January 16, 2017

Administrative notes

- Reminder: optional course survey due tomorrow: https://survey.ubc.ca/s/cpsc-100-student-experience-pre-2017w2
- Add drop deadline is also tomorrow
- Reminder: In the News individual call #1 due tomorrow
  - On Thursday we’ll start the group portion in class (you’ll need a group)
- Reminder: Two reading quizzes due tomorrow (the 20th)
- Midterm #1 February 1 in class
There’s a lot to keep track of!

Let’s start by listing all of the things that we need to keep track of, e.g., how many points the player’s hand is worth

In a group, discuss some things we need to keep track of

- Sum of the player’s cards
  - Stop the game if it goes over 21
- J = 10
  - Q = 10
  - K = 10
- Player’s amount of money left
- Bet placed per round
Most things that we need to keep track of, we track with **variables** (named quantities)

Variable: where on the $x$ axis is the mouse?  
Variable: where on the $y$ axis is the mouse?
Variables have values

Value of mouse x

Value of mouse y

Computational Thinking
http://www.ugrad.cs.ubc.ca/~cs100
We call our screen our “stage”. “Things” we add are called sprites.

A sprite is an object you can move on a larger scene.
When we sorted cards, each slot was a variable

Unsorted

Simple sort

Sorted

So were the markers and dividers
We can use variables in our code

- Some variables are built in (e.g., “answer” is the answer to a question in Snap)
- You can make your own variables

Variables are in orange. Things that are black writing on white are constants – actually that value
Here are some variables in our Blackjack code

- BankRoll
- Bet
- DealerHand
- PlayerHand
- CardValue
Clicker question

What is the value of “my_variable” after the following code is run?

A. 6
B. 9
C. 42
D. 54
E. Other

The code visualized shows the operation: `set my_variable to 6 * 9`, which results in `54`. Therefore, the correct answer is D. 54.
Order!

- Programs will execute exactly in the order that’s given
- So if we assign values to variables, they’ll set one value after another after another
Clicker question

What is the value of “Max CP” after all lines of this program are executed?

A. 0
B. 2
C. 42
D. None of the above
Clicker question

What is the value of “shoe size” after all lines of the program are executed?

A. 2
B. 39
C. 40
D. Other
Changing myself

What is the value of “age” after all lines of this program are executed?

A. 1
B. 40
C. 41
D. Other
Snap! Makes it easy to show non-text things

For example we might need pictures of cards and a dealer

In your group, list non-text things that we might need to show

Things in the app:
- Stand button
- Hit button
- Yes button (to start a new name)
- No button (to end the game)
Here are some sprites, but with no code. What happens if we click on them?
Now where we...
We had these sprites, but no code
Sprites send information to each other through something called “broadcast”

For example, the “No new hand button” doesn’t have much to do other than tell sprites that it’s been pressed:

Note: Even if you have programed, this may be new to you
For example, if we wanted, we could create a new broadcast message called “boogie”

- Then we’d need to both broadcast it and do something when it is received.
- Let’s broadcast the message if someone clicks on the dealer

  ![Broadcast Message](image)

  ![Process Broadcast](image)

- We also need to process the broadcast

- It doesn’t matter if you create the code to broadcast or receive the message first. But the first time you use it, you have to define it
Now we can program everything we need for the hit button
But what should the sprites do? Let’s go back to algorithms

• At this point, we can program simple algorithms like the hit button
• The dealer’s a little harder
• In your group
  • Create an algorithm for what the dealer sprite should do after dealing the second card to the player
  • It should include things like what to do if the player has 21 points (in this case there’s no point in asking for a hint)
  • Assume that you have access to broadcast methods for
    • Showing the hit button (and what follows)
    • Showing the stand button (and what follows)
    • Standing
Algorithms for what to do after the second card is dealt to the player

Show hit or stand buttons and wait for player to choose

If the player chooses to hit, add the value of the card to the total the player has

If it is below 21 wait for the player to hit or stand

If it is over 21, show losing message (the cat will say something like “sorry you lost”)
Question:

When trying to decide to ask for a hit or not after the second card is dealt, do you need to check for the case where the player has more than 21 points?

A. Yes
B. No
Solving this one requires conditionals

- Conditionals let us do different things in different cases
- Generally this looks like “if” with a possible “else”:

  Real world example:
  
  “**If** you eat your dinner
   Then you may have Nanaimo bars
   **Else**
   You may only have fruit”
Now we can write what should happen when you deal the second card to the player.

```plaintext
when I receive DealtSecondCardToPlayer
wait 1 secs
if PlayerHandTotal1 < 21
    broadcast ShowStand
    broadcast ShowHit
else
    say You have 21 points! You win unless the dealer gets 21.
    for 2 secs
    broadcast PlayerStand
```
What happens when this program is run?

A. It says “Yay math!” only
B. It says “Gnarly” only
C. It says “Yay works!
   Then it says “Gnarly”
   Then it says “Dude, program’s over.”
D. It says “Gnarly”
   Then it says “Dude, program’s over”
E. None of the above
Or we can write this program

What does the program say if you press the “P” key?

A. “A is for alpha”
B. “B is for beta”
C. “I don’t know any other Greek letters”
D. Other
Okay, great. Now let’s figure out what the dealer needs to do when the player stands

• In your group, write an algorithm for what the dealer should do when the player stands
• Assume you have access to broadcasting things like
  • Player Lost
  • Player Wins
  • Deal one card to the dealer
Algorithms for the dealer after the player stands

If the dealer has less than 16, then deal one card to itself.

The dealer will do the same thing again and again until it reaches 16 or it busts (goes over 21).
First, let’s flip the hole card and deal until we have 17 points.
Sometimes you want to do the same over and over again

A loop allows you to do the same thing over and over again, sometimes with a stopping condition, sometimes forever.
Now, let’s see who won

Group exercise: how do we check to see who won?

If the dealer > 21 points, then player wins

else if dealer < player, player wins

else player loses (because it would be a tie and we have already specified that in the event of a tie, the dealer will win)
Now we can write the rest of the code for when the player stands:

- **when I receive** `PlayerStand`
  - **broadcast** `FlipHoleCard` and wait

**repeat until** `DealerHandTotal` > 16
- **say** `Dealer takes another card for 2 secs`
- **broadcast** `DealOneToDealer` and wait

**if** `DealerHandTotal` > 21
- **broadcast** `PlayerWins`

**else**
- **if** `PlayerHandTotal1` > `DealerHandTotal`
  - **broadcast** `PlayerWins`
- **else**
  - **broadcast** `PlayerLost`
Clicker question:
Loop de loop. What does this code do?

A. It will say “times through” seven times
B. It will say “r” seven times
C. It will say each letter in the word “respect”
D. None of the above
Last loop clicker question

Will this program ever say “I still haven’t found what I’m looking for”?

A. Yes
B. No
That’s basically what you need to know about programming or this class

• We’ve hidden a lot of details. But those are the basics.
• Programming is a lot easier to learn by *doing* than by *seeing*, so you’ll do some more in lab.
Preparation for In The News groupwork in class on Thursday

- Make sure you submit your In The News article and comments by the deadline Wednesday
- Look over the articles submitted and come ready to plan to discuss an article
- You may want to form a group – look for the discussions to
  - Find group members
  - Register your group – you must do this in order to submit for credit!
Learning Goals Revisited

• [CT Building Block] Explain what a variable is in computer programming.

• [CT Building Block] Be able to trace through code using sequences of instructions, variables, loops, and conditional statements in short programs specified in a visual programming language such as Snap, or in other clearly expressed processes (which may or may not be computer related)

Note: Look carefully: it says be able to trace code, not write code. In order to help you do this, you will write a small amount of code in lab. You will not, however, be asked to write code on exam.