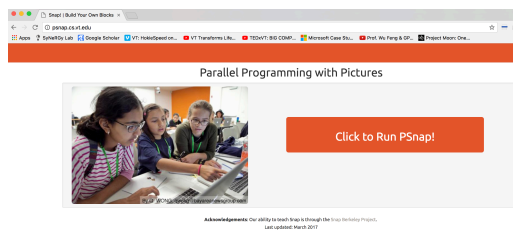
Principal Investigator: Wu FENG

Women in Computing Day

KnowledgeWorks II Bldg, 2202 Kraft Drive, Blacksburg, VA
March 23, 2018

Schedule

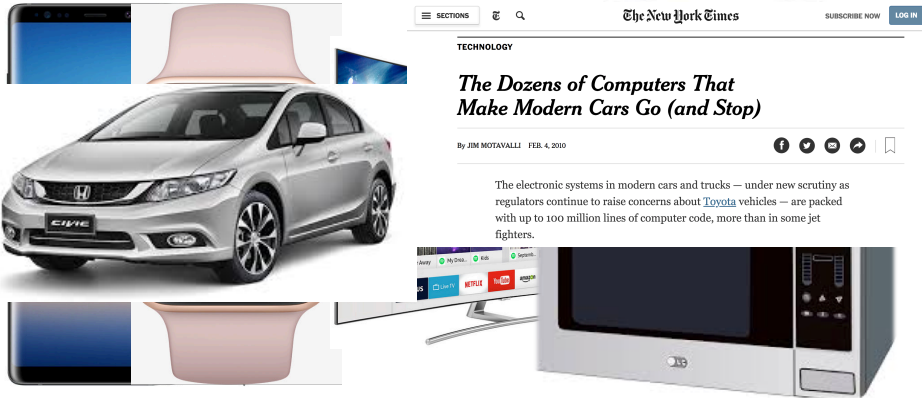
- PRE-SURVEY via the web browser on your laptop
- The power of computing



- POST-SURVEY via the web browser on your laptop

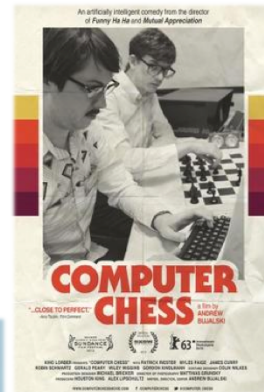
Importance of Computer Science

- How many computers does your family own?
 - More than you might think! Would you believe 60+?



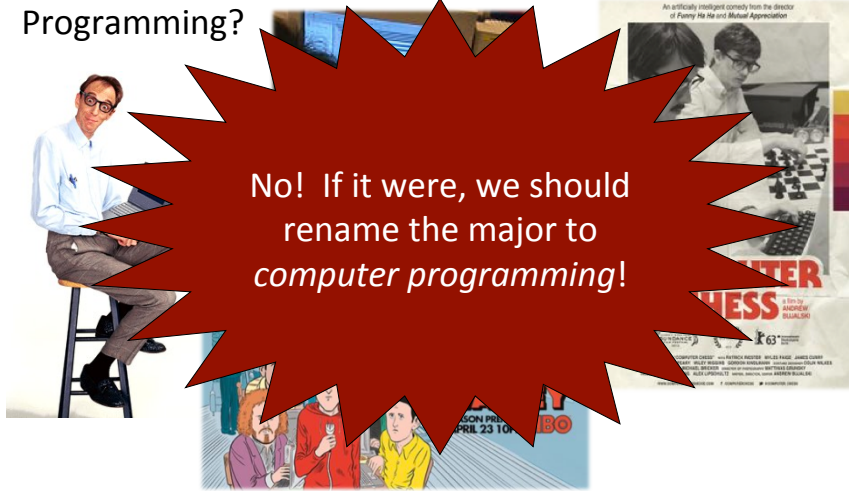
What is Computer Science?

- Programming?



What is Computer Science?

- Programming?



Importance of Computer Science

- What does computer science enable?
 - Solve important problems

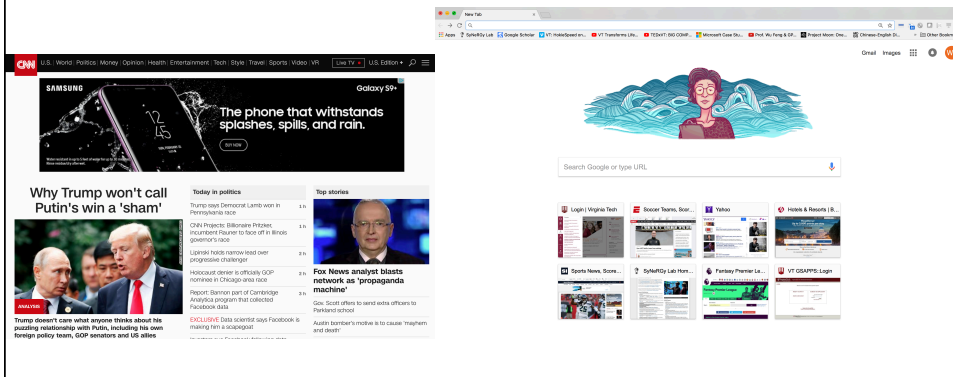
Importance of Computer Science

- What does computer science enable?
 - Solve important problems
 - Connect with people, e.g., at work or at play



Importance of Computer Science

- What does computer science enable?
 - Solve important problems
 - Connect with people, e.g., at work or at play
 - Collect and communicate information



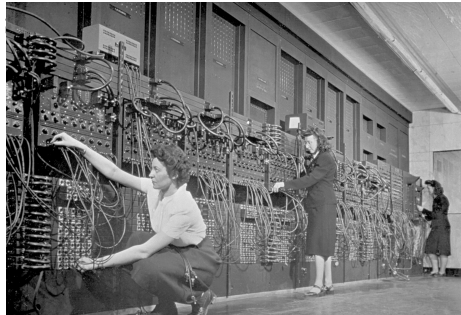
Importance of Computer Science

- What does computer science enable?
 - Solve important problems
 - Connect with people, e.g., at work or at play
 - Collect and communicate information
 - Create digital media & entertainment



The Computer Girls: 1967 Cosmo article highlights women in technology

by Elaine Burke 18 AUG 2015 6,183 VIEWS



Women computer operators program ENIAC, the first electronic digital computer, by plugging and unplugging cables and adjusting switches.



The mathematical brains behind the U.S. *first* launching of a human into outer space

Women are *Naturals* at Computing

“Women are ‘naturals’ at computer programming. It’s just like planning a dinner,” because it requires advance preparation, patience, and attention to detail.

- Quote by pioneering programmer Grace Hopper in the 1967 *Cosmopolitan* article ‘The Computer Girls’.



Overview

- Module 1: Creating a Serial Program
- “On Your Own” Bonus Modules
 - Module 2: Creating a Parallel Program
 - Module 3: Missile Command: Applying Parallelism to Gaming

Module 1: Creating a Serial Program

What is a Program?

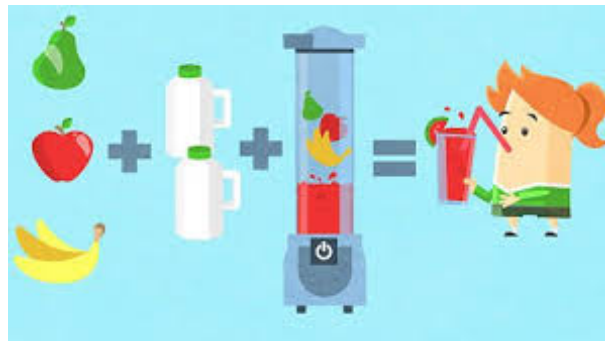
A program is an algorithm that runs on a computer.



What is an Algorithm?

An algorithm is a set of instructions that explains step by step how to do a task or solve a problem.

It's like a recipe:



Algorithm: PEMDAS

1. Parentheses
2. Exponents
3. Multiplication & Division
4. Addition & Subtraction

Solve:

$$4 + 5(3 - 1)^2$$

$$4 + 5(2)^2$$

$$4 + 5 * 4$$

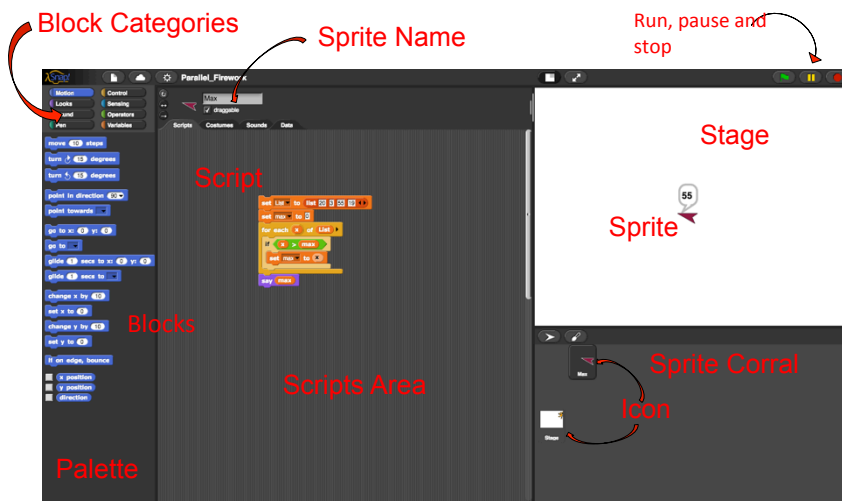
$$4 + 20$$

$$\boxed{24}$$

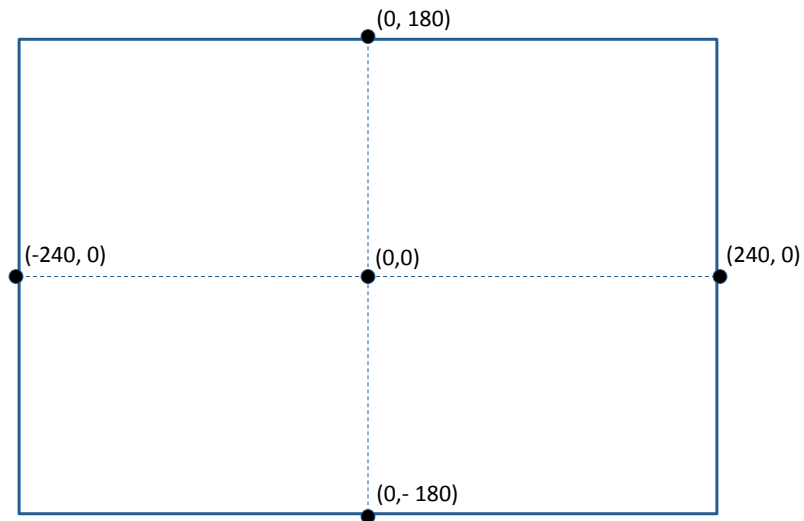
What You Will Learn

- Visual programming in Snap!
- How to do simple animation
- What are loops
- How to build a game

Introducing the Snap! Environment

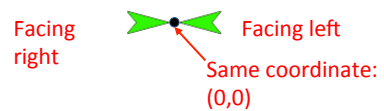


The Stage

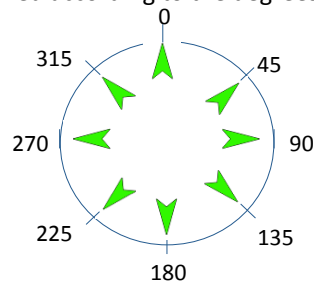


Sprite Orientation

- A sprite has both a direction in which it faces and its coordinate.
- By default, the coordinate point is set to be at the tip.
- When a sprite turns, it pivots about its coordinate:



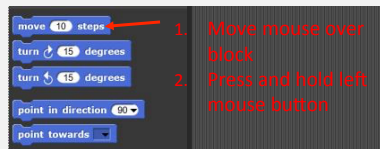
- Direction is specified according to the degrees of a circle:



Working with Blocks

Adding blocks to scripts area

Pick up and drag from palette:



1. Move mouse over block
2. Press and hold left mouse button

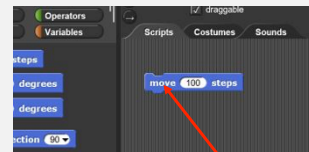
Drop into scripts area:



1. Slide mouse over to scripts area
2. Release left mouse button

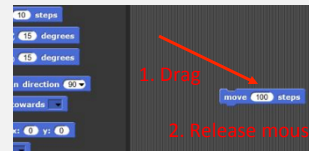
Moving blocks within scripts area

Pick up from scripts area:



1. Move mouse over block
2. Press and hold left mouse button

Drop in new location:



1. Drag
2. Release mouse

Working with Blocks

Connecting



1. Drag block toward bottom of block until a white line appears between the two blocks.
2. Release block and the two blocks will "snap" together.

Disconnecting

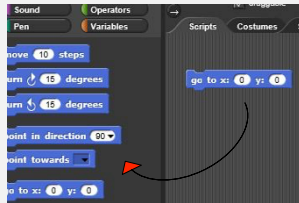


1. Grab bottom block and drag away from top block.
2. Place elsewhere in scripts area or discard on palette.

Working with Blocks

Deleting: Two Ways

1. Drag and drop back onto palette:



2. Right-click on block & select "delete":

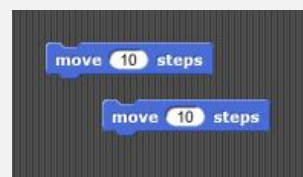


Duplicating


Right-click on block & select "duplicate":



Result:

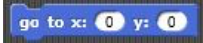


Moving Sprites

- Add a **move 100 steps** block to your scripts area and click on it to run it.
- Add a **point in direction 90** block to your scripts area.
- What happens when you click on it?
- Change direction using the drop-down menu of the **point in direction 90** block and then click on the block to run it.
 
 A screenshot of the 'point in direction 90' block with a context menu open. The menu options are: '(90) right', '(-90) left', '(0) up', and '(180) down'.
- Click again on the **move 100 steps** block and see how the sprites moves in the new direction it's facing.
- Change the inputs to the **move 100 steps** and **point in direction 90** blocks and see what happens when you run them. Try negative numbers.

Retrieving Lost Sprites

If you “lose” your sprite off the stage, try these options:

1. Execute a  to bring it back to center stage.
2. Right-click on the sprite icon located underneath the stage.

Select “show” from
the pop-up menu

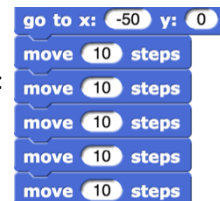


Moving Sprites: Intro to Animation

We want to move our sprite across the stage from left to right.

Start with a  block.

Add five  blocks to the script to get:



What happens when you run your script?

The sprite seems to move all 50 steps at once.

The computer executes so fast you miss all the moves in between.

How can we write the script so the sprite appears to glide across the stage?

Answer: 

Controlling Animation

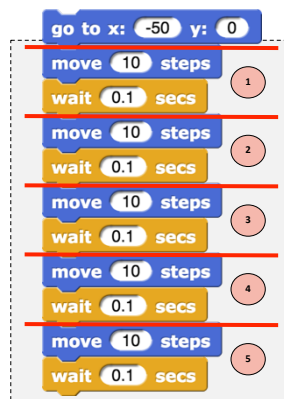
To slow animation enough to see it happen, pause between movements.

Insert wait blocks

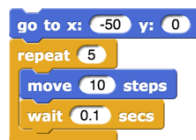


Loops: Identifying Patterns

The same "move and wait" sequence is repeated **5 times** and requires **10 blocks**.



We can write the script more efficiently using a  **repeat (5)** block:



Using the **repeat block**, write the same sequence using only **3 blocks**.

Gliding Across the Stage

- We want to make our sprite glide across the stage from left to right.
- The sprite starts off stage left at **(-275, 0)** and ends off stage right at **(275, 0)**.
- The sprite moves a total of **550** steps: $275 - (-275) = 275 + 275 = 550$.
- Suppose we want to move in steps of 10. That means the sprite has to move **550/10 = 55** times.
- For starters, choose a wait time of 0.05 secs.



- If you adjust the number of steps to something like 25, you'll have to change the number of times you repeat: $550/25=22$.

Looping Forever

How do we make our sprite go back to the beginning and glide across the stage over and over again?

Put your script inside a **forever** block.

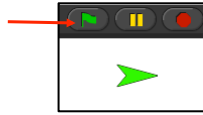


Your script should look something like this:



Starting Scripts

Click the start button above the top-right of the stage



What happens?

Add **when clicked** to the top of your script:



What happens when you click start now?

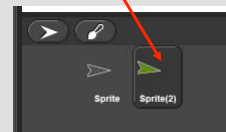
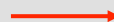
To stop your script click on the stop button

Adding New Sprites

To add a new Sprite:

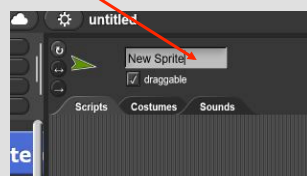


New sprite icon appears



To rename a Sprite:

Write here



Sprites and Scripts:

Each sprite has its own script

To edit the scripts for a sprite, click on that sprite's icon in the sprite corral and it's scripts will appear in the scripts area.

Other Ways to Move Sprites

Add these four scripts to your **new** sprite:

when **up arrow** key pressed

change y by 10

when **right arrow** key pressed

change x by 10

when **down arrow** key pressed

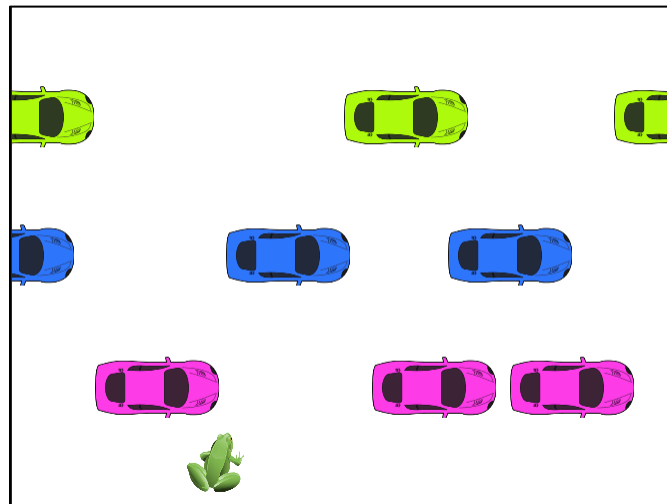
change y by -10

when **left arrow** key pressed

change x by -10

Now you can move your sprite around the stage using the arrow keys.

Frogger



Costumes

Set up the car sprite:

1. Under File, select **Costumes...**

2. Choose the color car you want from these

3. Rename the sprite **Car**

Set up the frog sprite:

1. Select the **frog-sitting.png** costume
2. Rename the sprite **Frogger**
3. If it isn't already, get Frogger to face right by clicking on a **point in direction 90**

Collision Detection

The game should end whenever the car and Frogger touch.

After each time the car **move 10 steps** it must then check if it hit Frogger.

If they're touching, then stop the program.

In the car's scripts area, create a new script using the blocks shown here:

Insert it into the 'glide' car script as shown:

Initializing Sprites

- When you start the game you want the sprites to begin at their initial spots
- The car is already set up: it starts off the left-hand side of the stage
- We'll have Frogger begin at the bottom of the stage and in the middle whenever the start button is pressed:

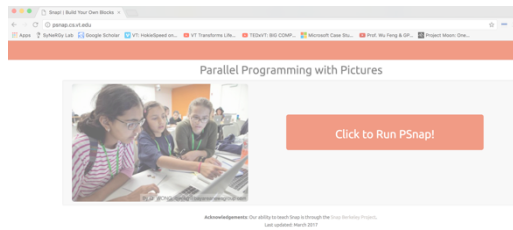


Exercises

- Add more cars.
 - different rows: change the Y-coordinate
 - same rows: delay starting each car by different times
- Make Frogger appear to jump by using costume changes.
- Change the background by editing the stage costume.
- Keep score (add points as you successfully jump higher).
- Add more lives.
- Change the stage to show "Game Over" or "You Win!".
- Save your programs.
- A full version of the game can be found under File->Open...->Examples->Frogger.

Schedule

- PRE-SURVEY via the web browser on your laptop
- The power of computing



- **POST-SURVEY via the web browser on your laptop**