CMSC427
Transformations II: Modeling

Credit: some slides from Dr. Zwicker

What next?

- GIVEN THE TOOLS OF …
- The standard rigid and affine transformations
- Their representation with matrices and homogeneous coordinates

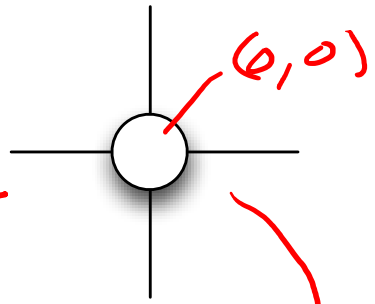
- WHAT CAN WE DO WITH THESE TOOLS?
- Modeling – how can we define transformations we want?
- Viewing – how can we use these tools to render objects like polygonal meshes?

Review: modeling with transformations

- Object space
- World space
- Create scene by transforming objects from object to world

- Object coordinate space

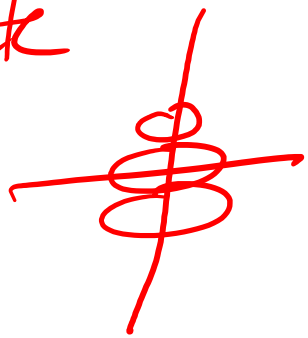
natural c. space

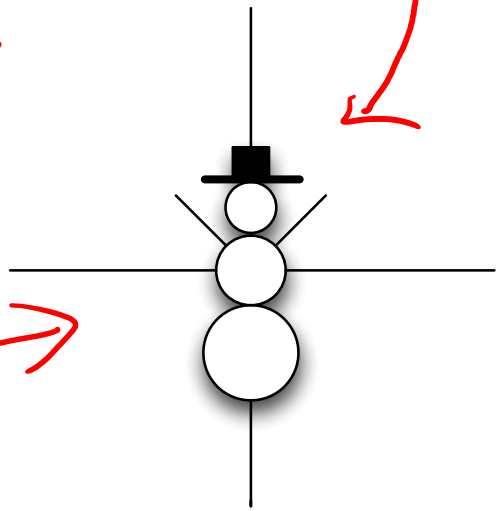


- World coordinate space

scene

intermediate object





Modeling

- Shape object
  - Size, reshape
- Place object
  - Position and orientation

Model matrix

$M = I$

$M = I * M_T$

$M = M_T * M_{RX}$

$M = M_T * M_{RX}$

Display pt = M * box

$DP = M_T * M_{RX} * M_{RX} * P$

```
// Processing example

size(200,200,P3D);

translate(width/2,height/2,0);
rotateY(PI/4);
rotateX(PI/4);

box(50);
```

3D

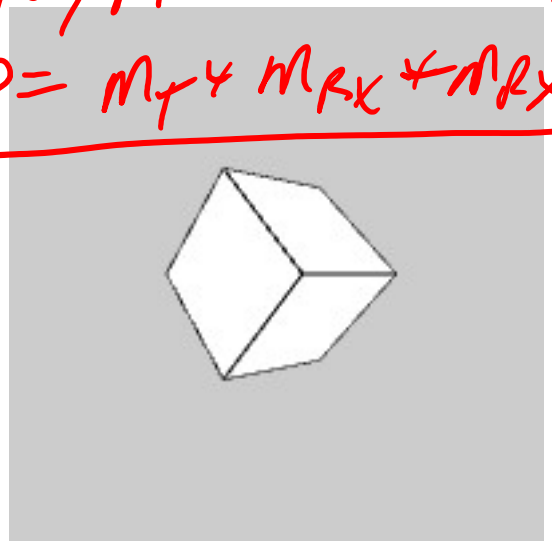
3 translate(width/2,height/2,0); M_T

2 rotateY(PI/4);

1 rotateX(PI/4);

last is first applied

box(50); – box has pts P



Transformations in Processing (OpenGL 1.0 and 2.0 style)

- **Transforms**
- applyMatrix()
- popMatrix()
- printMatrix()
- pushMatrix()
- resetMatrix()
- rotate()
- rotateX()
- rotateY()
- rotateZ()
- scale()
- shearX()
- shearY()
- translate()

manage transforms

basic trans

single global M matrix

```
resetMatrix();
// M=I
translate(5,5,5);
scale(2,1,2);
rotate(π/2);
//draw shapes
```

Transformations in Processing (OpenGL 1.0 and 2.0 style)

- **Transforms**
- applyMatrix()
- popMatrix()
- printMatrix()
- pushMatrix()
- resetMatrix()
- rotate()
- rotateX()
- rotateY()
- rotateZ()
- scale()
- shearX()
- shearY()
- translate()

- Camera
- beginCamera()
- camera()
- endCamera()
- frustum()
- ortho()
- perspective()

- Tracing
- printMatrix()
- printCamera()
- printProjection()

- Routines *not* in OpenGL 3.0/4.0 but in many utility libraries

Observation 1: Parametric transformations

- Instead of …

```
for (int t=0; t < 2*PI; t += 0.1) {
  float x = cx+ r*cos(t);
  float y = cy+ r*sin(t);
  ellipse(x,y,10,10);
}
```

- Use …

```
for (int t=0; t < 2*PI; t += 0.1) {
  float x = cx+ r*cos(t);
  float y = cy+ r*sin(t);
  translate(x,y); // Or more …
  complexObject();
}
```

ellipse(0,0,10,10);

object coordinates

- Why? Simplify the code for the object, enable complex transformations.
- Mesh not need understand transformations

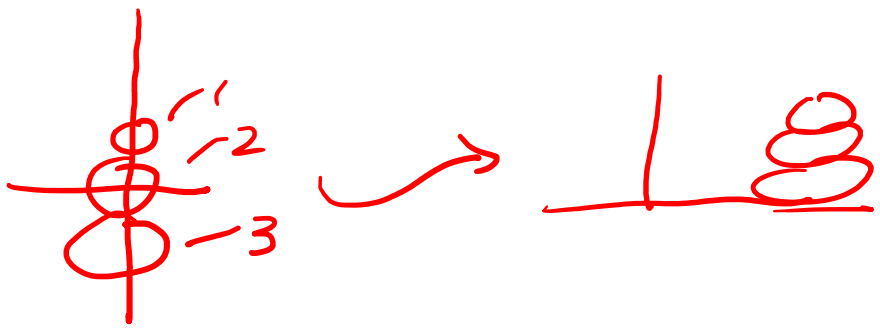
Observation 1: Problem!

- Transforms can accumulate

```
for (int t=0; t < 2*PI; t += 0.1) {
  float x = cx+ r*cos(t);
  float y = cy+ r*sin(t);
  pushMatrix();
  translate(x,y); // Or more …
  complexObject();
  popMatrix();
}
```

- translate(5,5);
- translate(10,50);
- Result: (15,15)

- pushMatrix() preserves existing matrix, popMatrix() restores

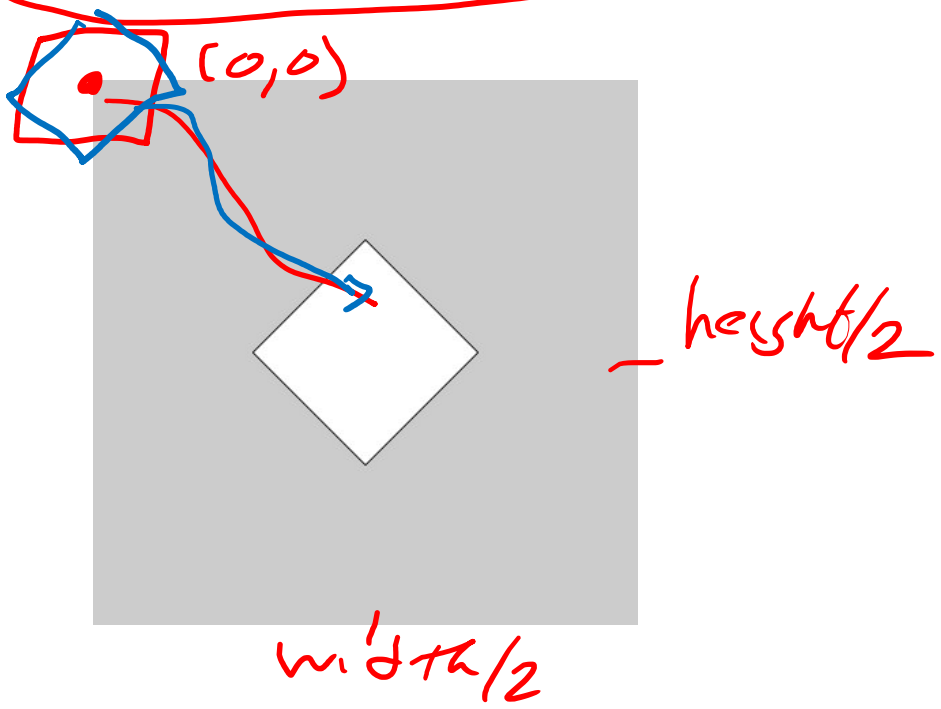


Observation II: Experimenting with 3D transforms

- Get to know transformations by experimentation
- Processing good for this
- Try
  - translate
  - scale
  - shearX, shearY
  - rotateX,rotateY,rotateZ
- Understand
  - Direction of x, y, z
  - Direction of rotations

```
// Basic code

size(400,400,P3D);
translate(width/2,height/2,0);
rotateZ(PI/4);
box(100);
```



Observation II: Problem (Processing scales outline stroke)

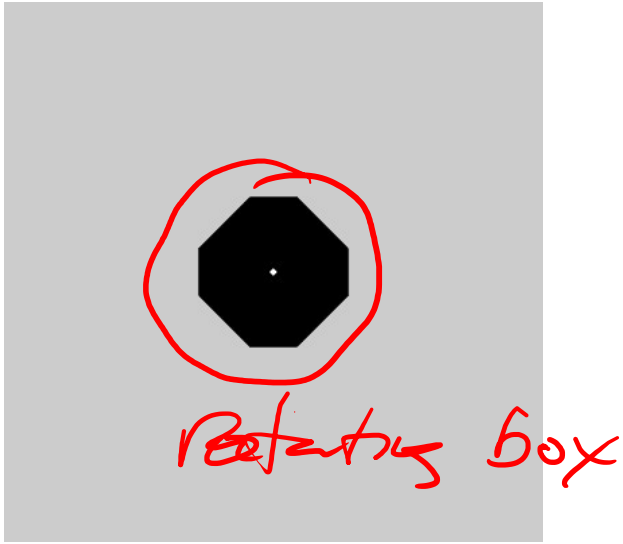
- More elegant code

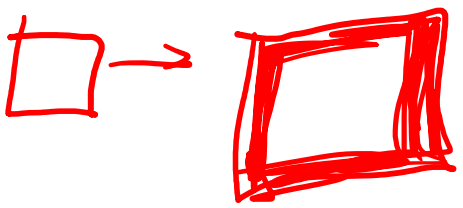
- scale(50);
- box(1);

many built-in objects are at (0,0)

- But we get this picture
- ?????????
- strokeWeight is scaled

scale of 1





- So … box(50) it is.

Observation 3: Tracing matrix

- Can print current transformation matrix to debug

```
size(400,400,P3D);
translate(width/2,height/2,0);
rotateZ(PI/4);
printMatrix();
box(50);
```

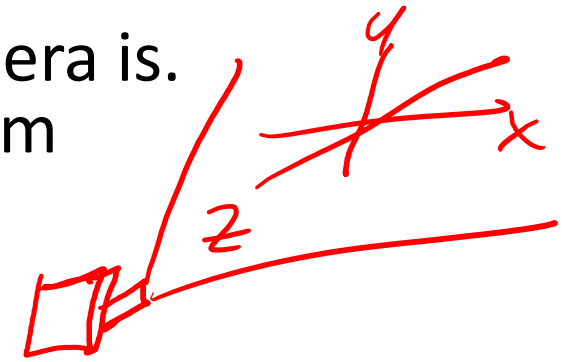
4x4 homogeneous

rotation and scale

```
000.7071 -000.7071  000.0000  000.0000
000.7071  000.7071  000.0000  000.0000
000.0000  000.0000  001.0000 -346.4102
000.0000  000.0000  000.0000  001.0000
```

translate

- Why -346? Where camera is. Viewing from +346 in Z



Experiments!

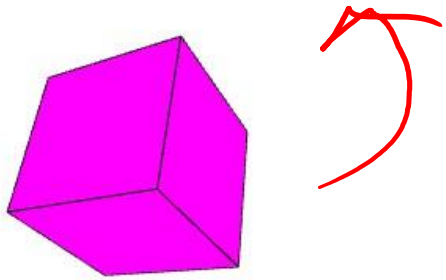
- Translate with positive and negative X,Y,Z
  - Figure out the coordinate system
- Rotate around X, Y, Z in different orders
- Scale non-uniformly in X,Y,Z
- Change order of scale, rotate, translate
- Try a shear

Transformations and animations

```
float theta = 0;

void setup(){
 size(400,400,P3D);
 fill(255,0,255);
}

void draw(){
 background(255);
 translate(width/2,height/2,0);
 rotateZ(theta);
 rotateX(theta);
 rotateZ(theta);
 box(100);
 theta += 0.01;
}
```



erase screen

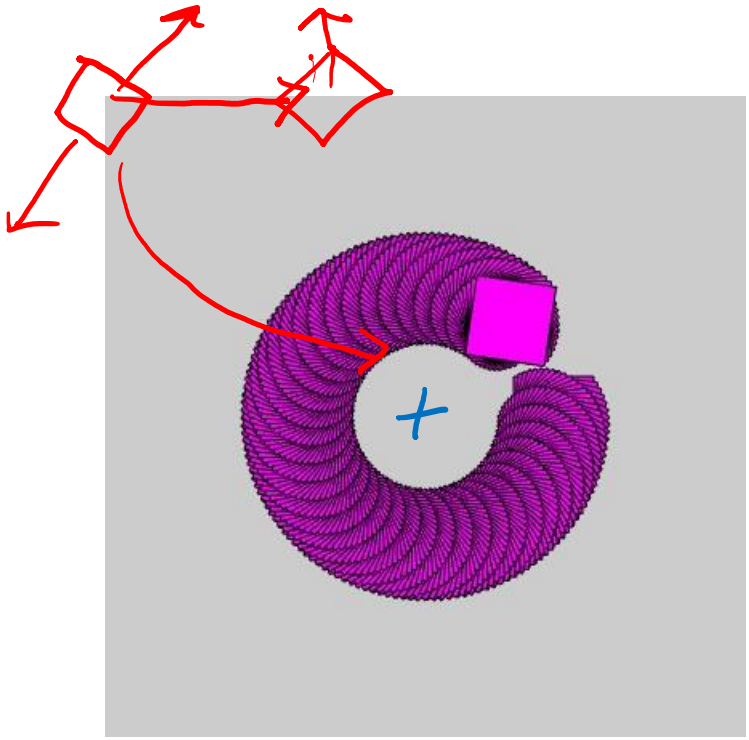
2 x Z

one time

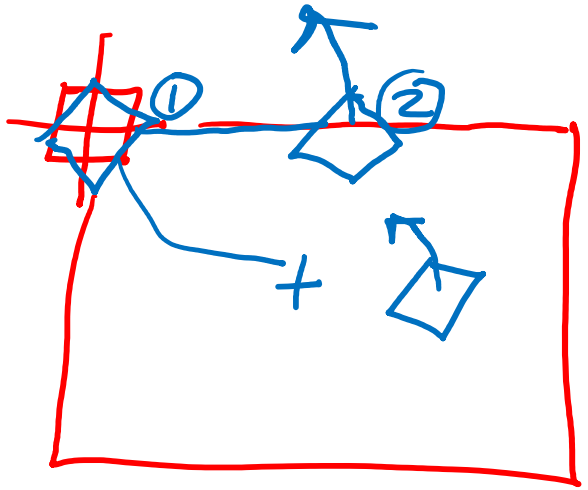
increment theta

Orbiting box?

- How animate box rotating around its center as it's orbiting the center of the sketch?



```
x = R cos(theta) + w/2
y = R sin(theta) + h/2
translate(w/2, h/2);
(2) { rotate(theta)
      translate(x, y); }
(1) { rotate(theta)
      box(10);
      theta += inc }
```



References

- For today's Processing experiments see
- https://processing.org/tutorials/p3d/
- https://processing.org/tutorials/transform2d/
- And Week4 module on Elms will have some examples