Computation as an Expressive Medium

Lab 3: Shapes, Rockets, Mice, Cookies and Random Stuff

Joshua Cuneo
Agenda Time

- Project 1
- Array Loops, PImage, Fonts
- Drawing polygons
- Trigonometry review
- random()
- Methods and objects
- Mouse functions (again)
- Assignment 2!
Project 1

From the central heartbeat of the central processor, to the obsessive timestamping of files and blog entries, to ever present clock displays, time is a fundamental feature of computation. Display the progress of time in a non-traditional way. It is OK to consider large temporal scales (e.g. seasons), but smaller temporal scales should also be displayed (or be available to be displayed, perhaps as a function of user input). You may make use of mouse input if you wish.
Debugging Tips

- Commenting
- `println()`
- Tracing
- Common errors
  - Semicolons
  - Parentheses and braces
  - Speling erorz
- Program arrangement
  - Variable declaration
  - `setup()`
  - `draw()`
Array Loops

for(int i = 0; i < numbers.length; i++)
{
    numbers[i] = 3;
}
Array Loops

```java
for(int i = 0; i < numbers.length; i++)
{
    numbers[i] = 3;
}
```
for(int i = 0; i < numbers.length; i++)
{
    numbers[i] = 3;
}
Array Loops

```java
for(int i = 0; i < numbers.length; i++)
{
    numbers[i] = 3;
}
```
Array Loops

```java
for(int i = 0; i < numbers.length; i++)
{
    numbers[i] = 3;
}
```
Array Loops

for(int i = 0; i < numbers.length; i++)
{
    numbers[i] = 3;
}
Array Loops

```java
for(int i = 0; i < numbers.length; i++)
{
    numbers[i] = 3;
}
```
1. Bring the image into Processing

   PImage startrek = loadImage("startrek.jpg");

2. Load the image to our canvas

   image(startrek, 0, 0);
Fonts

- Create a font on the fly
  
  ```
  PFont verdana = createFont("Verdana", 32);
  ```

- OR Import a .vlw file
  
  ```
  Pfont tng = loadFont("tng-32.vlw");
  ```

- Use the font
  
  ```
  textFont(tng);
  text("To Boldly Go", 10, 30);
  ```
Getting Ready for Assignment 2
Building Polygons

- `beginShape(POLYGON);`
  - Start the polygon
- `vertex(x, y);`
  - One per vertex point
- `endShape();`
  - Finish the polygon
Building Polygons

- `beginShape(POLYGON);`
- `vertex(10, 50);`
Building Polygons

- vertex(20, 10);
- vertex(30, 40);
- vertex(80, 60);
- vertex(40, 80);
Building Polygons

- endShape();
Let’s Use Arrays

```plaintext
int[] xvals = {10, 20, 30, 80, 40};
int[] yvals = {50, 10, 40, 60, 80};

beginShape(POLYGON);

for(int i = 0; i < xvals.length; i++)
{
    vertex(xvals[i], yvals[i]);
}

endShape();
```
WTF?

circle of radius 1 centered at (0,0)

Angles are formed counterclockwise from the x axis

\[ y = \sin \theta \]

\[ x = \cos \theta \]
Trig 101

\[ \sin(\Theta) = \frac{\text{opp}}{\text{hyp}} \]

\[ \cos(\Theta) = \frac{\text{adj}}{\text{hyp}} \]

\[ \tan(\Theta) = \frac{\text{opp}}{\text{adj}} \]
So What?
So What?
Working Example

\[
\cos(\Theta) = \frac{\text{adj}}{\text{hyp}}
\]

\[
\cos(60^\circ) = \frac{\text{adj}}{5}
\]

\[
5 \times \cos(60^\circ) = \text{adj}
\]

\[
\text{adj} = 2.5
\]
int x = 0;
int y = 0;

void draw()
{
    x = x + 5*cos(60);
    y = y + 5*sin(60);
}
A Pox on Radians

```c
int x = 0;
Int y = 0;

void draw()
{
    x = x + 5*cos(radians(60));
    y = y + 5*sin(radians(60));
}
```
random()

float r = random(0, 5);

0
0.5
2.3
2.776
3.7543929
4.9999999999999999...
random()

int r = int(random(0, 5));
random()

Name some uses
Methods, aka Functions

- **Method declaration**
  
  ```java
  void vendingMachine( int coinCents )
  {
      println("You inserted "+coinCents+" cents.");
  }
  ```

- **Method call**
  
  ```java
  int quarter = 25;
  vendingMachine(quarter);
  ```
Classes

C is for Cookie
Classes, aka Blueprints

class MyToy
{
    // fields (class variables)
    // methods (class functions)
}

class Cookie
{
    int numberChips;

    Cookie(int chips)
    {
        numberChips = chips;
    }

    void drawCookie()
    {
        ellipse(numberChips, numberChips, 10, 10);
    }
}
class Cookie
{
    int numberChips;

    Cookie(int chips)    
    {
        numberChips = chips;
    }

    void drawCookie()   
    {
        ellipse(numberChips, numberChips, 10, 10);
    }
}

Cookie chocChip = new Cookie(20);
Cookie doubleChocChip = new Cookie(40);
class Cookie
{
    int numberChips;

    Cookie(int chips)
    {
        numberChips = chips;
    }

    void drawCookie()
    {
        ellipse(numberChips, numberChips, 10, 10);
    }
}

void draw()
{
    chocChip.drawCookie();
    doubleChocChip.drawCookie();
}
Subclasses

new and improved
class ChocolateCookie extends Cookie
{
    int amtChoclate;

    ChocolateCookie(int amt)
    {
        super(amt);
        amtChoclate = amt;
    }

    void drawCookie()
    {
        fill(188, 143, 143);
        super.drawCookie();
    }
}
OOP, we'll do it again.

- Programming practices
- Methods
- Classes

The more the merrier!

MUHAHAHA!
Mice

- mouseButton
- mouseClicked()
- mouseDragged()
- mouseMoved()
- mousePressed()
- mouseReleased()

- mouseX
- mouseY
- pmouseX
- pMouseY
Assignment 2!

- A2-01: Using `beginShape()` and `endShape()`, create a composition with five or more vertices.
- A2-02: Using `beginShape()` and `endShape()`, create a composition with ten or more vertices.
- A2-03: Create an image different from A2-02, but using the same vertex data.
- A2-04: Write a function with one parameter and demonstrate it visually.
- A2-05: Write a function for drawing triangles and visually demonstrate its flexibility.
- A2-06: Write a function with three or more parameters and visually demonstrate its flexibility.
- A2-07: Create a dynamic animation using the `cos()` function as a generator for motion.
- A2-08: Create a dynamic animation using the `cos()` and `sin()` function as a generator for motion.
- A2-09: Move two visual elements across the screen using the `random()` function as a generator of movement. Give each element a unique nonlinear motion.
- A2-10: Create an event that begins when the mouse is pressed and ends when the mouse is released.
- A2-11: Create a responsive image that behaves differently when the mouse is moving and the mouse is dragging.
- A2-12: Create a button that changes the color of the background when it is clicked.
- A2-13: Program your moving elements from A2-09 but use classes to represent the two visual elements.
- A2-14: Create a subclass of one of the asteroids classes that adds a new capability. Some examples of what you could do: create a subclass of Rocket (or ArmedRocket) that shoots flame when the thrusters are fired and/or plays a sound when thrusters are fired, create a subclass of Asteroid that know when it's been hit (instead of doing this test in `draw()`), create a subclass of Asteroid that splits into two smaller Asteroids when it's hit.