COSC 111:
Computer Programming I

Dr. Bowen Hui

University of British Columbia
Okanagan

Learning Outcomes

- Students will be able to …
  - Develop an appreciation for the complexity in creating computer software

Learning Outcomes

- Students will be able to …
  - Develop an appreciation for the complexity in creating computer software
  - Build simple programs on your own from scratch

Learning Outcomes

- Students will be able to …
  - Develop an appreciation for the complexity in creating computer software
  - Build simple programs on your own from scratch
  - Contribute to existing programs given to you

Learning Outcomes

- Students will be able to …
  - Develop an appreciation for the complexity in creating computer software
  - Build simple programs on your own from scratch
  - Contribute to existing programs given to you
  - Find relevant resources to help trouble shoot programming problems

Learning Outcomes

- Students will be able to …
  - Develop an appreciation for the complexity in creating computer software
  - Build simple programs on your own from scratch
  - Contribute to existing programs given to you
  - Find relevant resources to help trouble shoot programming problems
  - Work with others to come up with solutions

Reflection on Assignments, Labs, In-class Activities

- Reading code
  - Comparing code with English class designs
- Adding to existing code
  - Changing small pieces
  - Adding your own methods
  - Changing lots of code in the Feed Me/Treasure Hunt games
- Writing your own classes
  - Game with various Math activities
  - Text messaging system
  - Library online catalogue

Recall the Robot Example

```
* map:
* - kitchen (K) - tea
* - door (D) - newspaper
* - office (O) - where the master is
* - library (L)
* - hallway (H)
* O - K -L
*     |              NORTH is up
*     H - D
*
```

- Robot moves around to get tea or pick up newspaper for its master

Recall the Robot Example (cont.)

goes to office
to service
its master

```
********** Testing Robot ...
loc: library
move: library, south, library
move: library, north, library
move: library, south, library
move: library, west, kitchen
move: kitchen, west, office
>> Please select a command:
>> 1. Pick up the newspaper
>> 2. Make me some tea
>> 3. Return to your own work
1
```

Recall the Robot Example (cont).

picks up newspaper, delivers it to the master

```
move: office, east, kitchen
move: kitchen, south, hallway
move: hallway, south, hallway
move: hallway, east, door
pick up newspaper
move: door, east, door
move: door, south, door
move: door, north, door
move: door, west, hallway
move: hallway, south, hallway
move: hallway, east, door
move: door, east, door
move: door, west, hallway
move: hallway, north, kitchen
move: kitchen, north, kitchen
move: kitchen, north, kitchen
move: kitchen, east, library
move: library, south, library
move: library, south, library
move: library, west, kitchen
move: kitchen, west, office
here's your newspaper
```

Recall the Robot Example (cont).

goes back to the library (not very efficiently)

```
move: office, east, kitchen
move: kitchen, west, office
move: office, west, office
move: office, west, office
move: office, east, kitchen
move: kitchen, north, kitchen
move: kitchen, south, hallway
move: hallway, south, hallway
move: hallway, west, hallway
move: hallway, west, hallway
move: hallway, south, hallway
move: hallway, east, door
move: door, south, door
move: door, north, door
move: door, north, door
move: door, west, hallway
move: hallway, north, kitchen
move: kitchen, west, office
move: office, north, office
move: office, south, office
move: office, west, office
move: office, east, kitchen
move: kitchen, east, library
back to cleaning the shelves
loc: library
```

Major Pieces

```
* map:
* - kitchen (K) - tea
* - door (D) - newspaper
* - office (O) - where the master is
* - library (L)
* - hallway (H)
* O - K -L
*     |             NORTH is up
*     H - D
*
```

- Has a map and some locations
- Moves are random directions
- Keeps trying to get to destination until success

you know how to do all this!

Handling Possible Moves

```
private void possibleMoves( int dir )
{
  if( currLoc == OFFICE && dir == EAST )          // O, EAST, K
    currLoc = KITCHEN;
  else if( currLoc == KITCHEN )
  {
    if( dir == WEST )                             // K, WEST, O
      currLoc = OFFICE;
    else if( dir == EAST )                        // K, EAST, L
      currLoc = LIBRARY;
    else if( dir == SOUTH )                       // K, SOUTH, H
      currLoc = HALLWAY;
  }
  else if( currLoc == LIBRARY && dir == WEST )    // L, WEST, K
    currLoc = KITCHEN;
  else if( currLoc == HALLWAY )
  {
    if( dir == NORTH )                            // H, NORTH, K
      currLoc = KITCHEN;
    else if( dir == EAST )                        // H, EAST, D
      currLoc = DOOR;
  }
  else if( currLoc == DOOR && dir == WEST )       // D, WEST, H
    currLoc = HALLWAY;
}
```

Encoding the Map

- Map is defined via:
  - A set of constants that refer to specific directions and locations
  - A set of possible moves

Encoding the Map

- Map is defined via:
  - A set of constants that refer to specific directions and locations
  - A set of possible moves
- Alternatively define map as a 2D array
- Tradeoffs in different representations

Encoding the Map

- Map is defined via:
  - A set of constants that refer to specific directions and locations
  - A set of possible moves
- Alternatively define map as a 2D array
- Tradeoffs in different representations
- Area in AI: knowledge representation
  - How to represent language? Facts? Beliefs? Your beliefs of other people's beliefs?
  - How to operate in a world using a specific representation? Limitations?

Moving the Robot

- Moves are random
  - Inefficient, not so smart

Moving the Robot

- Moves are random
  - Inefficient, not so smart
- Always try in order of N, E, S, W
  - Possible infinite loop

Moving the Robot

- Moves are random
  - Inefficient, not so smart
- Always try in order of N, E, S, W
  - Possible infinite loop
- Smarter solutions:
  - Unrepeated moves
  - Look ahead

Moving the Robot

- Moves are random
  - Inefficient, not so smart
- Always try in order of N, E, S, W
  - Possible infinite loop
- Smarter solutions:
  - Unrepeated moves
  - Look ahead
- Area in AI: planning and decision making
  - Which path gets me to the destination? In the shortest time? Least cost?
  - What if a path only works sometimes (probabilistic)? Which path is "best" with the highest chance of success?

Other Programs and Programming Languages

- Key concepts apply exactly the same way
- Big picture planning:
  - What does the program do?
  - What might the output look like?
- Detailed planning:
  - What are the steps, and in what particular order?
  - What information needs to be kept track of?
  - Does it need to be broken up into multiple classes?
  - Any libraries that can/should be used? (e.g. Random, Scanner)
- Don't worry if you haven't seen it before

Another Game Example

- An educational game that helps practice Japanese vocabulary

- Your programming skills are very transferrable

Other Languages: Lua

- Syntax is very similar to Java
- A scripting language, has different behaviours than a compiled programming language like Java
  - No compilation step, just runs the code
  - Therefore, no optimization
  - Does not warn you of errors, just crashes!
- Designed for fast prototyping

Initializing an Array

```
animals = {}.
animals[ 1] = spider
animals[ 2] = butterfly
animals[ 3] = dragon
animals[ 4] = frog
animals[ 5] = hippo
animals[ 6] = monkey
animals[ 7] = owl
animals[ 8] = reindeer
animals[ 9] = whale
animals[10] = bear
animals[11] = ant
animals[12] = raccoon
animals[13] = bird
animals[14] = fish
animals[15] = dog
animals[16] = lion
```

Setting Attributes in the Dictionary

```
fish = {}.
fish.english = "Fish"
fish.hira    = hiragana["sa"] .. hiragana["ka"] .. hiragana["na"]
fish.image   = "fish.png"
fish.audio   = audio.loadSound( "kitsune.wav" )

dog = {}.
dog.english = "Dog"..
dog.hira    = hiragana["i"] .. hiragana["nu"]
dog.image   = "dog.png"
dog.audio   = audio.loadSound( "inu.wav" )

lion = {}.
lion.english = "Lion"
lion.hira    = hiragana["si"] .. hiragana["si"].
lion.image   = "lion.png"
lion.audio   = audio.loadSound( "kitsune.wav" )
```

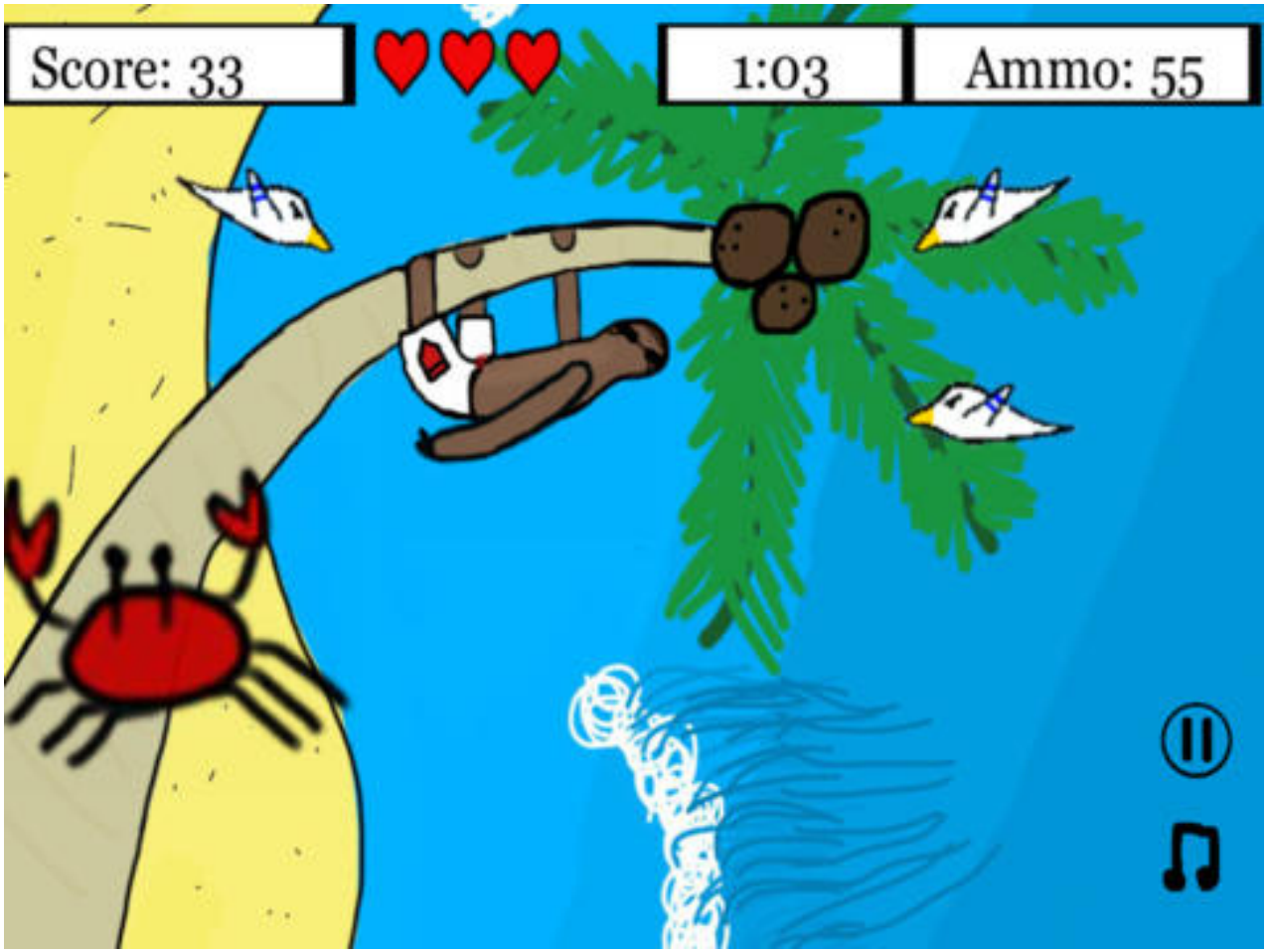
Displaying a Question

```
function showQuestion( grouping )
  questionCounter = questionCounter + 1
  clearPrevQuestion()
  -- generate 3 non-repeated images
  local len  = table.getn( animals )
  local num1 = math.random( 1, len );
  local num2 = genN2( num1, len )
  local num3 = genN3( num1, num2, len )
  -- pick a correct answer
  local ans = math.random( 1, 3 )
  -- set up correct answer
  local answerText
  if ans == 1 then
    currentAnswer = num1
    print( animals[num1].english )
    answerText = display.newText( animals[num1].hira, ansXpos, ansYpos, ansFont, ansSize )
    audio.play( animals[num1].audio )
    a1:addEventListener( "tap", rightAnswer )
    a2:addEventListener( "tap", wrongAnswer )
    a3:addEventListener( "tap", wrongAnswer )
  else
    if ans == 2 then
      currentAnswer = num2
      print( animals[num2].english )
      answerText = display.newText( animals[num2].hira, ansXpos, ansYpos, ansFont, ansSize )
      audio.play( animals[num2].audio )
      a1:addEventListener( "tap", wrongAnswer )
      a2:addEventListener( "tap", rightAnswer )
      a3:addEventListener( "tap", wrongAnswer )
    else
```

```
function wrongAnswer()
  audio.play( wrongSound )
  getNextQuestion()
end

function rightAnswer()
  audio.play( rightSound )
  gameScore = gameScore + 1
  getNextQuestion()
end
```

Stranded Sloth (Also in Lua)



Other Languages: Scheme

- Everything is an expression
  - Numbers, strings, procedures, etc.
  - Everything is true except #f (which denotes false)
- Simple and consistent syntax
- Belongs to a family of languages called functional programming languages
- Most commonly used programming languages are procedural

Examples

```
> 10
10
> (+ 5 5)
10
> (+ 4 4 2)
10
> (+ (+ 1 2) (+ 3 4))
10
```

Examples

```
> 10
10
> (+ 5 5)
10
> (+ 4 4 2)
10
> (+ (+ 1 2) (+ 3 4))
10
```

```
> (define pi 3.141592654)
> pi
3.141592654
> (define (sphere-volume r)
    (* (/ 4 3) pi r r r))
> sphere-volume
<<proc>>
> (sphere-volume 5)
523.598775666665
```

Examples

```
> 10
10
> (+ 5 5)
10
> (+ 4 4 2)
10
> (+ (+ 1 2) (+ 3 4))
10
```

```
> (define (triangle-area base height)
    (* 0.5 base height))
> (triangle-area 2 3)
3
```

```
> (define pi 3.141592654)
> pi
3.141592654
> (define (sphere-volume r)
    (* (/ 4 3) pi r r r))
> sphere-volume
<<proc>>
> (sphere-volume 5)
523.598775666665
```

Examples

```
> 10
10
> (+ 5 5)
10
> (+ 4 4 2)
10
> (+ (+ 1 2) (+ 3 4))
10
```

```
> (define (triangle-area base height)
    (* 0.5 base height))
> (triangle-area 2 3)
3
```

```
> (define pi 3.141592654)
> pi
3.141592654
> (define (sphere-volume r)
    (* (/ 4 3) pi r r r))
> sphere-volume
<<proc>>
> (sphere-volume 5)
523.598775666665
```

```
> (define (square x)
    (* x x))
> (define (sum-of-squares sq x y)
    (+ (sq x) (sq y)))
> (square 2)
4
> (sum-of-squares square 2 3)
13
;; same as (+ (square 2) (square 3))
```

Who Uses Scheme?

- Mostly in university courses
- Variants like Lisp is used in many areas:
  - Original version of part of the Yahoo! store
  - Language processing
  - Market analysis and stock trading
  - Aircraft analysis
  - Music composition, analysis, notation
- For more, see http://www.lispworks.com/success-stories/index.html

Other Languages: Matlab

- (Almost) everything is a matrix!
  - And you can have strings (sequence of chars)
  - Matlab programmers avoid strings for efficiency reasons
- Mostly used for mathematical modeling, data and image processing, simulations and graphs
- Actually code resembles mathematical notation
- Lots of built in math functions
- Also categorized as a procedural language

Examples

```
function m = l2norm( x )
% function m = l2norm( x )
% compute the L2 squared norm of a vector

m = sum(x(:).^2);
```

```
function rvec = reversevec( vec )
% function rvec = reversevec( vec )
rvec = vec(end:-1:1);
% len = length(vec);
% i = 1;
% while 1,
%   if len > 0
%     rvec(i) = vec(len);
%     len = len - 1;
%     i = i + 1;
%   else
%     break;
%   end;
% end;
```

```
> x = [1 2 3 4]
x = [1 2 3 4]
> x = [1 2 3 4];
> l2norm( x )
30
> y = reversevec( x )
y = [4 3 2 1]
>
```

```
a = [ 1 0 0
      1 0 0
      0 1 0 ]
b = [ 0 1 0
      0 0 1
      1 0 0 ]
R = [ 2 0 4.5 ]
beta = 0.9
n = 3                       % number of states
K = 2                       % number of actions
A = zeros(n,n,K)            % set of actions
A(:,:,1) = a
A(:,:,2) = b

inf = 100

V = [];                     % set up value iteration table
V = zeros(n,inf);
V(:,1) = R(s);

% for s = 1:n
%   for j = 1:inf
%     if j == 1
%       V(s,j) = R(s);
%     end
%   end
% end
```

```
for iter = 1:inf,
  for s = 1:n,
    if iter > 1,
        ev = [];
        for actk = 1:K,
            val = A(s,:,actk)*V(:,iter-1);
            ev = [ ev, val ];
        end;
        % if iter == inf, ev, end;
        V(s,iter) = R(s) + beta*max(ev);
    end
    if iter == inf,
        % ev,
        [high, best] = max(ev);
        if( best == 1 )
            p = ' a';
        elseif( best == 2 )
            p = ' b';
        end
        temp = V(s,iter);
        tmp2 = strcat( num2str( temp ), p )
    end
  end
end
```

Next Class

- Matt
  - Undergraduate Research Award (URA)
  - Intelligent tutor for Physics
  - Java implementation of a model of student behaviour and statistical algorithm to estimate how well student knows Physics
- Ethan/Matt
  - Project from my “Intelligent User Interfaces” class
  - Work-Smart monitoring using Kinect camera (live demo!)
  - Java implementation of website analysis and statistical model, C code for Kinect, Java code to control Web browser