

UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 202 Introduction to Computation Professor Andrea Arpaci-Dusseau
Fall 2010

Lecture 26: How does a computer... prevent race conditions?

Today's Scenario

Imagine: You've written a great Scratch program

- Lots of interacting Sprites, variables
- Most of the time it works like you expect...


But, sometimes... Funny results

- Sprites disappear/reappear in unexpected ways
- Points don't increment
- Hard to be sure -- is something wrong or not?

Answer Today

- What is going wrong?
- How can we fix this type of problem?

Easiest Possible Game



User controls cat with arrow keys

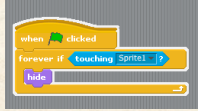
Cat picks up 6 objects for points

Game over when pick up all 6 objects

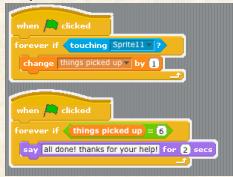
How might one implement this?

Problem: Asking Same Question Twice

Sprite 11 (baseball)



Sprite 1 (cat)



Why won't this code always work?

If ball sees "touching Sprite1" and hides first, Sprite 1 won't see "touching ball", won't increment!

Problem:
Two Sprites ask same question, and get different answers!

Solution?

Only one Sprite asks question; how?

- Ball could inc variable
- Broadcast answer (if others need to know)

Why does this happen?

Concurrency in Scratch

- Every **script stack** executes concurrently (appears simultaneous) with all others

Concurrency usually good thing:

- Can do many things at "same" time!
- Multiple Sprites can be moving at same time
- Play music in background
- Multiple Sprites can be checking different conditions
 - If key pressed
 - If touching another Sprite

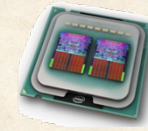
Many Concurrent Environments

Multiprogramming on single processor:

- Context switch quickly between active processes: Time sharing
- Application view: Context switches can happen at any time!

Parallel Systems

- Multiprocessors
- Distributed systems
- Multiple processes running at **same time**
- Can greatly improve performance



Problem of Concurrency: Race Conditions!

Race condition: Ordering of instructions across scripts impacts results

- Ordering: How scripts are scheduled

Results: Sometimes get result A, sometimes get result B...

Problematic when multiple scripts access **shared state**

- Access + modify what appears on stage (touching vs. hiding)
- Access + modify same variables

Second Example: Monkey Game

Many things happening concurrently!

- Multiple bananas falling from tree
- Thief monkey moving
- User moves monkey with keys
 - Up and l/r simultaneously
 - More efficient way to move with keys



More Efficient Movement

Jump: Monkey moves up, waits, moves back down

```

when up arrow key pressed
repeat 10
  change y by 10
wait 0.25 secs
repeat 10
  change y by -10
  
```

Left right movement: Lets user hold down keys

```

when clicked
go to x: 48 y: -83
show
forever
  if key left arrow pressed?
    point in direction -90
    move 5 steps
    next costume
  if key right arrow pressed?
    point in direction 90
    move 5 steps
    next costume
  
```

Avoiding Race Conditions

Banana Scripts

```

when clicked
show
go to x: pick random 150 to 95 y: pick random 180 to 60
forever
  change y by pick random 0 to 4
  if touching monkey?
    change Bananas by 1
  go to x: pick random 150 to 95 y: pick random 180 to 60
  if touching edge?
    go to x: pick random 150 to 95 y: pick random 180 to 60
  
```

Only Banana Sprite asks question "touching"

- Increments shared variable
- Goto new position

Monkey does not ask same question

- Monkey doesn't need to know answer

Avoiding Race Conditions

Thief Script

```

when clicked
show
go to x: 163 y: -97
point in direction 90
forever
  move 1 steps
  if on edge, bounce
  if touching monkey?
    change Stolen Bananas by Bananas
    set Bananas to 0
    say Thanks! for 2 secs
    broadcast Sad Monkey
  
```

Monkey Script

```

when I receive Sad Monkey
say Oh no! for 2 secs
  
```

New Situation: Two Sprites need answer

Actions when Thief and Monkey meet

- Change Banana count
- Thief says Thanks
- Monkey says "Oh no!"

How to avoid Race condition?

- Only one sprite asks questions
- Broadcast message to other

Scripts for Simplified Bug on a Plate very similar (check out code!)

How is Concurrency Implemented in Scratch?

How does Scratch environment pick block to run next?

Repeat until all blocks completed

Run "few" commands from each *stack*

(Remember last position in each stack)


Update screen

Order of stacks is unknown!

- Don't know which stack will be first or next
- Could pick different stack each time
- Cannot assume any order across stacks!
- May differ from run to run, across versions, machines, web version...

Example: Concurrent Initialization

Multiple stacks initialize same variable (test)




**What will Sprite say?
What will be final value of test?**

Test could be:
0, 1, 2, 3, or 4!

Conclusion:
Cannot make any assumption about stack ordering

Example: How many Meows?

Confused Cat Scripts




How many meows?
Could be 0, 1, or 5!

How to ensure initialize correctly? (assume want test = 5 before repeat loop)

Must control order blocks are executed

Easiest Fix: Remove Concurrency

Single script does everything No concurrency within a script



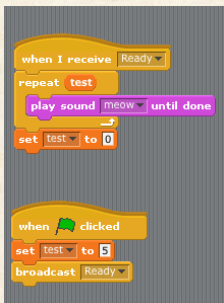
Blocks in single script execute in order

Guaranteed to initialize variables before entering repeat loop

Doesn't work if multiple initial scripts use "test" variable

General Solution: Control Order of Scripts

Correct Initialization



Use broadcast/recieve

When Green Flag Clicked

- Perform initialization of variables
- Broadcast Ready

When Receive Ready

- Guaranteed everything initialized correctly
- Ready to Go!

How do we reason about Concurrency?

Problem:

Difficult to build programs when no assumptions about switches between stacks

Solution:

Atomic operation: Will not be interrupted in the middle

What happens if not atomic and switch between two related instructions?

- State of world could change



Need to cross intersection: wait until no cars

Look to right: no cars

Look to left: no cars

Decide to drive across road

Accident!

What happened?

Something changed between when you checked and when you started to drive



Another Example

Problem:

Difficult to build programs when no assumptions about switches between stacks

Solution:

Atomic operation: Will not be interrupted in the middle

What happens if not atomic and switch between two related instructions?

- State of world could change



Need to sit down on a chair

Look to behind you: there's a chair

Decide to sit down

Embarrassing fall on floor!

What happened?

Something changed between when you checked and when you started to act

What is Atomic in Scratch?

Scratch: Each command block executes atomically except:

Blocks that wait

- Specified amount of time
 - Examples: "wait," "glide", "say"
- For something to finish
 - Examples: "play sound and wait", "broadcast and wait"

When encounter waiting block, check condition

- If not done, Scratch continues to next stack
- If done, Scratch goes to next block after wait block

Are Multiple Blocks in Same Script Atomic?

Scratch executes some number of blocks in each stack before moving to next stack

How many blocks does Scratch run in each stack?

Scratch runs all blocks in one stack until

- Reach waiting block
- Reach end of stack
- Reach end of innermost loop

Example: move, next costume, turn: Atomic

Adding Unique Items to a List

- What is code trying to do?
- Only add items to Unique List if not already there
- Will this code work?
- Yes! Why?
 - Each checks if item in list; if not, adds it
- Critical section: instructions that must be executed without interruption
- What is critical section here?
- What is shared variable?
 - Unique List
 - Two blocks:
 - if not Unique List contains x
 - Add x to Unique List
 - If no interruptions, works fine!

Adding Unique Items to List: With an Interruption!

- Why won't this code work?
- Critical section no longer guaranteed to be atomic!
- Will schedule other script when each calls "say"

Adding Unique Items to List: With an Interruption!

- Why won't this code work?
- Critical section no longer guaranteed to be atomic!
- Will schedule other script when each calls "wait"

Today's Summary

- Concurrency: Entities appear to run simultaneously
- Scratch: Concurrency occurs across **Script Stacks**
 - Unknown ordering across stacks
 - Challenge: Avoid **Race Conditions** (unpredictable results) when switch between scripts
 - Switches between scripts after inner loop or waiting blocks
- No Reading
- Announcements
- HW 6 due Friday before class
 - Last chance to finish Project 1 demos