

CS116X Midterm Review – Winter 2015

This review will most likely not cover everything that could possibly be on the exam. The following is intended to help you study but remember that you may be tested on anything covered in Modules 1-5 as well as lectures, assignments, and labs.

Module 1: Processing Review

Types

Every value in a program has a type. Every variable in a program has a type.

Declarations

A declaration introduces a new name into a program.

Expressions

An expression is a fragment of code that yields a definite value.

Statements

A statement is a fragment of code that performs an action.

Module 1 Questions

1. List 6 types that exist in Processing.
2. Give 2 different examples of a declaration.
3. Give 2 different examples of an expression
4. Give 2 different examples of a statement.

Module 2: Input and Output

Reading and displaying images and illustrations

- Images: `PImage`, `loadImage()`, `image()`
- Illustrations: `PShape`, `loadShape()`, `shape()`

Writing images, illustrations, and animations

- Images: `save(filename)`, `saveFrame()`
- Illustrations:
 - `import processing.pdf.*;`
 - `beginRecord(PDF, "filename.pdf");`
 - `endRecord();`

Reading text

`loadStrings(filename)` will produce an array of `Strings`, where each element in the array is one line from the file called `filename`.

Writing text

The following code will write "Text to go in file.." into the file output.txt:

```
PrintWriter writer = createWriter("output.txt");
writer.println("Text to go in file..");
writer.flush();
writer.close();
exit();
```

Module 2 Questions

1. What happens when the following code is run, given that "file.txt" contains the text below:

```
This is a text file.
CS116X is a course
for
GBDA students.

void setup() {
    String[] info = loadStrings("file.txt");
    println(info[2]);
}
```

Module 3: User Interfaces

Model-View-Controller Paradigm

- The **model** is the underlying piece of information that the program is *about*.
- The **view** is the means by which the program expresses the current state of the model.
- The **controller** is the set of controls through which the **user** operates the program.

Direct Manipulation

To determine which handle was manipulated we need to do *hit testing*, which can be tricky, especially if we do not have simple shapes.

ControlP5

Code needed for every sketch using *ControlP5*:

You do NOT need to memorize this.

```
import controlP5.*;

ControlP5 cp5;

void setup() {
  cp5 = new ControlP5(this);
}
```

Steps to add a controller to a *ControlP5* sketch:

1. Use functions such as `addButton()`, `addKnob()`, and `addSlider()` to create the appropriate controller visible in the sketch.
2. Use functions such as `.setValue(0)`, and `.setPosition(100,100)` to send “messages” to configure the controller.
3. Add behavior to the controllers. Use the `controlEvent()` function to make the controllers do something. You can use the `isFrom()` message to determine which controller was manipulated.

Module 3 Questions

1. Use arrows to show the relationship between the model, view, controller, and user.

VIEW

USER

MODEL

CONTROLLER

2. Add a slider to the sketch below, that has the name “Slide Me”, it is 200 wide and 100 tall, with range 0-100 and initial value 50.

```
import controlP5.*;

ControlP5 cp5;

color c = color(255);

void setup() {
  size(500, 500);

  cp5 = new ControlP5(this);

}
```

3. Briefly explain what happens when the Button (called "Button") is clicked in the following sketch.

```

import controlP5.*;

ControlP5 cp5;

color c = color(255);

void setup() {
  size(500, 500);

  cp5 = new ControlP5(this);

  Button button = cp5.addButton("Click Me")
    .setPosition(width/2, height/2)
    .setSize(200, 100);
}

void draw() {
  background(c);
}

void controlEvent(ControlEvent ev){
  c = color(random(255), random(255), random(255));
}

```

Module 4: Physics and Animation

Newton's Laws of motion

- The First Law – “An object in motion tends to remain in motion. An object at rest tends to remain at rest.”
- The Second Law – $F = ma$
 - Gravity
 - Friction
 - Springs
- The Third Law – *Collisions*

Steps to run a physics simulation:

1. Calculate all the forces on each object.
2. Use the forces to update the speeds.
3. Use the speeds to update the positions.
4. Process any collisions.
5. Draw the current state of the world.

When working in two dimensions, you need to make sure these things are updated in both the x and y directions.

Fisica

Code needed for every *Fisica* sketch:

You do NOT need to memorize this.

```
import fisica.*;

FWorld world;

void setup() {
  Fisica.init(this);
  world = new FWorld();
}

void draw() {
  world.step();
  world.draw();
}
```

Animation Principles and Digital Design

- Squash and Stretch
- Slow in, slow out
- Anticipation and follow-through

Module 4 Questions

1. Add a circle (called ball) to the world in the sketch below. Its position must be in the middle of the canvas and it should have size 50.

```

import fisica.*;

FWorld world;

void setup() {
  size(500, 500);

  Fisica.init(this);
  world = new FWorld();

  world.setEdges();

}

void draw() {
  background(255);
  world.step();
  world.draw();
}

```

Module 5: Geometric Context

Translate

`translate(tx, ty)`: from now on move (0,0) to (tx, ty).

Rotate

`rotate(theta)`: from now on draw everything in a context that has been rotated by an angle theta around the point (0,0).

Scale

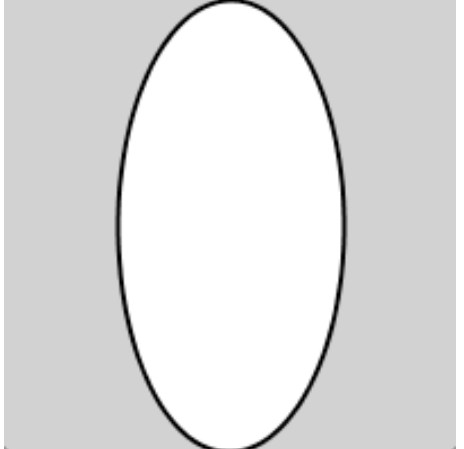

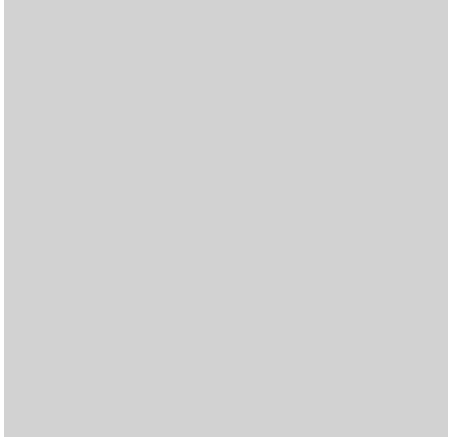
`scale(sx, sy)`: from now on draw everything in a context that has been scaled by a factor of sx in the x direction and sy in the y direction relative to the point (0,0).

Pushing and Popping

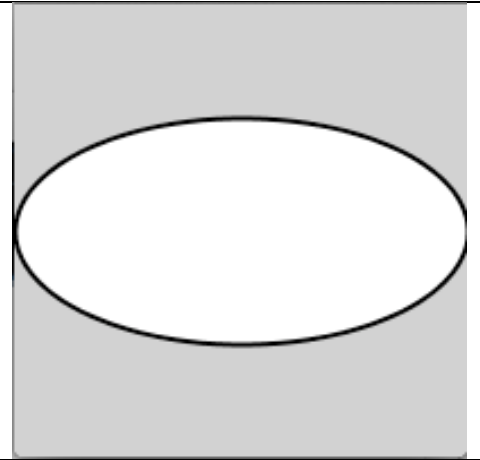
The `pushMatrix()` function saves the current coordinate system to the stack and `popMatrix()` restores the prior coordinate system.

Module 5 Questions

1. Match the code below with its correct sketch.

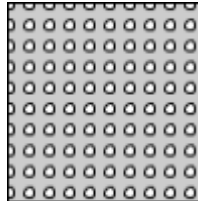
<pre>void setup() { size(200, 200); translate(width/2, height/2); scale(2); ellipse(0, 0, 100, 50); }</pre>	
<pre>void setup() { size(200, 200); rotate(PI/2); scale(2); translate(width/2, height/2); ellipse(0, 0, 100, 50); }</pre>	
<pre>void setup() { size(200, 200); scale(2); translate(width/2, height/2); ellipse(0, 0, 100, 50); }</pre>	


```
void setup() {  
  size(200,200);  
  
  translate(width/2, height/2);  
  rotate(PI/2);  
  scale(2);  
  
  ellipse(0, 0, 100, 50);  
}
```



Sample Questions

1) (CS 115x Refresher) Fill in the blanks so that the sketch produces the following drawing.



```
int diameter = 5;
int spacing = 10;

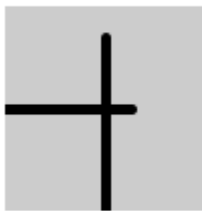
for ( _____; _____; _____) {
  for ( _____; _____; _____) {
    ellipse( _____, _____, _____, _____);
  }
}
```

Note: For full marks, you should not assume that the canvas will be a particular size.

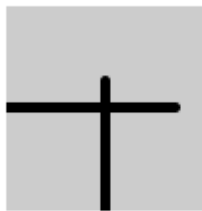
2) Pretend it's currently 7:50 pm. Which drawing will be the product of the following sketch?

```
int h = hour() % 12;
int m = minute();

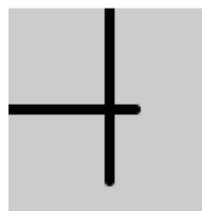
strokeWeight(5);
line(0, height/2, map(h, 0, 11, 0, width), height/2);
line(width/2, map(m, 0, 59, width, 0), width/2, height);
```



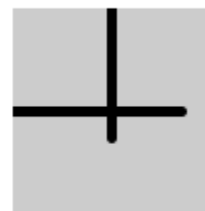
(A)



(B)



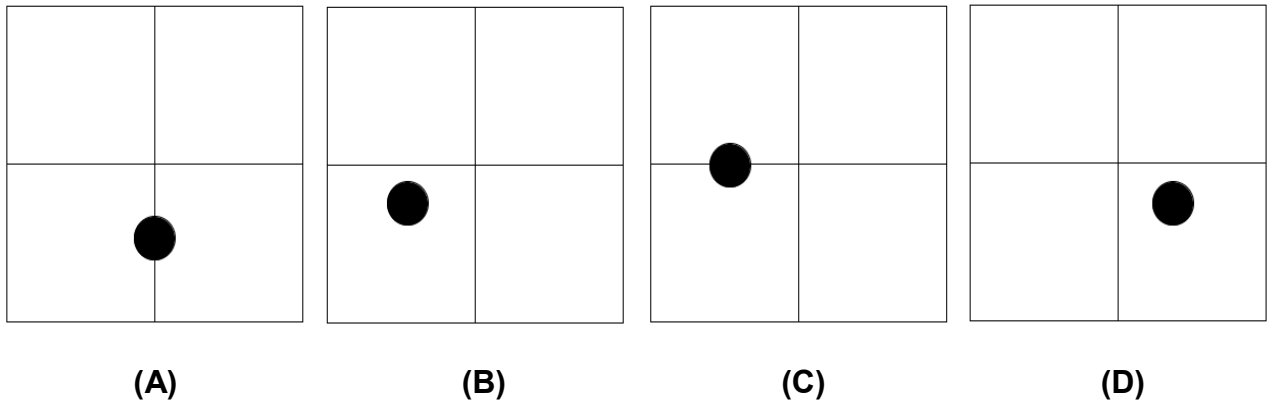
(C)



(D)

3) Given the following setup() function for a standard Fisica simulation, where would you expect the ball to be after approximately 10 calls to world.step() (i.e. 10 frames)? You may assume that the ball does not reach any edge within this time.

```
void setup() {  
  size(300, 300);  
  
  Fisica.init(this);  
  world = new FWorld();  
  world.setGravity(0, 200);  
  
  ball = new FCircle(40);  
  ball.setFill(0);  
  ball.setPosition(width/2, height/2);  
  ball.setVelocity(-100, 0);  
  ball.setRestitution(1);  
  ball.setDamping(0);  
  ball.setFriction(0);  
  world.add(ball);  
}
```



Note: The vertical and horizontal lines are merely visual guides for the middle of each axis.

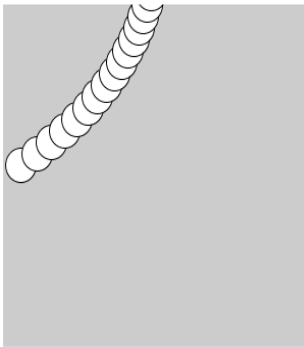
4) What will be the output of the following sketch?

```
float x, y;
float vx = 20;
float vy = -10;

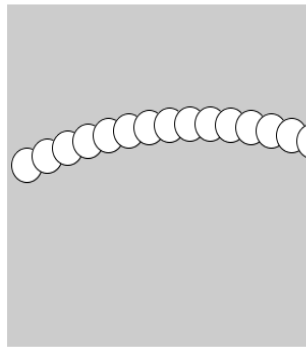
void setup() {
  size(300, 300);
  x = 0;
  y = height/2;
}

void step() {
  vx *= 0.9;
  vy = (vy + 1);
  x += vx;
  y += vy;
}

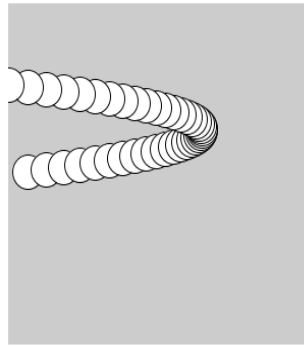
void draw() {
  step();
  ellipse(x, y, 30, 30);
}
```



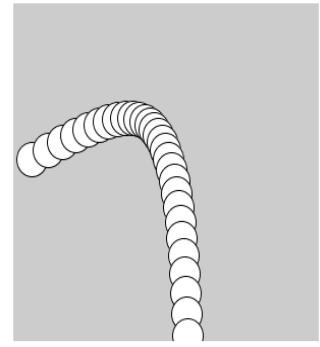
(A)



(B)



(C)



(D)