Transformations 2

CS110
Transformations Reviewed

We can think of the Processing window as being a field of pixels on which we draw.

How do we know where processing will draw the ellipse?

```
size(500,500);
ellipse(250,250,200,200);
```
Transformations Reviewed

How we’ve done things thus far:

```plaintext
size(500,500);
ellipse(250,250,200,200);
```

Center pixel
Transformations Reviewed

How we’ve done things thus far:

```plain
size(500,500);
ellipse(250,250,200,200);
```

Width and height in pixels
Transformations Reviewed

How we’ve done things thus far:

```
size(500,500);
ellipse(250,250,200,200);
```
Separating the Display Window from the Coordinate System

• We are used to thinking of the arguments passed to drawing functions as specifying which pixels to draw the object at.

• Transformations introduce the idea of a coordinate system that is not necessarily the same as pixel positions within the window.
The Coordinate System

Origin (0,0)
Initial Alignment of Main Window
A Simple Example Revisited

```javascript
size(500,500);
ellipse(250,250,200,200);
```

Note: not 250 Pixels!
A Simple Example Revisited

```
size(500,500);
ellipse(250,250,200,200);
```

Note: not 200 Pixels!
A Simple Example Revisited

```plaintext
size(500,500);
ellipse(250,250,200,200);
```
A Simple Example Revisited

```
size(500,500);
ellipse(250,250,200,200);
```

Once we have computed the shape in the coordinate system, all we do is transfer it to the main window!

Once the shape is transferred, its appearance is not affected by subsequent transformations
So What are Transformations?

Transformations simply move the coordinate system without moving the processing window!

Three main operations:

translate()

scale()

rotate()
Translate

size(500,500);
translate(100,100);
ellipse(250,250,200,200);
Translate

size(500,500);
translate(100,100);
ellipse(250,250,200,200);
Translate

```cpp
size(500, 500);
translate(100, 100);
ellipse(250, 250, 200, 200);
```

Note: not 250 Pixels!
Translate

```
size(500,500);
translate(100,100);
ellipse(250,250,200,200);
```

Note: not 250 Pixels!
Translate

size(500,500);
translate(100,100);
ellipse(250,250,200,200);

Note: not 200 Pixels!
Translate

\texttt{size(500,500);}
\texttt{translate(100,100);}
\texttt{ellipse(250,250,200,200);}
Translate

```javascript
size(500,500);
translate(100,100);
ellipse(250,250,200,200);
```

Once we have computed the shape in the coordinate system, all we do is transfer it to the main window!
scale(1,.5);
elipse(250,250,200,200);
Scale

```javascript
size(500, 500);
scale(1, .5);
ellipse(250, 250, 200, 200);
```
Scale

```plaintext
size(500,500);
scale(1,.5);
ellipse(250,250,200,200);
```

Note: not 250 Pixels!
Scale

```plaintext
size(500,500);
scale(1, .5);
ellipse(250, 250, 200, 200);
```

Note: not 200 Pixels!
scale(1,.5);
Once we have computed the shape in the coordinate system, all we do is transfer it to the main window!
Rotate

\begin{align*}
\text{size}(500,500) ; \\
\text{rotate}(\pi/6) ; \\
\text{ellipse}(250,250,200,200) ; \\
\text{angle in degrees} = \text{angle in radians} \times \frac{180}{\pi}
\end{align*}
Rotate

\[
\text{angle in degrees} = \text{angle in radians} \times \frac{180}{\pi}
\]

```javascript
size(500, 500);
rotate(\pi/6);
ellipse(250, 250, 200, 200);
```
size(500,500);
rotate(PI/6);
ellipse(250,250,200,200);

Note: not 250 Pixels!

250 Units

Origin (0,0)
Center point

250 Units

250 Units
Rotate

```
size(500,500);
rotate(\pi/6);
ellipse(250,250,200,200);
```

Note: not 200 Pixels!
size(500,500);
rotate(\pi/6);
ellipse(250,250,200,200);
Scale

```javascript
size(500,500);
rotate(PI/6);
ellipse(250,250,200,200);
```

Once we have computed the shape in the coordinate system, all we do is transfer it to the main window!
Combining Transformations

size(500,500);
translate(250, 250);
rotate(PI/6);
Combining Transformations

\begin{verbatim}
size(500,500);
translate(250,250);
rotate(\pi/6);
\end{verbatim}

Origin (0,0)
Combining Transformations

```javascript
size(500,500);
translate(250,250);
rotate(\pi/6);
```
Is this the same as the previous example?

\begin{verbatim}
size(500,500);
rotate(Pi/6);
translate(250,250);
\end{verbatim}
Transformations and the draw loop

• All transformations are reset each time the draw loop is called
• We will see another way to undo transformations in a few minutes
Ball With Eye Example Revisited
Ball With Eye Code

```
Ball[] balls = new Ball[10];

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);

  // Create all new Ball objects
  for (int i = 0; i < balls.length; i++) {
    balls[i] = new Ball();
  }
}

void draw() {
  background(255);
  for (int i = 0; i < balls.length; i++) {
    balls[i].update();
    balls[i].display();
  }
}
```
Ball With Eye Code

```java
void display() {
    fill(c);
    translate(sx, sy);
    if (vx < 0) {
        scale(-1, 1);
    }
    // draw the body of the ball
    ellipse(0, 0, d, d);
    // drawing code cut for brevity
    if (vx < 0) {
        scale(-1, 1);
    }
    translate(-sx,-sy);
}

class Ball {
    // Fields
    float ay = 0.2; // y acceleration (gravity)
    float sx; // x position
    float sy; // y position
    float vx; // x velocity
    float vy; // y velocity
    float d;
    color c;
}
```
pushMatrix() and popMatrix()

Undoing transformations manually is annoying

Alternative:

pushMatrix(): save the current coordinate system

popMatrix(): revert to the most recently saved coordinate system
Push and Pop Matrix Example

```plaintext
size(500,500);
pushMatrix();
translate(250,250);
pushMatrix();
rotate(PI/6);
popMatrix();
popMatrix();
```
Push and Pop Matrix Example

```javascript
size(500, 500);
pushMatrix();
translate(250, 250);
pushMatrix();
rotate(Pi/6);
popMatrix();
popMatrix();
```

Saved Coordinate Systems:
Push and Pop Matrix Example

```javascript
size(500,500);
pushMatrix();
translate(250,250);
pushMatrix();
rotate(PI/6);
popMatrix();
popMatrix();
```

Saved Coordinate Systems:
Push and Pop Matrix Example

```cpp
size(500, 500);
pushMatrix();
translate(250, 250);
pushMatrix();
rotate(PI/6);
popMatrix();
popMatrix();
```

Saved Coordinate Systems:
Push and Pop Matrix Example

```
size(500, 500);
pushMatrix();
translate(250, 250);
pushMatrix();
rotate(Pi/6);
popMatrix();
popMatrix();
```

Saved Coordinate Systems:
Push and Pop Matrix Example

```plaintext
size(500, 500);
pushMatrix();
translate(250, 250);
pushMatrix();
rotate(PI/6);
popMatrix();
popMatrix();
```

Saved Coordinate Systems:
Push and Pop Matrix Example

```java
size(500, 500);
pushMatrix();
translate(250, 250);
pushMatrix();
rotate(PI/6);
popMatrix();
popMatrix();
```

Saved Coordinate Systems:
Advertisement the Next Computer Science Course: Data Structures

• Data structures is all about how to organize data in the computer in order to perform interesting computation

• The data structure we store the coordinate systems is called a stack

  Last in first out structure
  Always take the top plate
  (unless you are a magician)
Advertisement Part 2

Where does the matrix part come in?!?!?
How Can We Modify The Ball Code?

```java
void display() {
    fill(c);
    translate(sx, sy);
    if (vx < 0) {
        scale(-1, 1);
    }
    // draw the body of the ball
    ellipse(0, 0, d, d);

    // drawing code cut for brevity
    if (vx < 0) {
        scale(-1, 1);
    }
    translate(-sx,-sy);
}

class Ball {
    // Fields
    float ay = 0.2; // y acceleration (gravity)
    float sx; // x position
    float sy; // y position
    float vx; // x velocity
    float vy; // y velocity
    float d;
    color c;
}
```
Rotating Ball Example
Spirograph

http://www.youtube.com/watch?v=LbvmKzf_wr4
Spirograph Mechanism

- **Center of larger circle**
- **Center of smaller circle**
- **innerOffset**
- **smallerRadius**
- **largerRadius**
- Pen tip goes here
Spirograph Example
Spirograph Motion

The motion of the pen-tip can be decomposed into two rotations.

Rotation 1: rotation around the center of the larger circle.
Spirograph Motion

The motion of the pen-tip can be decomposed into two rotations

Rotation 2: rotation around the center of smaller

innerAngle = -outerAngle \times \frac{largerRadius}{smallerRadius}
Spirograph Code Starter

```java
float smallerRadius;
float largerRadius;
float innerOffset = 100;
float outerAngle = 0;
float dOuterAngle = .01;
float innerAngle = 0;
color penColor = color(255, 0, 0);

void setup() {
  size(500, 500);
  smooth();
  background(255);
  largerRadius = width/2.0;
  smallerRadius = width*7.0/25.0;
  frameRate(1000);
}

void mouseClicked() {
  // choose a new random pen color, inner circle size, and inner offset
  penColor = color(random(0, 255), random(0, 255), random(0, 255));
  smallerRadius = random(0, largerRadius);
  innerOffset = random(0, smallerRadius);
}
```
Example: Rotated Polygons

```java
void setup() {
    size(600, 600);
    smooth();
    noFill();
    translate(width/2, height/2);
    float numPentagons = 250;
    for (int i=0; i<numPentagons; i++) {
        rotate(random(0, PI/4));
        stroke(50-i, 50+i, 150+i, i);
        pentagon(numPentagons-i);
    }
}

void pentagon(float r) {
    beginShape();
    for (float theta=0; theta<TWO_PI; theta += TWO_PI/5) {
        vertex(cos(theta)*r, sin(theta)*r);
    }
    endShape(CLOSE);
}
```
A starfield using transformations.
We want to find the point where each star is projected on our viewport.

\[
\frac{x'}{z'} = \frac{x}{z}
\]

\[
x' = z'\left(\frac{x}{z}\right)
\]
class Star {
    // Star coordinates in 3D
    float x;
    float y;
    float z;

    Star() {
        x = random(-5000, 5000);
        y = random(-5000, 5000);
        z = random(0, 2000);
    }

    void update() {
        // Move star closer to viewport
        z -= 10;

        // Reset star if it passes viewport
        if (z <= 0.0)
            reset();
    }

    void reset() {
        // Reset star to a position far away
        x = random(-5000, 5000);
        y = random(-5000, 5000);
        z = 2000.0;
    }

    void draw() {
        // Project star only viewport
        float offsetX = 100.0*(x/z);
        float offsetY = 100.0*(y/z);
        float scaleZ = 0.0001*(2000.0-z);

        // Draw this star
        pushMatrix();
        translate(offsetX, offsetY);
        scale(scaleZ);
        ellipse(0,0,20,20);
        popMatrix();
    }
}
// starfield

// Array of stars
Star[] stars = new Star[400];

void setup() {
    size(600, 600);
    smooth();
    stroke(255);
    strokeWeight(5);
    rectMode(CENTER);

    // Init all stars
    for (int i=0; i<stars.length; i++)
        stars[i] = new Star();
}

void draw() {
    background(0);

    // Draw all stars wrt center of screen
    translate(0.5*width, 0.5*height);

    // Update and draw all stars
    for (int i=0; i<stars.length; i++) {
        stars[i].update();
        stars[i].draw();
    }
}