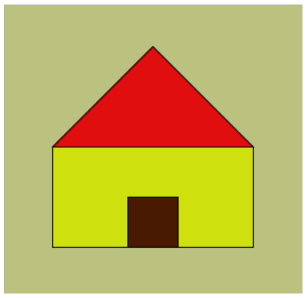
Simple Program Structure

```
// Create and set canvas
size(width, height);
smooth();
background(color);

// Draw something
…
// Draw something else
…
// etc.
```

GXK2013 1

Simple Program Structure

```
// Draw a simple house
// Create and set canvas

size(300, 300);
smooth();
background(187, 193, 127);

// wall
fill(206, 224, 14);
rect(50, 150, 200, 100);

// Draw Door
fill(72, 26, 2);
rect(125, 200, 50, 50);

// Draw roof
fill(224, 14, 14);
triangle(50, 150, 150, 50, 250, 150);
```



GXK2013 2

Program Structure: Dynamic Mode

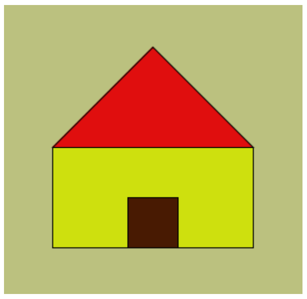
Most Processing programs we will write will have the following structure:

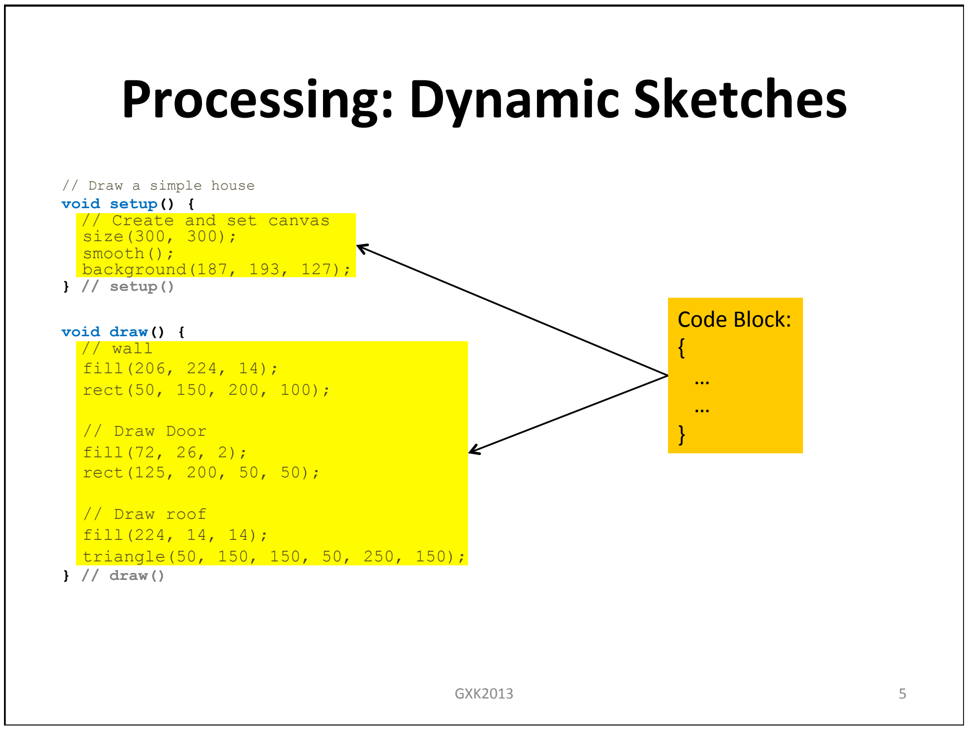
```
<Declare variables>

void setup() {

   <initial canvas set up goes here>

} // setup()

void draw() {

   <drawing stuff goes here>

} // draw()
```

Program Structure: Dynamic Mode

Most Processing programs we will write will have the following structure:

```
// Draw a simple house
void setup() {
  // Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()

void draw() {
  // wall
  fill(206, 224, 14);
  rect(50, 150, 200, 100);

  // Draw Door
  fill(72, 26, 2);
  rect(125, 200, 50, 50);

  // Draw roof
  fill(224, 14, 14);
  triangle(50, 150, 150, 50, 250, 150);
} // draw()
```



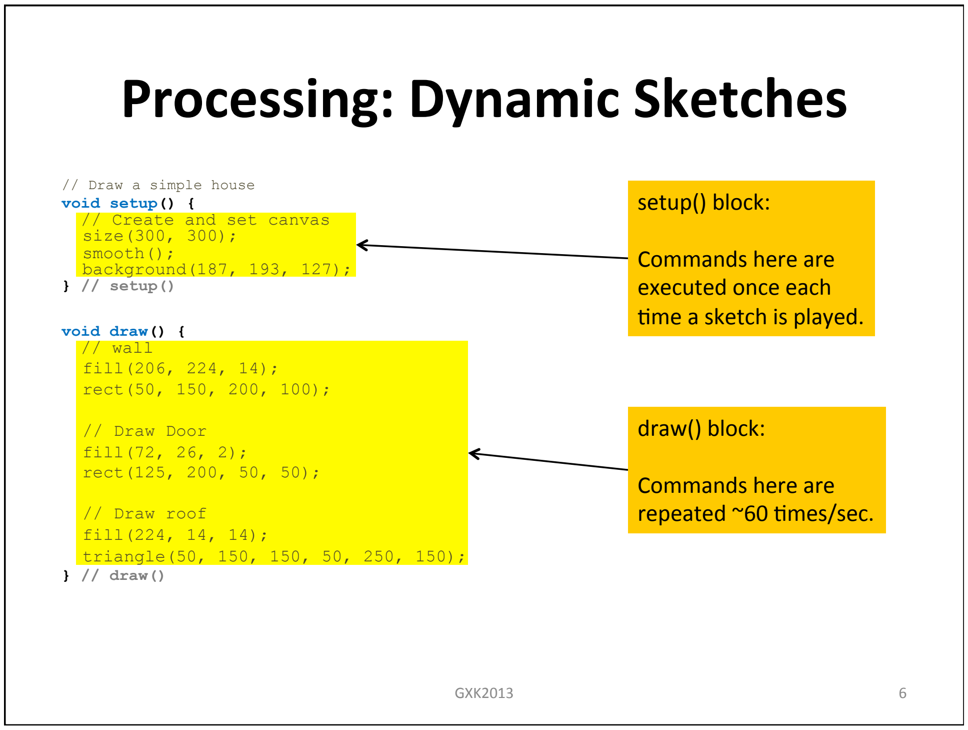


Processing: Dynamic Sketches

```
// Draw a simple house
void setup() {
  // Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()

void draw() {
  // wall
  fill(206, 224, 14);
  rect(50, 150, 200, 100);

  // Draw Door
  fill(72, 26, 2);
  rect(125, 200, 50, 50);

  // Draw roof
  fill(224, 14, 14);
  triangle(50, 150, 150, 50, 250, 150);
} // draw()
```

Code Block:
{
…
…
}

GXK2013 5



Processing: Dynamic Sketches

```
// Draw a simple house
void setup() {
  // Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()

void draw() {
  // wall
  fill(206, 224, 14);
  rect(50, 150, 200, 100);

  // Draw Door
  fill(72, 26, 2);
  rect(125, 200, 50, 50);

  // Draw roof
  fill(224, 14, 14);
  triangle(50, 150, 150, 50, 250, 150);
} // draw()
```

setup() block:

Commands here are executed once each time a sketch is played.

draw() block:

Commands here are repeated ~60 times/sec.

GXK2013 6

Processing: Dynamic Sketches

```
// Draw a simple house
void setup() {
  // Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()

void draw() {
  // wall
  fill(206, 224, 14);
  rect(50, 150, 200, 100);

  // Draw Door
  fill(72, 26, 2);
  rect(125, 200, 50, 50);

  // Draw roof
  fill(224, 14, 14);
  triangle(50, 150, 150, 50, 250, 150);
} // draw()
```

But…

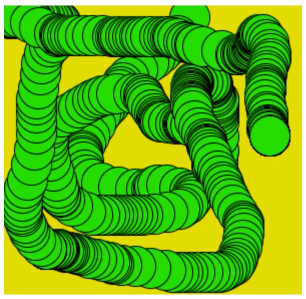
What are these???

For now…
Necessary syntax

More later…

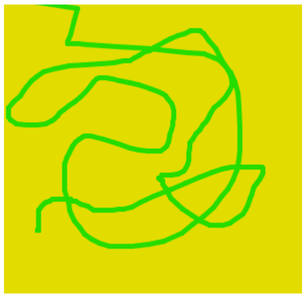
Something More Interesting…

```
color color1 = color(227, 220, 0);
color color2 = color(37, 220, 0);
color color3 = color(0);

void setup() {
  // create and set canvas
  size(300, 300);
  smooth();
  background(color1);
} // setup()


void draw() {
  stroke(color3);
  fill(color2);
  ellipse(mouseX, mouseY, 40, 40);
} // draw()
```



Predefined variables: pmouseX, pmouseY

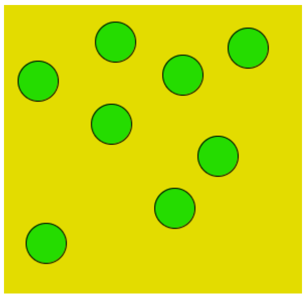
```
color color1 = color(227, 220, 0);
color color2 = color(37, 220, 0);
color color3 = color(0);

void setup() {
  // create and set canvas
  size(300, 300);
  smooth();
  background(color1);
} // setup()


void draw() {
  stroke(color2);
  strokeWeight(5);
  line(pmouseX, pmouseY, mouseX, mouseY);
} // draw()
```



Events: More Interactivity

```
color color1 = color(227, 220, 0);
color color2 = color(37, 220, 0);
color color3 = color(0);

void setup() {
  // create and set canvas
  size(300, 300);
  smooth();
  background(color1);
} // setup()


void draw() {
  // nothing here, but is required
} // draw()

void mousePressed() {
  stroke(color3);
  fill(color2);
  ellipse(mouseX, mouseY, 40, 40);
} // mousePressed()
```

Circles are drawn
ONLY when mouse is pressed.



Something More Interesting…

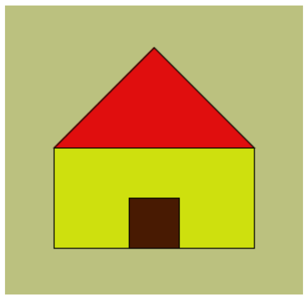
```
color color1 = color(227, 220, 0);
color color2 = color(37, 220, 0);
color color3 = color(0);

void setup() {
  // create and set canvas
  size(300, 300);
  smooth();
  background(color1);
} // setup()


void draw() {
  stroke(color3);
  fill(color2);
  ellipse(mouseX, mouseY, 40, 40);
} // draw()
```

What happens when…

You move the `background(…)` command to draw()?

Redo: A Better House Sketch

```
// Draw a simple house
int houseX = 50;                 // bottom left corner of house
int houseY = 250;

int houseHeight = 200;           // overall width and height of house
int houseWidth = 200;

int wallHeight = houseHeight/2;  // height of wall is 1/2 of house height
int roofHeight = houseHeight/2;
int doorHeight = houseHeight/4;
int doorWidth = houseWidth/4;

void setup() {
  // Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()

void draw() {
  // wall
  fill(206, 224, 14);
  rect(houseX, houseY - wallHeight,
       houseWidth, wallHeight);

  // Draw Door
  fill(72, 26, 2);
  rect(houseX + houseWidth/2 - doorWidth/2, houseY-doorHeight,
       doorWidth, doorHeight);

  // Draw roof
  fill(224, 14, 14);
  triangle(houseX, houseY - wallHeight,
           houseX+houseWidth/2, houseY-houseHeight,
           houseX+houseWidth, houseY-wallHeight);
} // draw()
```



Controlling Frame Rate

```
frameRate(N);
```
Changes frame rate to N times/ second

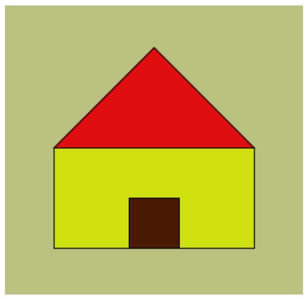
```
<Declare variables>

void setup() {
   …
   frameRate(30);
} // setup()

void draw() {

   <drawing stuff goes here>

} // draw()
```

```
noLoop()
```
Controls the use of frame rate.

```
<Declare variables>

void setup() {
   …
   noLoop();
} // setup()

void draw() {

   <drawing stuff goes here>

} // draw()
```

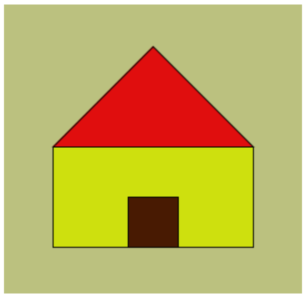
House() function example

```
// Draw a simple house

void setup() {// Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()


void draw() {
  // Draw a house at 50, 250 in 200x200 pixels
  house(50, 250, 200, 200);
} // draw()

void house(int houseX, int houseY, int houseWidth, int houseHeight) {
  // Draw a house at <houseX, houseY> (bottom left corner)
  // with width houseWidth and height houseHeight

  int wallHeight = houseHeight/2;  // height of wall is 1/2 of house height
  int roofHeight = houseHeight/2;
  int doorHeight = houseHeight/4;
  int doorWidth = houseWidth/4;

  // wall
  fill(206, 224, 14);
  rect(houseX, houseY - wallHeight, houseWidth, wallHeight);

  // Draw Door
  fill(72, 26, 2);
  rect(houseX + houseWidth/2 - doorWidth/2, houseY-doorHeight, doorWidth, doorHeight);

  // Draw roof
  fill(224, 14, 14);
  triangle(houseX, houseY - wallHeight, houseX+houseWidth/2, houseY-houseHeight, houseX+houseWidth, houseY-wallHeight);
} // house()
```



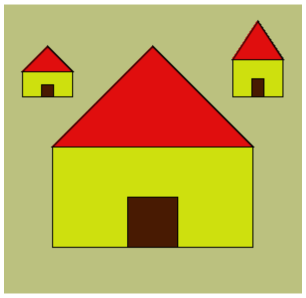
House() function example

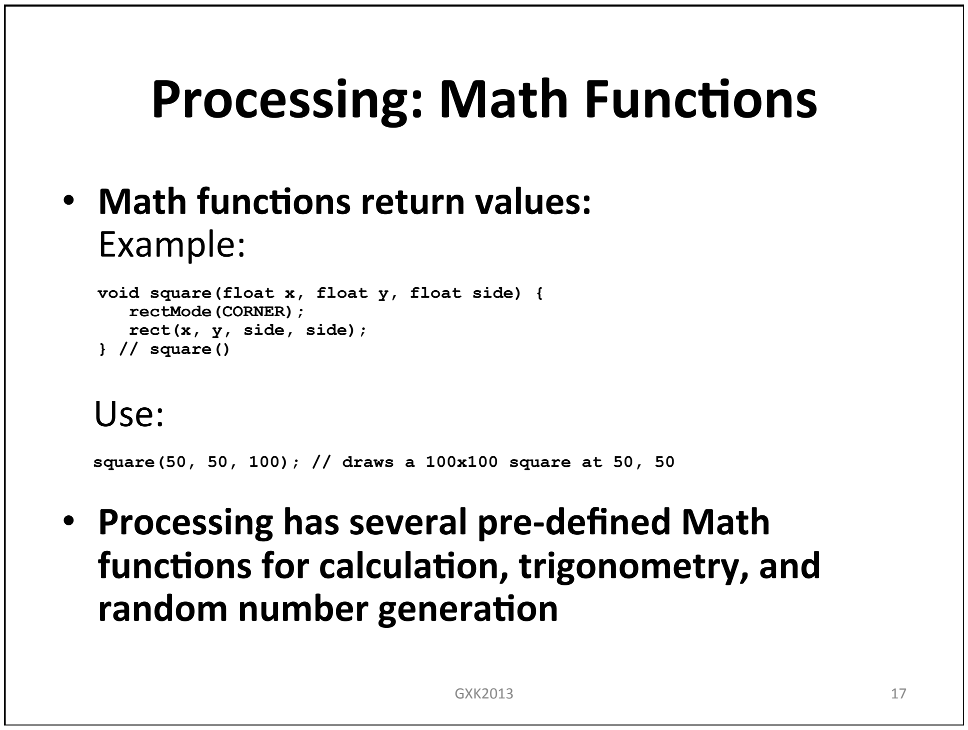
```
// Draw a simple house

void setup() {// Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()


void draw() {
  // Draw a house at 50, 250 in 200x200 pixels
  house(50, 250, 200, 200);
} // draw()

void house(int houseX, int houseY, int houseWidth, int houseHeight) {
  // Draw a house at <houseX, houseY> (bottom left corner)
  // with width houseWidth and height houseHeight

  int wallHeight = houseHeight/2;  // height of wall is 1/2 of house height
  int roofHeight = houseHeight/2;
  int doorHeight = houseHeight/4;
  int doorWidth = houseWidth/4;

  // wall
  fill(206, 224, 14);
  rect(houseX, houseY - wallHeight, houseWidth, wallHeight);

  // Draw Door
  fill(72, 26, 2);
  rect(houseX + houseWidth/2 - doorWidth/2, houseY-doorHeight, doorWidth, doorHeight);

  // Draw roof
  fill(224, 14, 14);
  triangle(houseX, houseY - wallHeight, houseX+houseWidth/2, houseY-houseHeight, houseX+houseWidth, houseY-wallHeight);
} // house()
```



House() function example

```
// Draw a simple house

void setup() {// Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()

void draw() {
  // Draw a house at 50, 250 in 200x200 pixels
  house(50, 250, 200, 200);
  house(20, 100, 50, 50);
  house(230, 100, 50, 75);
} // draw()

void house(int houseX, int houseY, int houseWidth, int houseHeight) {
  // Draw a house at <houseX, houseY> (bottom left corner)
  // with width houseWidth and height houseHeight

  ...

} // house()
```





Processing: Math Functions

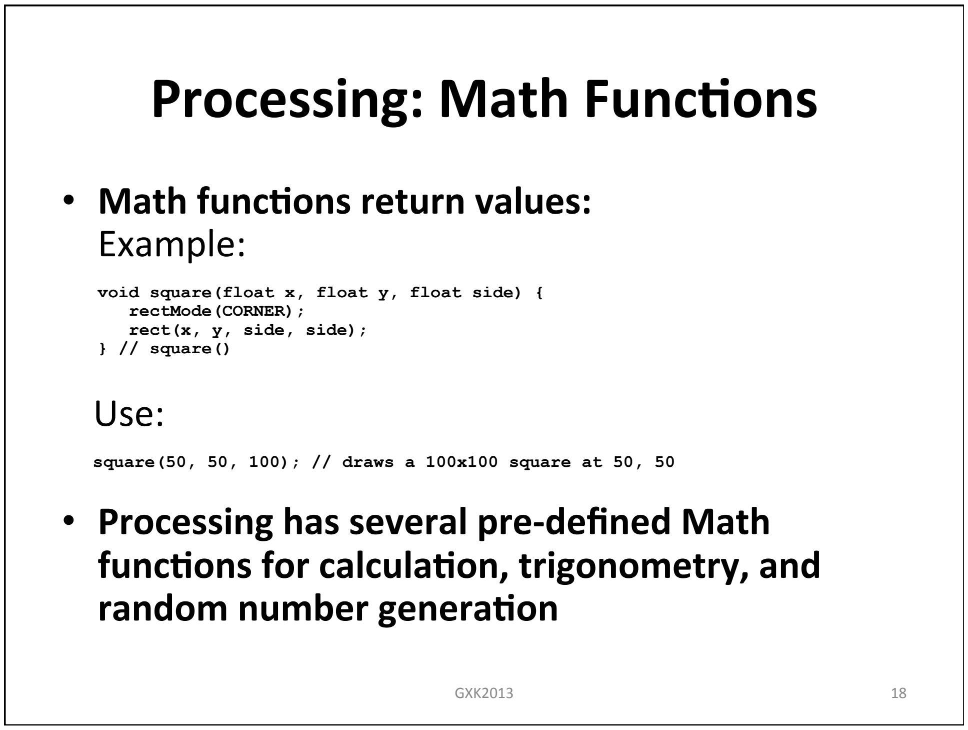
- **Math functions return values:**
  Example:

```
void square(float x, float y, float side) {
   rectMode(CORNER);
   rect(x, y, side, side);
} // square()
```

  Use:

```
square(50, 50, 100); // draws a 100x100 square at 50, 50
```

- **Processing has several pre-defined Math functions for calculation, trigonometry, and random number generation**



Processing: Math Functions

- **Math functions return values:**
  Example:

```
void square(float x, float y, float side) {
   rectMode(CORNER);
   rect(x, y, side, side);
} // square()
```

  Use:

```
square(50, 50, 100); // draws a 100x100 square at 50, 50
```

- **Processing has several pre-defined Math functions for calculation, trigonometry, and random number generation**

Processing: Pre-defined Math Functions

- **Calculation**
  abs(), ceil(), constrain(), dist(), exp(), floor(), lerp()
  log(), mag(), map(), max(), min(), norm(), pow()
  round(), sq(), sqrt()
- **Trigonometry**
  acos(), asin(), atan(), atan2(), cos(), degrees(),
  radians(), sin(), tan()
- **Random**
  noise(), noiseDetail(), noiseSeed(), random(),
  randomGaussian(), randomSeed()

Math Functions: Examples

- **Calculation**

```
float x, y;
y = 42;
x = sqrt(y);
```

- **Trigonometry**

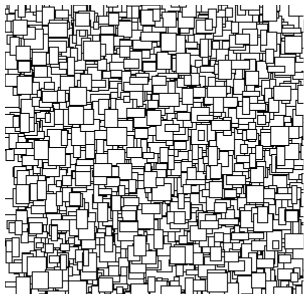
```
float rad = radians(180);
float deg = degrees(PI/4);
```

- **Random**

```
float x = random(10);     // returns a random number [0.0..10.0)
float y = random(1, 6);   // returns a random number [1.0, 6.0)
int ix = int(random(10)); // returns a random number [0..10)
int iy = int(random(1, 6));// returns a random number [1..6)
```

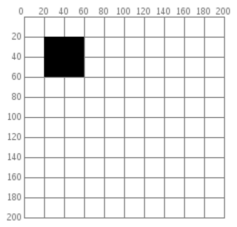
Example: Using random()

```
void setup() {// Create and set canvas
  size(300, 300);
  smooth();
  background(255);
} // setup()

void draw() {
  stroke(0);
  rect(random(width),
       random(height),
       random(5, 20),
       random(5, 20));
} // draw();
```



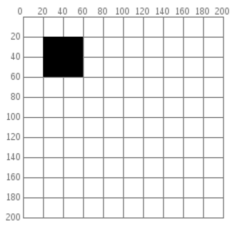
2D Transformations: Translate

rect(20, 20, 40, 40);

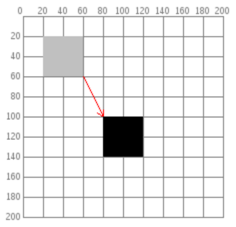


2D Transformations: Translate

rect(20, 20, 40, 40);

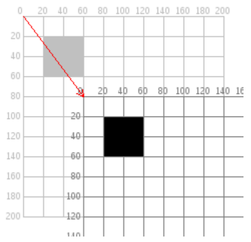


rect(20+60, 20+80, 40, 40);



2D Transformations: Translate

translate(60, 80);
rect(20, 20, 40, 40);



Preserving Context

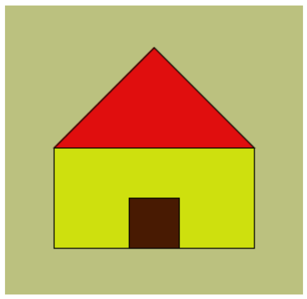
- **translate()** will change the coordinate system for the entire duration of the draw() cycle. It resets at each cycle.
- Use **pushMatrix()** and **popMatrix()** to preserve context during a draw() cycle. i.e.

```
pushMatrix();
translate(<x>, <y>);
<Do something in the new coordinate context>
popMatrix();
```

GXK2013 25

Example: House() again!

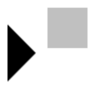
```
// Draw a simple house

void setup() {// Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()


void draw() {
  // Draw a house at 50, 250 in 200x200 pixels
  house(50, 250, 200, 200);
} // draw()

void house(int houseX, int houseY, int houseWidth, int houseHeight) {
  // Draw a house at <houseX, houseY> (bottom left corner)
  // with width houseWidth and height houseHeight

  int wallHeight = houseHeight/2;  // height of wall is 1/2 of house height
  int roofHeight = houseHeight/2;
  int doorHeight = houseHeight/4;
  int doorWidth = houseWidth/4;

  pushMatrix();
  translate(houseX, houseY);
  // wall
  fill(206, 224, 14);
  rect(0, -wallHeight, houseWidth, wallHeight);

  // Draw Door
  fill(72, 26, 2);
  rect(houseWidth/2 - doorWidth/2, -doorHeight, doorWidth, doorHeight);

  // Draw roof
  fill(224, 14, 14);
  triangle(0, -wallHeight, houseWidth/2, -houseHeight, houseWidth, -wallHeight);
  popMatrix();
} // house()
```

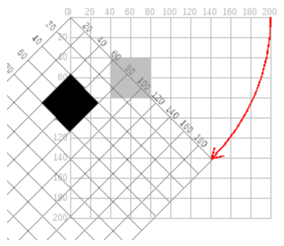


GXK2013 26

2D Transformations: Rotate

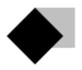
```
void setup() {
  size(200, 200);
  background(255);
  smooth();
  fill(192);
  noStroke();

  rect(40, 40, 40, 40);

  pushMatrix();
  rotate(radians(45));
  fill(0);
  rect(40, 40, 40, 40);
  popMatrix();
} // setup()
```

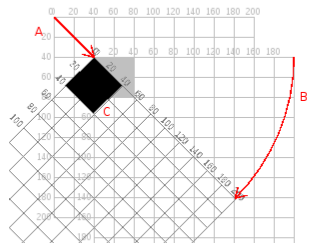




2D Transformations: Rotate

```
void setup() {
  size(200, 200);
  background(255);
  smooth();
  fill(192);
  noStroke();

  rect(40, 40, 40, 40);

  pushMatrix(); // move the origin to the pivot point
  translate(40, 40); // then pivot the grid
  rotate(radians(45)); // and draw the square at the origin
  fill(0);
  rect(0, 0, 40, 40);
  popMatrix();
} // setup()
```





2D Transformations: Scaling

```
void setup() {
  size(200,200);
  background(255);

  stroke(128);
  rect(20, 20, 40, 40);

  stroke(0);
  pushMatrix();
  scale(2.0);
  rect(20, 20, 40, 40);
  popMatrix();
} //setup()
```