

Graphics with Processing

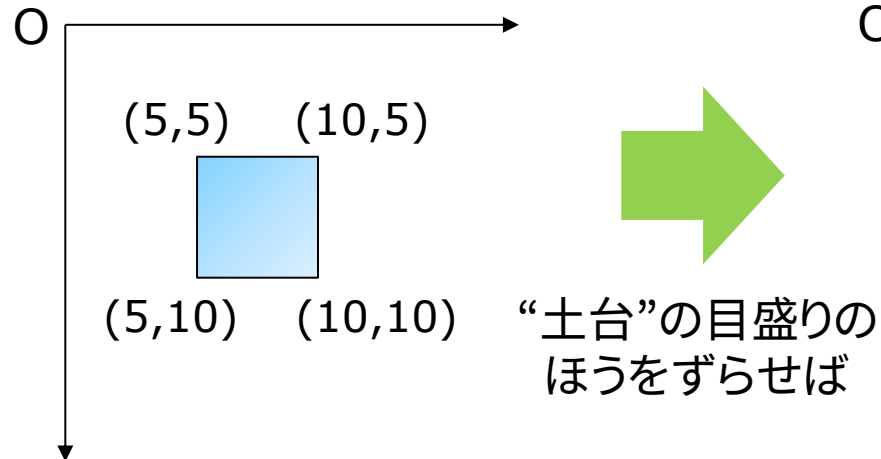
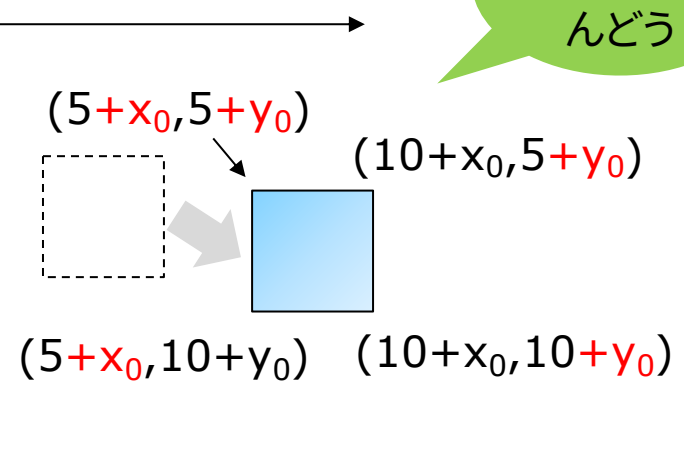
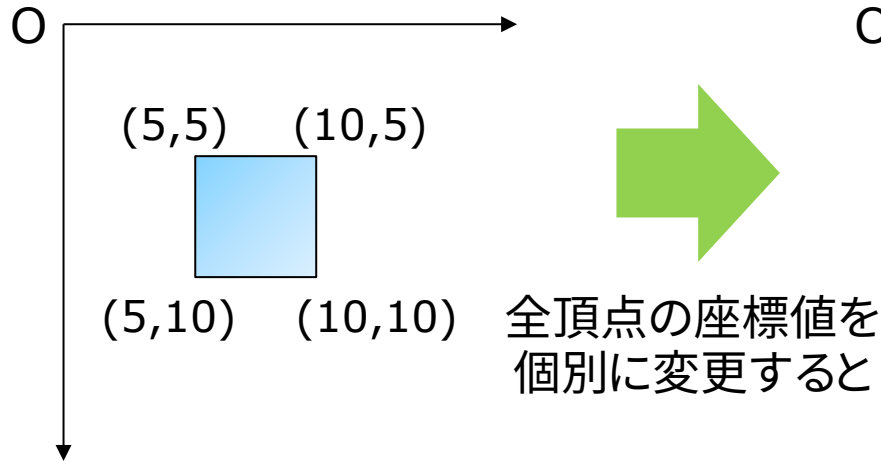


2019-06 座標変換と同次座標

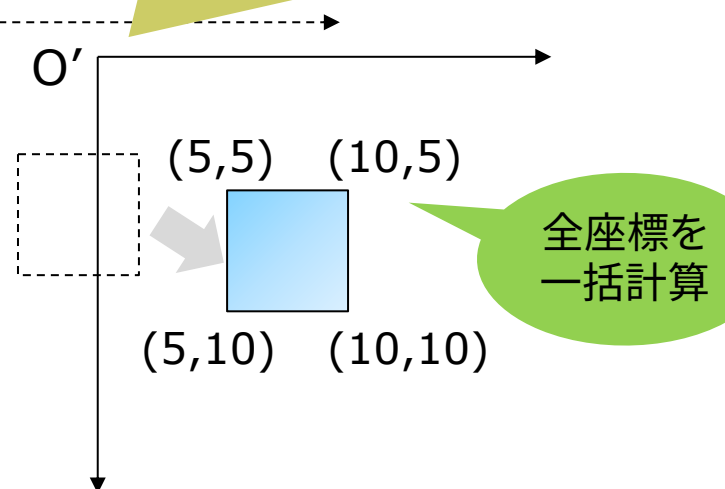
<http://vilab.org>

塩澤秀和

6.0 図形の移動や変形



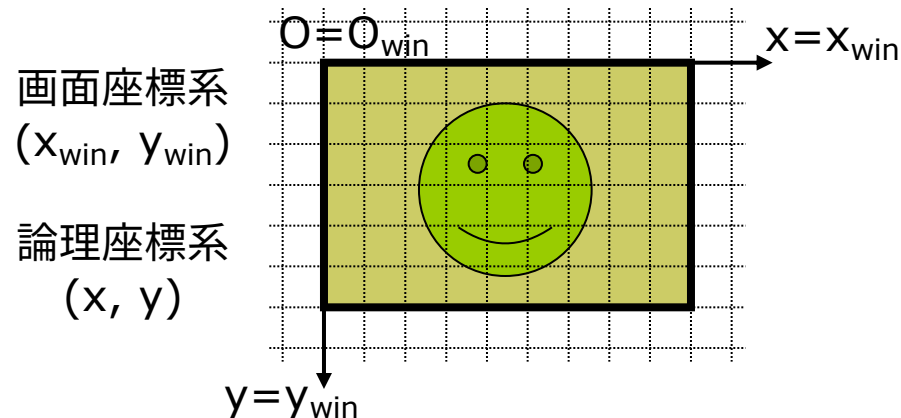
(x_0, y_0) を新しい(0, 0)に付け替える



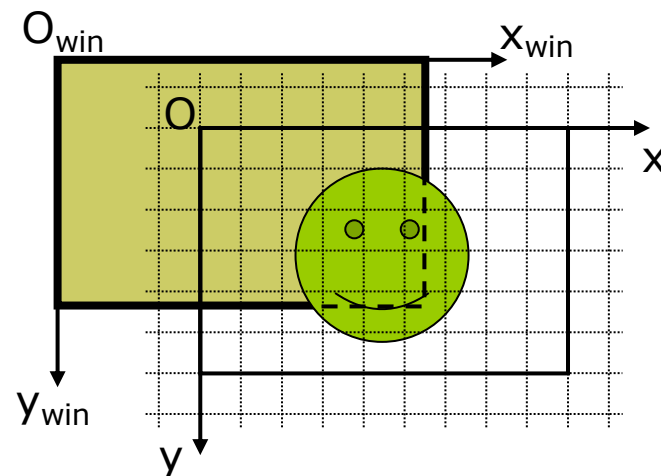
6.1* 座標系

座標系の利用

- 図形の移動
 - 図形を別の位置に表示するとき、全座標の変更を書くのは面倒
⇒ 座標の“変換”でまとめて計算
- 座標系=目盛りのつけ方
 - 原点の位置
 - x軸とy軸の方向
 - x軸とy軸の目盛りの刻み
- 画面座標系
 - 左上隅を(0,0)とする初期状態
 - ウィンドウ内のピクセル位置
- 論理座標系
 - 描画命令で使うためのxy座標
 - 画面座標系とは違う描きやすい目盛りのつけ方に変更できる



初期状態(画面座標系=論理座標系)



描画用の原点をずらした例

6.2* 座標変換 (p.22)

座標変換と幾何変換

□ 座標変換

- 同じ点を別の座標系の値で表す
- または, 同じ座標系の中で点を移動すると考えてもよい(6.14)

$$\begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{f} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

□ CGにおける座標変換

- 論理座標(描画命令の座標値)
→ 画面座標(ピクセル位置)
- 図形の各点に対応する画面のピクセル位置をまとめて計算
- 3Dの場合, 遠近法の計算等のために何段階も組み合わせる

幾何変換(幾何学的変換)

□ 平行移動

$$x' = x + x_0$$

$$y' = y + y_0$$

例) 原点を画面の(10, 20)に移動し, 座標(5, 7)に点を打つと, 画面では(15, 27)に表示

□ 拡大・縮小

$$x' = \alpha x$$

$$y' = \beta y$$

例) 目盛りを横2倍, 縦3倍に拡大して, 座標(5, 3)に点を打つと, 画面では(10, 9)に表示

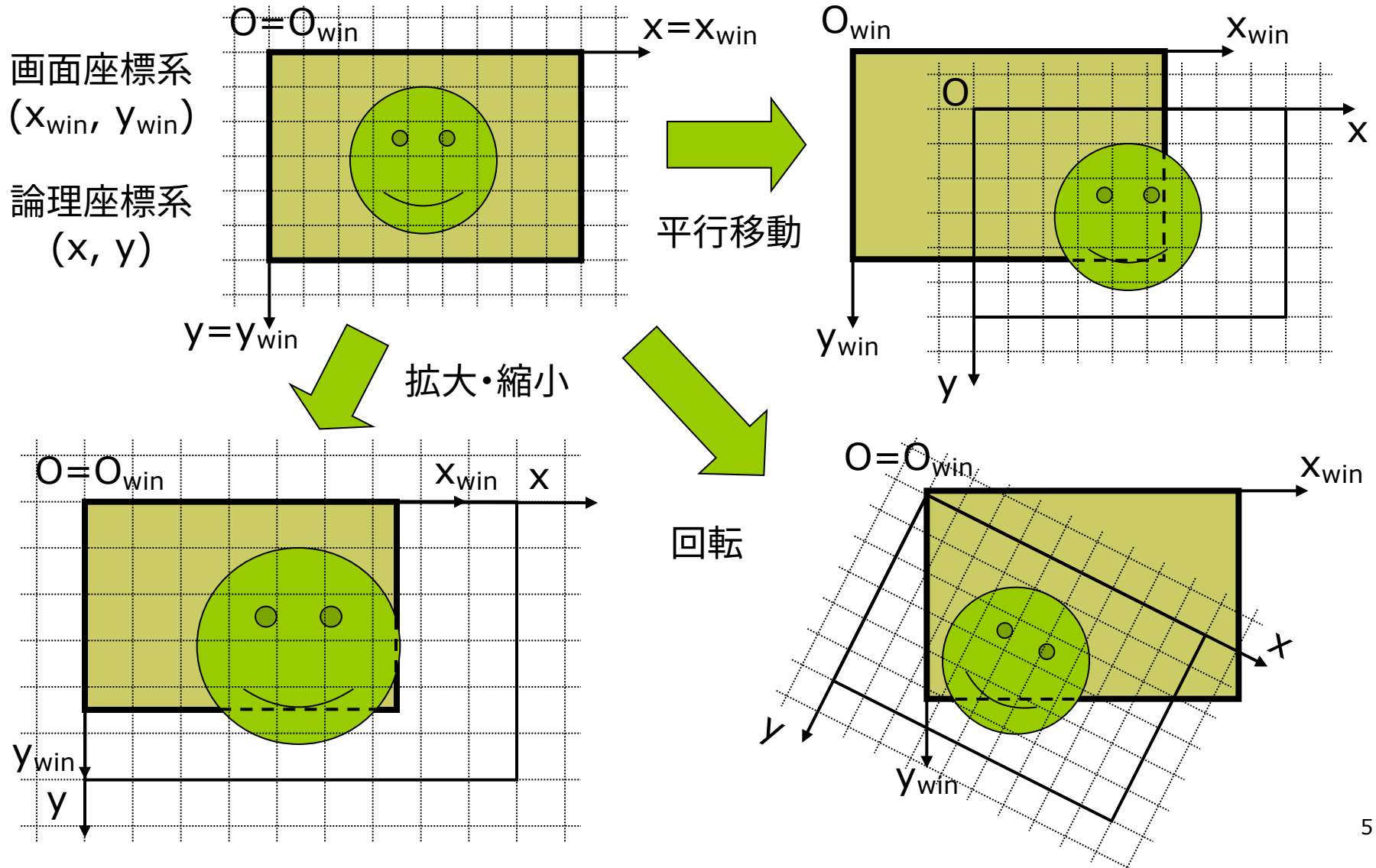
□ 回転

$$x' = x \cos \theta - y \sin \theta$$

$$y' = x \sin \theta + y \cos \theta$$

導出方法は6.13参照

6.3* 幾何変換の効果



6.4 幾何変換関数

幾何変換関数

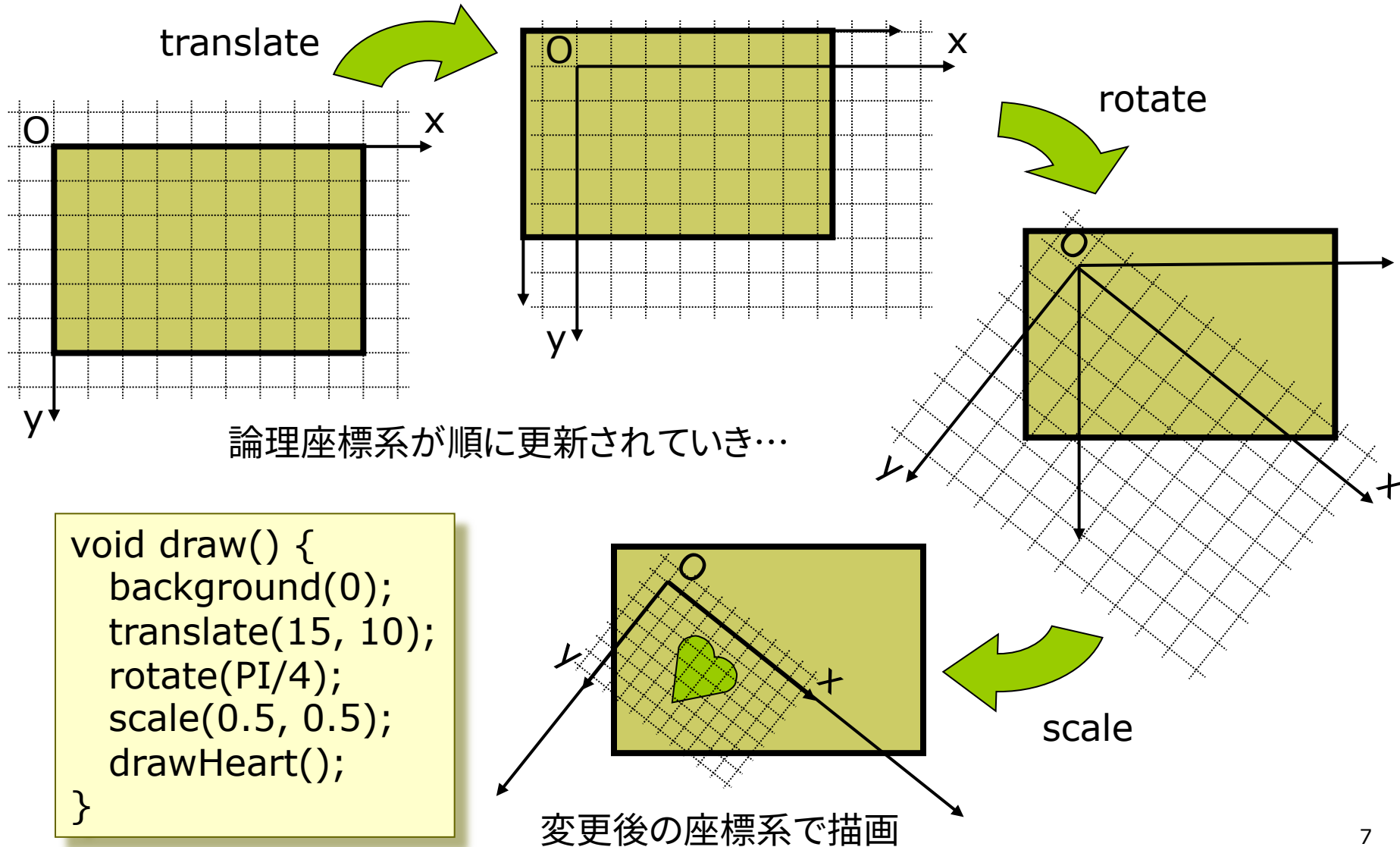
- `translate(x0, y0)`
 - 座標系を平行移動(原点を移動)
 - x軸方向に x_0 移動
 - y軸方向に y_0 移動
 - Processingではy軸は**下向き**
- `scale(α , β)`
 - 座標系を拡大または縮小
 - x軸方向(左右)に α 倍
 - y軸方向(上下)に β 倍
 - 原点が中心に全体が拡大
- `rotate(θ)`
 - 座標系を回転
 - 原点中心に θ ラジアン回転
 - Processingで+方向は**時計回り**

```
float bai = 1.0; // 拡大率

void setup() {
  size(400, 400);
  rectMode(CENTER);
  frameRate(30);
}

void draw() {
  background(255);
  // (200,200)を新しい原点に設定
  translate(200, 200);
  scale(bai);
  strokeWeight(10);
  fill(128, 128, 255);
  rect(0, 0, 50, 50);
  bai += 0.02;
  if (bai > 8.0) bai = 1.0;
}
```

6.5* 幾何変換の合成 (p.22)



6.6* 同次座標表現 (p.19)

それぞれ展開し
6.2の式と比較
して確かめよ

同次座標表現

- 座標の一括計算に適した形式

2次元直交座標 2次元同次座標

$$(x, y) \Leftrightarrow (x, y, 1)$$

同じ座標

- 同次座標表現による座標変換

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

行列の掛け算ですべての幾何変換を表せる

幾何変換行列

- 平行移動

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- 拡大・縮小

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- 回転

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

6.7* 合成変換行列 (p.28)

座標変換の合成

- 座標変換を積み重ねる

$$P_{win} = M_1 M_2 M_3 \cdots M_n P$$

$$M = M_1 M_2 M_3 \cdots M_n$$

何段階もの変換をまとめた行列

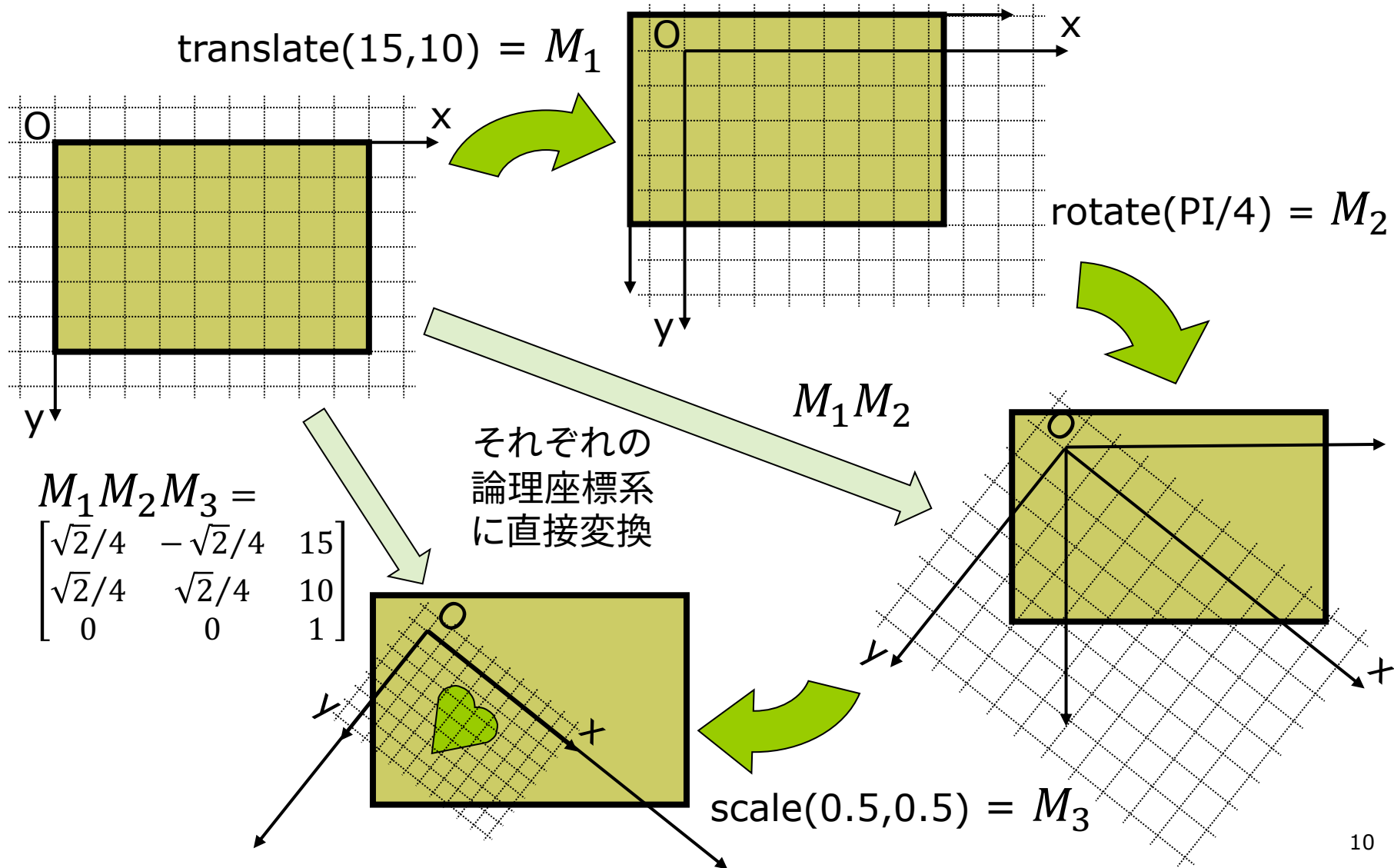
```
void draw() {
  background(0);
  translate(15, 10); // 変換 M1
  rotate(PI/4);     // 変換 M2
  scale(0.5, 0.5);  // 変換 M3
  drawHeart();
}
```

- 右上の例の合成変換行列

$$\begin{bmatrix} x_{win} \\ y_{win} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 15 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) & 0 \\ \sin(\pi/4) & \cos(\pi/4) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_{win} \\ y_{win} \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2}/4 & -\sqrt{2}/4 & 15 \\ \sqrt{2}/4 & \sqrt{2}/4 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \therefore M = \begin{bmatrix} \sqrt{2}/4 & -\sqrt{2}/4 & 15 \\ \sqrt{2}/4 & \sqrt{2}/4 & 10 \\ 0 & 0 & 1 \end{bmatrix}$$

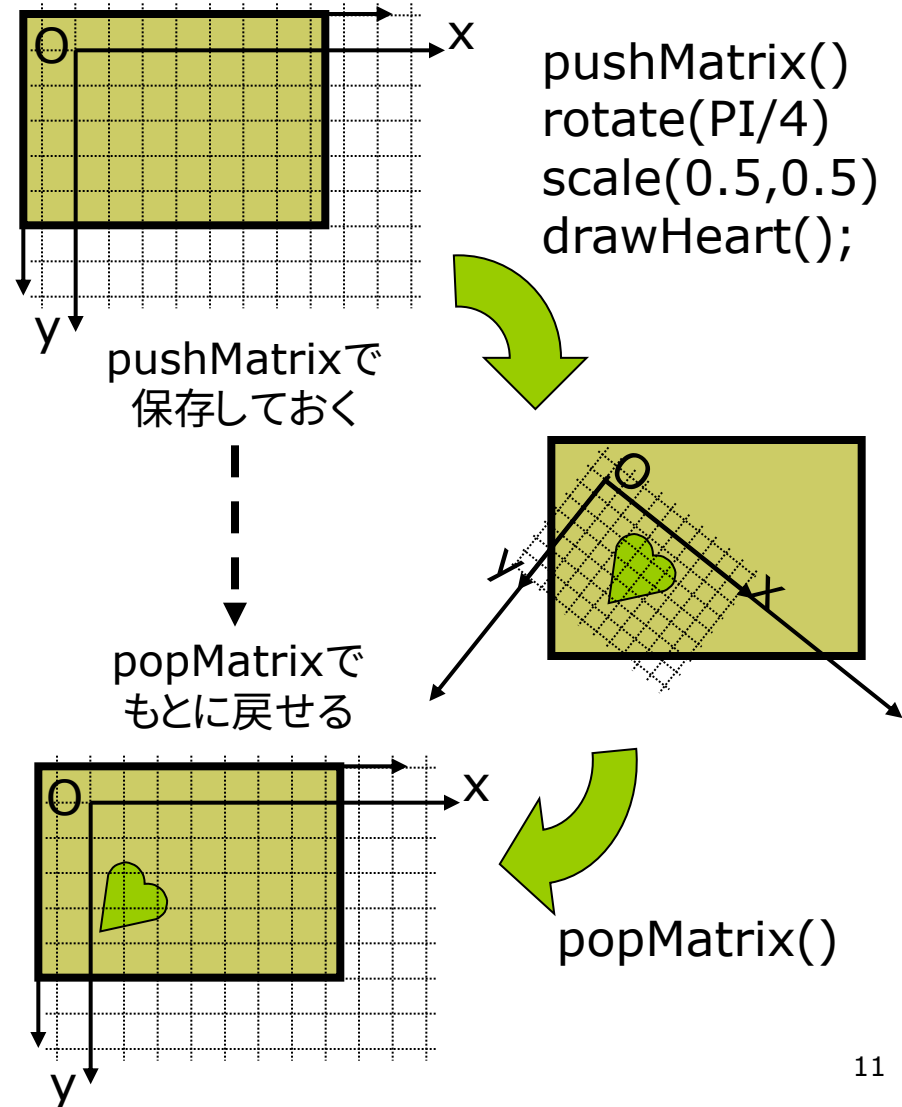
6.8* 合成変換行列の意味



6.8 変換行列の操作 (p.54)

行列スタックの操作

- システム変換行列
 - 現在の描画命令で使用される合成変換行列=論理座標系
 - 幾何変換 (translate, rotate, scale) のたびに合成されていく
- pushMatrix()
 - システム変換行列 (現在の論理座標系) を一時的に保存する
- popMatrix()
 - 最近保存した変換行列を戻す
 - pushMatrix() と必ず対にする
- resetMatrix()
 - システム変換行列を初期化し, 保存した行列も全て破棄する
 - 画面座標系=論理座標系になる



6.9 行列操作の例

```
void setup() {
  size(600, 400);
  rectMode(CENTER);
}

void draw() {
  background(#8080e0);
  pushMatrix();
  translate(mouseX, mouseY);
  rotate(radians(frameCount));
  fill(#ffd0d0); rect(0,0, 50, 50);
  popMatrix();
  // ここで座標系が初期状態に戻る
  pushMatrix();
  translate(width/2, height/2);
  rotate(-radians(frameCount));
  fill(#ffff00); rect(0,0, 50, 50);
  popMatrix();
}
```

```
void setup() {
  size(600, 400);
}

void draw() {
  background(128);
  float a = radians(frameCount);
  translate(width/2, height/2);
  pushMatrix();
  rotate(a);
  translate(100, 0);
  ellipse(0, 0, 40, 40);
  pushMatrix(); // 入れ子にできる
  rotate(4 * a);
  translate(50, 0);
  scale(cos(a));
  ellipse(0, 0, 40, 40);
  popMatrix();
  popMatrix();
  scale(sin(a));
  ellipse(0, 0, 40, 40);
}
```

6.10* 演習課題

課題

- 6.11はスマイリー(にこちゃん)を2つ描画するプログラムである

問1) 中心と外側の顔の描画位置を決めている**合成変換行列**($M_{\text{中心}}$ と $M_{\text{外側}}$)の**両方**を求めなさい

- $M_{\text{中心}}$ は右のヒント参照
- 式は6.15を参考に簡単化せよ
- 次回,**A4レポート用紙**で提出

問2) このプログラムに幾何変換の関数を2つ加えて,外側の顔の大きさを半分にして,顔の向きは回転しないようにしなさい

- ただし,顔を描画する関数は,変更したり追加したりしないこと
- プログラムと画面印刷を提出

問1の $M_{\text{中心}}$ のヒント

- $M_{\text{中心}}$ は次の2つの変換の合成
 - $M_1 = \text{translate}(200, 200)$
 - $M_2 = \text{rotate}(-a)$
- それぞれの行列表現は

$$M_1 = \begin{bmatrix} 1 & 0 & 200 \\ 0 & 1 & 200 \\ 0 & 0 & 1 \end{bmatrix}$$

カッコ内のマイナスは外す(6.15)

$$M_2 = \begin{bmatrix} \cos(-a) & -\sin(-a) & 0 \\ \sin(-a) & \cos(-a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- $M_{\text{中心}}$ はこの2つの合成なので

$$M = \begin{bmatrix} 1 & 0 & 200 \\ 0 & 1 & 200 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-a) & -\sin(-a) & 0 \\ \sin(-a) & \cos(-a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

6.11 演習課題 (続き)

```
void setup() {
  size(400, 400);
  frameRate(30);
}

void drawSmiley() {
  ellipseMode(CENTER);
  strokeWeight(3);
  stroke(0); fill(#ffff00);
  ellipse(0, 0, 100, 100);
  noStroke(); fill(0);
  ellipse(-15, -15, 12, 12);
  ellipse( 15, -15, 12, 12);
  stroke(#ff0000); noFill();
  bezier(-25, 20, -10, 35,
         10, 35, 25, 20);
}
```

```
void draw() {
  float a = radians(frameCount);
  background(255);
  translate(200, 200); // 原点移動
  // ★
  pushMatrix();
  rotate(-a);
  drawSmiley();
  popMatrix();
  // ★
  pushMatrix();
  rotate(-a);
  translate(130, 0);
  // ここに2つ幾何変換を追加する
  drawSmiley();
  popMatrix();
  // ★
}
```

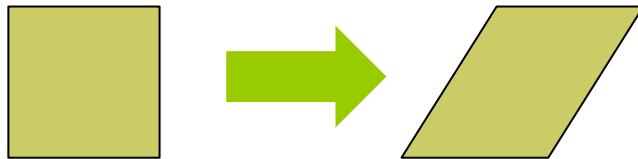
★のところ
の座標系は
同じになる

6.12 参考：せん断と鏡映 (p.26)

せん(剪)断/スキュー/シアー

□ 斜めにゆがめる変換

- 座標系を平行四辺形にゆがめる
- 変換後も平行関係は保たれる



□ shearX(角度)

- x軸方向のせん断
- x軸より上は左に, x軸より下は右にずれていくように歪める
- y軸を指定の角度だけ傾ける

$$\begin{aligned} x' &= x + a y \\ y' &= y \end{aligned} \quad (a = \tan \theta)$$

□ shearY(角度)

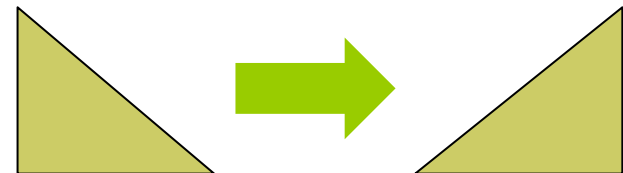
- y軸方向のせん断

$$\begin{aligned} x' &= x \\ y' &= b x + y \end{aligned} \quad (b = \tan \theta)$$

鏡映(反転)

□ 負の拡大縮小変換

- x軸またはy軸を基準に反転
- 例) scale(-1, 1)



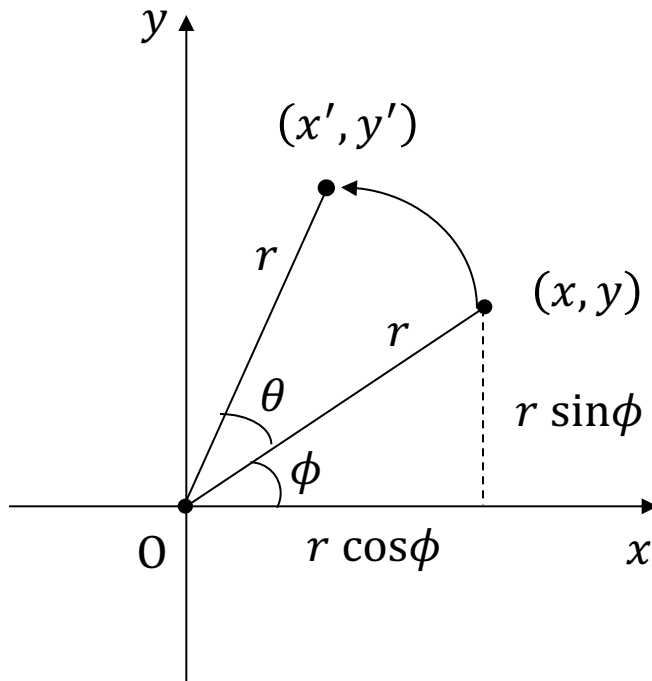
図の例の
変換式

$$\begin{aligned} x' &= (-1) \cdot x \\ y' &= y \end{aligned}$$

6.13 参考：回転行列の導出

初期位置 (x, y) θ 回転後 (x', y')

$$\begin{cases} x = r \cos \phi \\ y = r \sin \phi \end{cases} \quad \begin{cases} x' = r \cos(\phi + \theta) \\ y' = r \sin(\phi + \theta) \end{cases}$$



展開計算(加法定理)

$$\begin{aligned} x' &= r (\cos \phi \cos \theta - \sin \phi \sin \theta) \\ &= r \cos \phi \cdot \cos \theta - r \sin \phi \cdot \sin \theta \\ &= x \cos \theta - y \sin \theta \end{aligned}$$

$$\begin{aligned} y' &= r (\sin \phi \cos \theta + \cos \phi \sin \theta) \\ &= r \sin \phi \cdot \cos \theta + r \cos \phi \cdot \sin \theta \\ &= y \cos \theta + x \sin \theta \\ &= x \sin \theta + y \cos \theta \end{aligned}$$

行列形式

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

6.14 参考：図形移動での考え方 (p.29)

合成変換 (6.7) の2通りの解釈

□ 座標系全体が更新 (6.5)

- 左から行列計算

$$P_{win} = ((M_1 M_2) M_3) P$$

- 座標系 (全体) に変換が適用されていき, 新しい座標系の中で図形が描画される

□ 図形の頂点が移動 (右図)

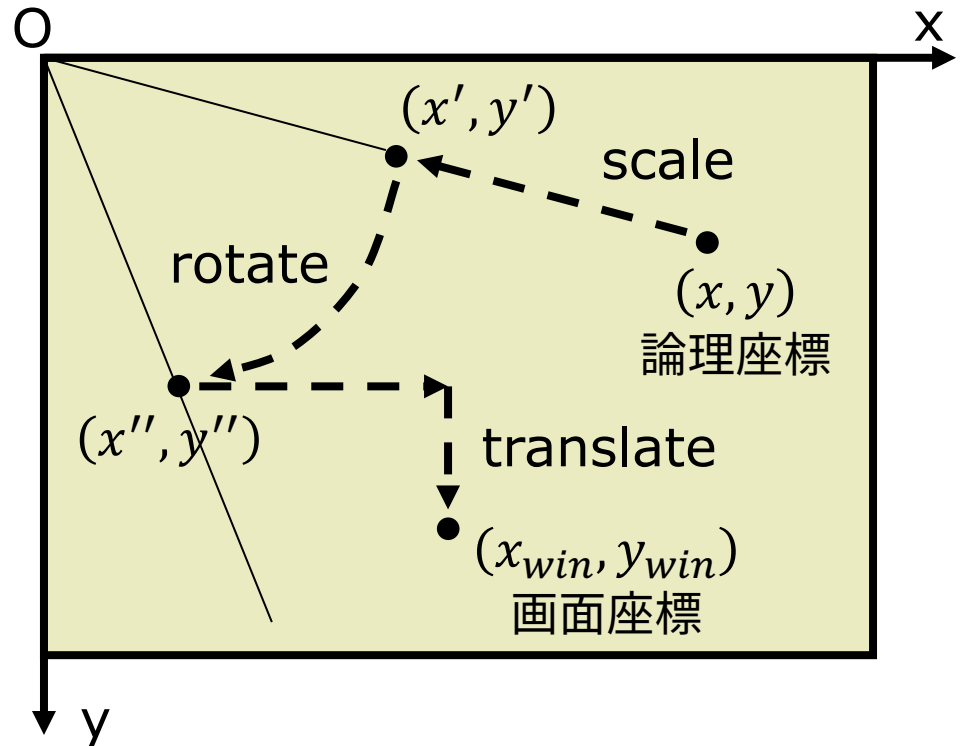
- 右から行列計算

$$P_{win} = M_1 (M_2 (M_3 P))$$

- 座標系は変わらず, その中で図形の座標に変換が適用されていき, 描画位置が決まる

□ 数学的には等価

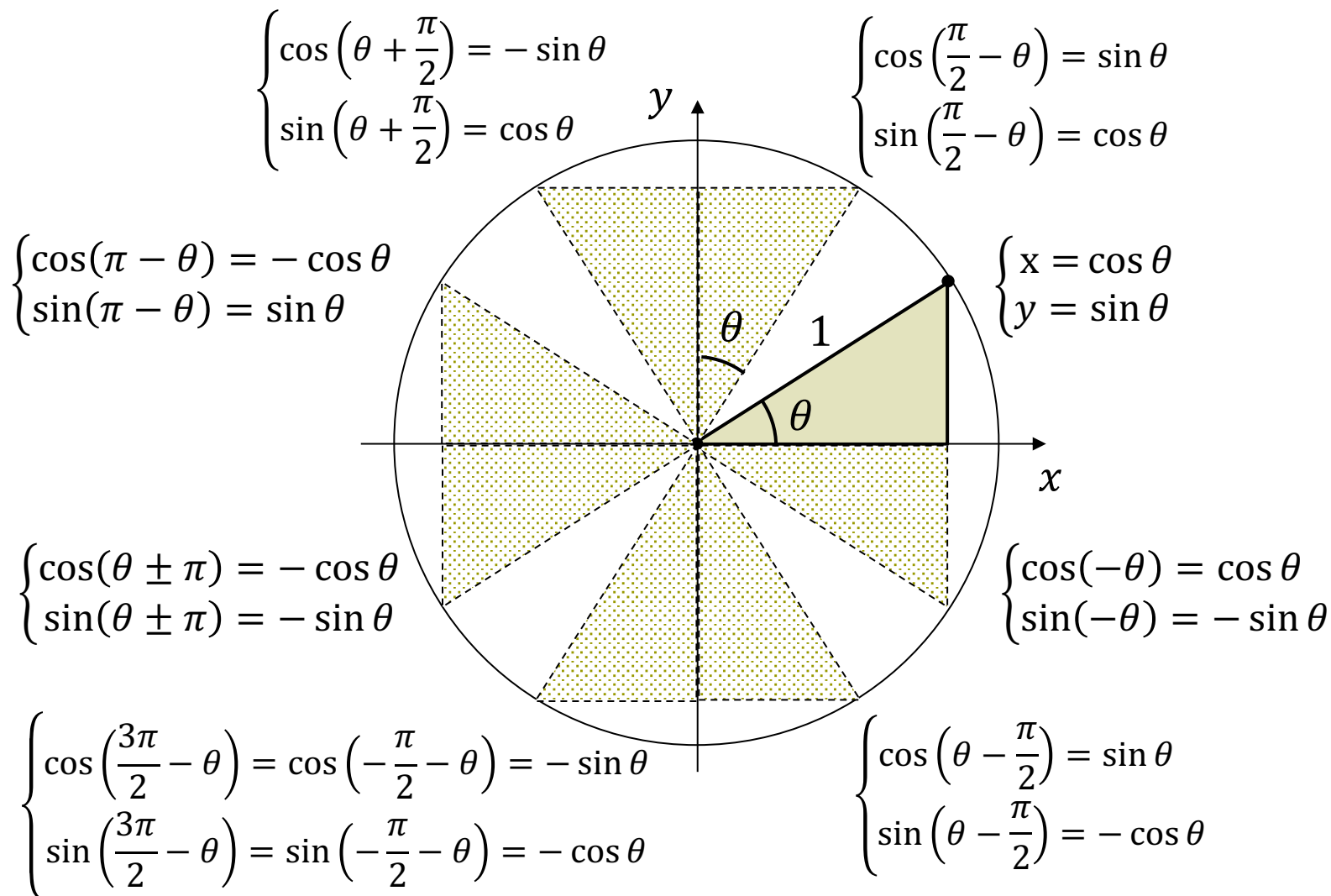
- 後者の考え方の場合, プログラムでは下から変換を考える



```
translate(15, 10);
rotate(PI/4);
scale(0.5, 0.5);
point(x, y);
```

変換を下から考える

6.15 参考：三角関数の関係式



6.16 参考：一次変換とアフィン変換

座標変換と幾何変換

□ 座標変換

$$\begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{f} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

□ 座標変換の合成

- いくつかの座標変換の「合成」で
論理座標 → 画面座標を計算

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} \rightarrow \dots \rightarrow \begin{pmatrix} x_{win} \\ y_{win} \end{pmatrix}$$

□ 幾何変換(幾何学的変換)

- 平行移動
- 拡大・縮小
- 回転

ベクトルと行列による表現

□ 1次変換

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

なぜ平行移動は表現できないか？

- 拡大・縮小と回転が表現可能
⇒ 各自, 対応する行列を求めよ

□ アフィン変換

- すべての幾何変換を表現可能

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

$$\Leftrightarrow \begin{cases} x' = ax + by + e \\ y' = cx + dy + f \end{cases}$$

定数項を付加

6.17 参考：行列計算の確認

$$\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + e \\ cx + dy + f \\ 1 \end{bmatrix}$$

i 行目

j 行目

$$\begin{bmatrix} a_1 & b_1 & e_1 \\ c_1 & d_1 & f_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_2 & b_2 & e_2 \\ c_2 & d_2 & f_2 \\ 0 & 0 & 1 \end{bmatrix}$$

左の行列の i 行目と
右の行列の j 列目の内積
= 積の行列の i 行 j 列

$$= \begin{bmatrix} a_1 a_2 + b_1 c_2 & a_1 b_2 + b_1 d_2 & a_1 e_2 + b_1 f_2 + e_1 \\ c_1 a_2 + d_1 c_2 & c_1 b_2 + d_1 d_2 & c_1 e_2 + d_1 f_2 + f_1 \\ 0 & 0 & 1 \end{bmatrix}$$