Lecture 22:
How does the computer help you with Project 2?

Project 2

Due 2 weeks from today: In-class demos
- See web page for lots more details and implementation hints

Objective
- Create points-based game
  - Educational (ask user questions; type or click answer)
  - Action (avoid objects, pick up objects)

Requirements
- **Points**: Track with a Variable and display
- **Instructions**: Tell user how to play
- **Assessment**: At end, give message to user depending on points
- **High Score List**: Display sorted list of 10 High Scores; tell user if they achieved high score. (This will take some care.)
- **Multiple Levels**: At least 3 levels of difficulty. Higher levels might involve more difficult questions, make objects move more quickly, have less time, or have more objects to avoid.
  - Save this for last!
  - Goal: No changes in code; Just variables!
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

Why won’t this code always work?

If Sprite 11 sees “touching Sprite1” and hides first, Sprite 1 won’t see “touching Sprite 11”, won’t increment!

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

Solution: Only one asks question
- Sprite 11 could inc variable
- Broadcast answer (if others need to know)
Why does this happen?

Concurrency

- 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!

Scenario:
Shared state across multiple scripts
  • Access + modify what appears on stage (touching vs. hiding)
  • Access + modify same variables

Race condition: Ordering of instructions across scripts impacts results
  • Ordering: How scripts are scheduled
Results: Sometimes get result A, sometimes get result B...

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

Left right movement: Lets user hold down keys

Avoiding Race Conditions

Banana Scripts

Previous Lesson

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

**New Situation:**
**Two Sprites need answer**

Actions when Thief and Monkey meet
- Change Banana count
- Thief says Thanks
- Monkey says “Oh no!”

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?

Scheduling implementation in Scratch:
Repeat until all stack scripts done
Run few commands from each stack
  (Remember last position in each stack)
Update screen

Order of picking 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 be the output?
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 scripts use “test” variable

General Solution: Control Order of Scripts

Correct Initialization

Use broadcast/receive

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

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" and "play note"
- 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

Multiple Blocks in Same Script Executed Atomically?

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
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

Short lesson for Project 2:
• Only one sprite should ask same question
• Control initialization of variables before others run

Concurrency: Across Scripts in Scratch
• Challenge: Avoid Race Conditions when switch between scripts
• Scratch: Unknown ordering across scripts
• Switches between scripts after inner loop or waiting blocks

Announcements
• HW 6 due today
• Project 2: Due Friday: Two weeks from Today
• Exam 1 Returned: Ave: 85, Median: 90