arch497i

Week 6

Quiz Review
Project 1 Review
Work time?
Indeterminacy, probability, noise
Some useful math
2-D arrays
Work

Quiz

1. Circle true or false

T F Variables are names that point to a place in the computer's memory.
T F Conditionals always evaluate a boolean expression
T F "Declaration" and "initialization" refer to the same process.
T F Arrays are primitive data types
T F false is a possible value of a variable of type boolean.
T F Decimal numbers are often stored as "floating point values" float
T F Conditionals allow us to iterate through large data sets
T F Libraries are packages of code that enhance the core functionality of a language
T F A function always returns a value

1. Circle true or false

T F Variables are names that point to a place in the computer's memory.
T F Conditionals always evaluate a boolean expression
T **F** "Declaration" and "initialization" refer to the same process.
T **F** Arrays are primitive data types
T F `false` is a possible value of a variable of type boolean.
T F Decimal numbers are often stored as "floating point values" float
T **F** Conditionals allow us to iterate through large data sets
T F Libraries are packages of code that enhance the core functionality of a language
T **F** A function always returns a value

2. `height` and `frameCount` are examples of

Compound data types
Object Oriented Programming
Classes
System variables
None of the above

2. `height` and `frameCount` are examples of

Compound data types
Object Oriented Programming
Classes
System variables
None of the above

3. Consider the following code

```
println ("The division between 10 and 2 is: " + divide(10, 2));

void divide(int a, int b){

      float result = a/b;

}
```

3. Consider the following code

```
println ("The division between 10 and 2 is: " + divide(10, 2));

void divide(int a, int b){

      float result = a/b;

}
```

Add a line of code for the return value:
`return result;`

Change void to float (or int)
void doesn't return a value

```
//solution
float divide(int a, int b){
  float result = a/b;
  return result;
}
```

3. Consider a grading system where numbers are turned into letters. Fill in the blanks in the following code to complete the Boolean expression.

```
float grade = random(0, 100);

if (_________) {

  println("Assign letter grade A");

} else if (______________) {

  println(____________________);

} else if (______________) {

  println(____________________);

} else if (______________) {

  println(____________________);

} else {

  println(____________________);

}
```

3. Consider a grading system where numbers are turned into letters. Fill in the blanks in the following code to complete the Boolean expression.

```
float grade = random(0, 100);
if (_grade >= 80) {
  println("Assign letter grade A");
} else if (__grade >= 60____) {
  println(__"Assign letter grade B"__);
} else if (__grade >= 40____) {
  println(__"Assign letter grade C"__);
} else if (__grade >= 20____) {
  println(__"Assign letter grade D"__);
} else {
  println(__"Assign letter grade F"__);
}
```

4. Consider the following code and write the output

```
int x = 5;
float y = 10.3;
int r1 = 0;
int r2 = 20;
float r3 = y - x;
boolean b = false;
r1 = (int)y/5;
println(r1);
if (y > x){
  b = !b;
}
println(b);
println(r3);
```

4. Consider the following code and write the output

```
int x = 5;
float y = 10.3;
int r1 = 0;
int r2 = 20;
float r3 = y - x;
boolean b = false;
r1 = (int)y/5;
println(r1);
if (y > x){
  b = !b;
}
println(b);
println(r3);
```

```
2
true
5.3
```

5. In the space below, write code for a class named "Tower," which contains the necessary information to instantiate "Tower" objects of a particular height, number of floors, number of elevators, and the bounding volume dimensions. Objects from this class can be added and removed floors, and can be queried for total area.

5. In the space below, write code for a class named “Tower,” which contains the necessary information to instantiate “Tower” objects of a particular height, number of floors, number of elevators, and the bounding volume dimensions. Objects from this class can be added and removed floors, and can be queried for total area.

```
Class Tower{

// data

float towerHeight, towerWidth,
towerDepth;
int numberOfElevators, numberOfFloors

// constructor

Tower(float h, float w, float d, int
nOfE, nOfF){

  towerHeight = h;
  towerWidth = w;
  towerDepth = d;
  numberOfElevators = nOfE;
  numberOfFloors = nOfF;

}
```

```
// methods

  void addFloor(){
    numberOfFloors ++;
  }

  void removeFloow(){
    numberOfFloors --;
  }

  float area(){
    return w*d*nOfF;
  }

} // end of class
```

6. Complete the code below to make the car move and come back.

```
int x = 0;
int y = 50;
int s = 1;

void setup(){
 size(200, 200);
 smooth();
}

void draw(){
  background(255);
  ____________________________;
  ____________________________;
  ____________________________;
}
void move(){
 x = x + s;
}
void bounce(){
 if ((x >= width) || (x <= 0)){
  s *= -1;
 }
}
void drawCar(int x, int y, int size, color c){
 int o = size/4;
 rectMode(CENTER);
 stroke(200);
 fill(c);
 rect(x, y, size, size/2);
 fill(200);
 rect(x-o, y-o, o, o/2);
 rect(x+o, y-o, o, o/2);
 rect(x-o, y+o, o, o/2);
 rect(x+o, y+o, o, o/2);
}
```

6. Complete the code below to make the car move and come back.

```
int x = 0;
int y = 50;
int s = 1;

void setup(){
 size(200, 200);
 smooth();
}

void draw(){
  background(255);
  ____________________________;
  ____________________________;
  ____________________________;
}
void move(){
 x = x + s;
}
void bounce(){
 if ((x >= width) || (x <= 0)){
  s *= -1;
 }
}
void drawCar(int x, int y, int size, color c){
 int o = size/4;
 rectMode(CENTER);
 stroke(200);
 fill(c);
 rect(x, y, size, size/2);
 fill(200);
 rect(x-o, y-o, o, o/2);
 rect(x+o, y-o, o, o/2);
 rect(x-o, y+o, o, o/2);
 rect(x+o, y+o, o, o/2);
}
```

```
move();
bounce();
drawCar(x, y, 50, color(50, 50, 10));
```

7. Complete the code below to draw an ellipse when the mouse is NOT pressed, and a square when it is pressed.

```
if (__________________){
  ellipse(width/2, height/2, 100, 100);
} else {
  rect (______________);
}
```

7. Complete the code below to draw an ellipse when the mouse is NOT pressed, and a square when it is pressed.

```
if (!mousepressed___){
  ellipse(width/2, height/2, 100, 100);
} else {
  rect (_width/w, height);/2, 100, 100
}
```

8. Consider the code below. Complete it so that the circle only starts moving once the mouse has been pressed.

```
boolean ______ = __________;
int circleX = 0;
int circleY = 100;
void setup(){
  size(200, 200);
}
void draw(){
  background(100);
  stroke(255);
  fill(0);
  ellipse(circleX, circleY, 50, 50);
  ______________________________
  ______________________________
  ______________________________
}
void mousePressed(){
  _x++;____________________
}
```

8. Consider the code below. Complete it so that the circle only starts moving once the mouse has been pressed.

```
boolean _m =__ = ___false__;
int circleX = 0;
int circleY = 100;
void setup(){
  size(200, 200);
}
void draw(){
  background(100);
  stroke(255);
  fill(0);
  ellipse(circleX, circleY, 50, 50);
     if (m){
       circleX+=1;
  ___}___________________________
  ______________________________
}
void mousePressed(){
  _m = !m;__________________
}
```

9. Consider the following pseudo-code and implement the code in the space below.

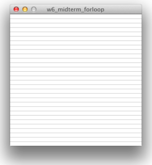
1. Draw a white background
2. Drawn horizontal and vertical lines to divide the window in 4 quadrants.
3. If the mouse is in the top left corner, draw a black rectangle in the top left corner.
4. If the mouse is in the top right corner, draw a black rectangle in the top right corner.
5. If the mouse is in the bottom left corner, draw a black rectangle in the top right corner.
6. If the mouse is in the bottom right corner, draw a black rectangle in the bottom right

9. Consider the following pseudo-code and implement the code in the space below.

```
1. Draw a white background
2. Drawn horizontal and vertical lines to divide the window in 4 quadrants.
3. If the mouse is in the top left corner, draw a black rectangle in the top left corner.
4. If the mouse is in the top right corner, draw a black rectangle in the top right corner.
5. If the mouse is in the bottom left corner, draw a black rectangle in the top right corner.
6. If the mouse is in the bottom right corner, draw a black rectangle in the bottom right
```

```
void setup(){
  size(200, 200);
}
void draw(){
  background(255);
  stroke(0);
  line(100, 0, 100, 200);
  line(0, 100, 200, 100);
  noStroke();
  fill(0);
  if (mouseX<100 && mouseY<100){
    rect(0, 0, 100, 100);
  } else if (mouseX>100 && mouseY<100){
      rect(100, 0, 100, 100);
  } else if (mouseX<100 && mouseY>100){
      rect(0, 100, 100, 100);
  } else if (mouseX>100 && mouseY>100){
      rect(100, 100, 100, 100);
  }
  }
}
```

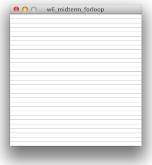
10.



Consider the image above. Complete the code below to produces the image.

```
size(300, 300);
background(255);
for (______________________________){
 stroke(200);
 line(____________________);
}
```

10.



Consider the image above. Complete the code below to produces the image.

```
size(300, 300);
background(255);
for (__int i = 0; i < 30; i++__){
 stroke}(200);
 line(_0, i*10, width, i*10_);
}
```

11. Which of the following is advisable when you are having trouble debugging your code?

a. Call a friend
b. Take a break from writing the code and come back fresh
c. Use `println()` statements to try to trace the error
d. Modularize, simplify and/or clean your code
e. All of the above

11. Which of the following is advisable when you are having trouble debugging your code?

a. Call a friend

b. Take a break from writing the code and come back fresh

c. Use `println()` statements to try to trace the error

d. Modularize, simplify and/or clean your code

[✓ e.] All of the above

Useful math: Probability

Key to the simulation of dynamic systems
Useful to design non-deterministic scenarios by "weighing" the otherwise uniform random function.

1/2	1/6	1/6*1/6

Useful math: Probability

Key to the simulation of dynamic systems
Useful to design non-deterministic scenarios by "weighing" the otherwise uniform random function.

1/2	1/6	1/6*1/6
0.5	0.16	0.027
50%	16%	2.7%

Useful math: Probability

What is the probability of drawing two aces in a row from the deck of cards?

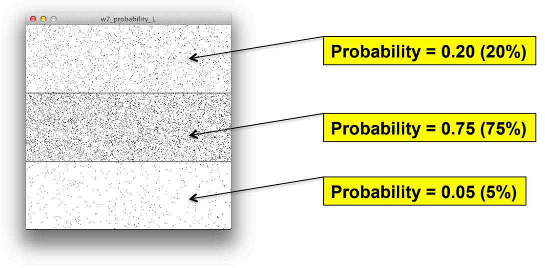
Useful math: Probability

What is the probability of drawing two aces in a row from the deck of cards?

4/52*3/52 = 0.000369822 = 0.00452488688 (approx 0.45%)

Useful math: Probability

How to code it? Try coding the following sketch.

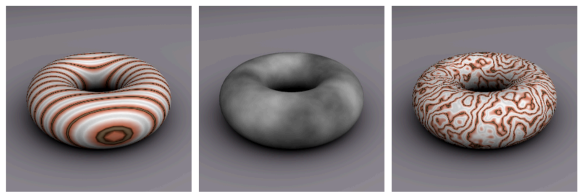


Useful math: Probability

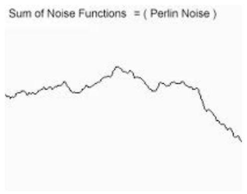
How to code it? Try coding the following sketch.

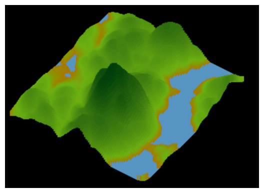
```
…
float PROB_TOP = 0.2;
float PROB_MIDDLE = 0.75;
…

void setup(){
…
  // creates the points
  for (int i = 0; i < POPULATION; i ++){
    r = random(1.0);
    if (r < PROB_TOP){
      point(random(0, width), random(0, height/3));
    } else if (r < PROB_TOP+PROB_MIDDLE){
      point(random(0, width), random(height/3.0, 2*(height/3)));
    } else {
      point(random(0, width), random(2*(height/3), height));
    }
  }
}
```

Constraining indeterminacy: Perlin Noise



http://legakis.net/justin/MarbleApplet/

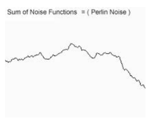




Constraining indeterminacy: Perlin Noise

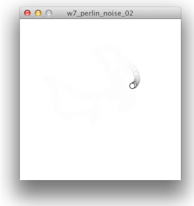
http://legakis.net/justin/MarbleApplet/

```
float t = 0.0;
float increment =0.01;

void setup(){
 size(300, 300);
}

void draw(){

  background(255);
  float noisevalue = noise(t) * width;
  //println(noisevalue);

  ellipse(width/2, height/2, noisevalue, noisevalue);
  t += increment;

}
```



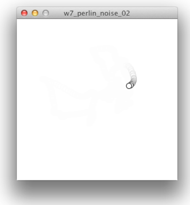
Constraining indeterminacy: Perlin Noise

Can you use Perlin noise to change the location of a circle?



Constraining indeterminacy: Perlin Noise

Can you use Perlin noise to change the location of a circle?



```
float xtime = 0.0;
float ytime = 100.0; // start at a different
point in time to avoid duplicity
float increment = 0.01;

void setup(){
 size(300, 300);
 background(255);
 smooth();
}

void draw(){
  background(255);
  float x = noise(xtime) * width;
  float y = noise(ytime) * height;
  ellipse(x, y, 10, 10);
  xtime += increment;
  ytime += increment;

}
```

Angles

Required for any rotation or 3-D operation in CG.

Basics

-PI (3.1416…) is the number of times any circle's diameter is contained in its circumference.

-A radian is the angle at which the ratio of an arc's length is equal to the circle's radius.

- PI radians is equivalent to a 180 degrees angle.

- A 2 PI radians angle is equivalent to the full circle.

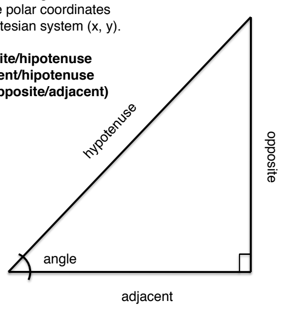
-To work with degrees is easy:

```
float angle = radians(90);
rotate (angle);
```

Some trig: *soh cah toa*

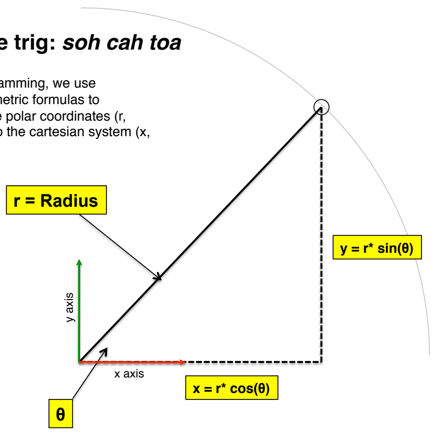
In programming, we use trigonometric formulas to translate polar coordinates (r, angles) to the cartesian system (x, y).

sin(angle) = opposite/hipotenuse
cos(angle) = adjacent/hipotenuse
tangent(angle) = opposite/adjacent)



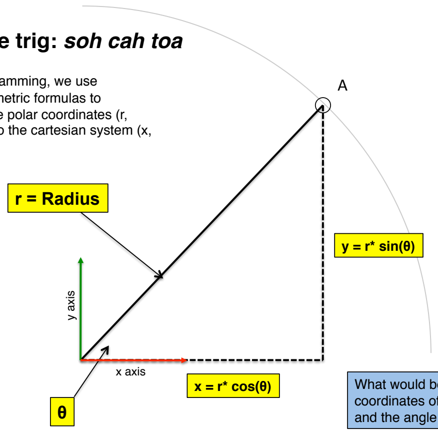
Some trig: *soh cah toa*

In programming, we use trigonometric formulas to translate polar coordinates (r, angle) to the cartesian system (x, y).



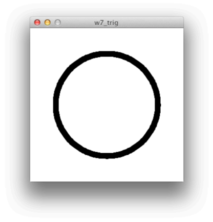
Some trig: *soh cah toa*

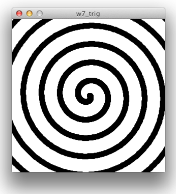
In programming, we use trigonometric formulas to translate polar coordinates (r, angle) to the cartesian system (x, y).



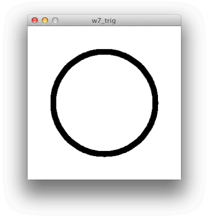
What would be the x, y coordinates of point A if r = 50 and the angle is 80 degrees?

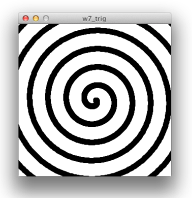
Some trig: *soh cah toa*





Some trig: *soh cah toa*

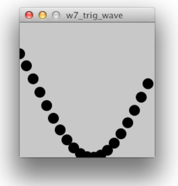


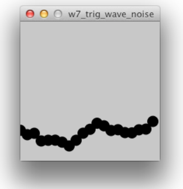


```
void draw(){

  x = width/2 + r*cos(radians(angle));
  y = height/2 + r*sin(radians(angle));

  ellipse(x, y, 10, 10);

  angle += 1;
  //r +=0.1;
}
```

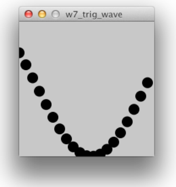
```
void draw(){

  x = width/2 + r*cos(radians(angle));
  y = height/2 + r*sin(radians(angle));

  ellipse(x, y, 10, 10);

  angle += 1;
  r -=0.1;
}
```

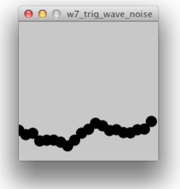
Some trig: *soh cah toa*





Some trig: *soh cah toa*





```
void draw(){
  background(200);
  theta += 0.02;
  noStroke();
  fill(0);
  float x = theta;
  for (int i = 0; i < 20; i++){
    float y = sin(x)*height/2;
    ellipse(i*10, y+height/2, 16, 16);
    x+=0.2;
  }
}
```

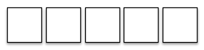
```
void draw(){
  background(200);
  theta += 0.02;
  noStroke();
  fill(0);
  float x = theta;
  for (int i = 0; i < 20; i++){
    float y = noise(x)*height/2;
    ellipse(i*10, y+height/2, 16, 16);
    x+=0.2;
  }
}
```

Two-dimensional arrays

2-D Arrays are arrays of arrays.

Declaration of 1D array

```
int size = 5;
int [] myArray = new int [size]
```

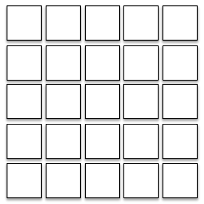


Two-dimensional arrays

2-D Arrays are arrays of arrays.

Declaration of 2D array

```
int cols = 5;
int rows = 5;
int [][] myArray = new int [cols][rows]
```

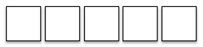


Two-dimensional arrays

2-D Arrays are arrays of arrays.

Initialization and manipulation of 1D arrays

```
for (int i = 0; i < myArray.length; i++){
  myArray[i] = ////some  object or value
}
```



Two-dimensional arrays

2-D Arrays are arrays of arrays.

Initialization and manipulation of 2D arrays

```
for (int i = 0; i < cols; i++){
  for (j = 0; j < rows; j ++){
    myArray[i][j] = //some object or value
  }
}
```