08
이미지 다루기

08.01
이미지 읽고 쓰기

아직까지 이미지를 정식으로 다루지는 않았지만 이미 여러 번 이미지 파일을 다루었습니다. 이제 정식으로 이미지 파일을 다루는 법을 배워봅시다.

내 컴퓨터에서 이미지 읽기

우선 이미지 파일을 추가합니다. 이미지 파일을 드래그해서 프로세싱 코드 편집창 위로 떨굽시다.

이렇게 이미지 파일을 추가한 이미지를 프로세싱에서 사용하는 데는 두 단계가 필요합니다.

이미지 파일 변수 지정

우선 이미지를 읽어들여야 합니다. 이미지를 불러와 이미지 파일을 지칭하는 변수에 저장을 합니다. 이미지 파일 자체를 변수에 저장하는 것이 아니라 이미지 파일을 참고(reference)하는 방법은 편리하고 여러 군데서 사용됩니다.

이미지 파일을 가리키는 변수는 PImage 타입을 띕니다. 더 정확하게 이야기하자면 PImage 클래스의 인스턴스라고 지칭하는 것이 맞지만 일단은 넘어가기로 합시다. 이것이 무슨 말인지 궁금하시면 조금만 기다려주세요. 객체지향 프로그래밍에 대해 배우게 되면 이 말이 어떤 뜻인지 알

게 될 것입니다. 이미지 파일을 가리키는 변수는 일단 PImage 타입을 가진다고 이해하고 넘어갑시다.

```
PImage img;
```

loadImage() 함수는 이미지 파일의 위치를 입력값을 받아 변수에 저장합니다.

```
PImage img = loadImage("yourImage.jpg");
```

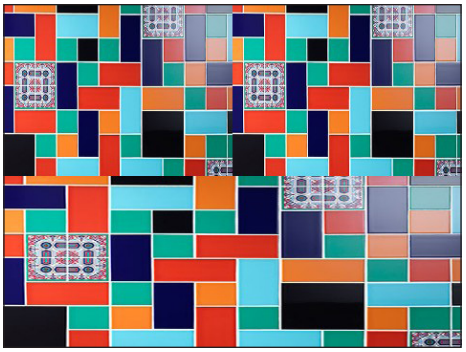
파일의 정확한 위치를 지정하면 내 컴퓨터에 있는 이미지 파일을 어떤 것이나 변수에 저장할 수 있습니다.

화면에 이미지 띄우기

이제 정말 화면에 이미지를 표시할 차례입니다. image() 함수를 이용해 이미지를 화면에 띄울 수 있습니다. 자세한 설명을 하기 전 일단 동작하는 예를 보고 설명을 이어갑시다.

복잡한 기능을 가지고 있는 프로그램을 한 번에 모두 작성하려고 하는 시도는 120%의 확률로 실패합니다. 언제나 작게 조금씩 기능을 추가하고 새로운 라인을 추가하면 자주 제대로 돌아가는지 확인하는 습관을 기르는 것도 좋겠습니다.

```
//myHelloImage
PImage img;
   void setup() {
      size(480, 360);
      img = loadImage("myImage.jpg");
   }

   void draw() {
      image(img, 0, 0, 240, 180);
      image(img, 240, 0, 240, 180);
      image(img, 0, 180, 480, 180);
   }
```



myHelloImage

[원본 링크](https://www.wallsandfloors.co.uk/range/jewel-tone-prismatic-tiles)

image() 파일의 입력값을 살펴봅시다.

```
image(PImage img, float x, float y, float
imageWidth, float imageHeight);
```

표시할 이미지를 정하고, 이미지의 위치를 정합니다. 그리고 이미지의 너비와 높이를 정합니다. 혹시 너비와 높이를 따로 정하지 않으면 화면의 크기인 width와 height를 기본값으로 사용합니다.

웹에서 이미지 읽기

웹에서 이미지를 가져오는 것과 내 컴퓨터에서 이미지 파일을 가져오는 것은 차이가 거의 없습니다. 이미지 파일의 위치를 loadImage()에 입력하는 대신 이미지 파일의 URL주소를 입력해야 하는 차이가 있을 뿐입니다.

```
//myHelloWebImage
String URL = "http://cfile3.uf.tistory.com/
image/222B06505657E2700AD5B9";

PImage img;

void setup() {
   size(740, 1108);
   img = loadImage(URL, "jpg");
}

void draw() {
   image(img, 0, 0);
}
```

myHelloWebImage

이제 이미지 파일 주소만 알면 언제든지 파일을 화면에 띄울 수 있습니다.

08.02
이미지 변형하기

주어진 이미지를 그대로 사용하는데는 한계가 있습니다. 우선 이미지에서 원하는 부분을 떼어내는 것부터 연습합시다. 다음 섹션에서 모자이크를 만들때 바로 써먹을 기술입니다.

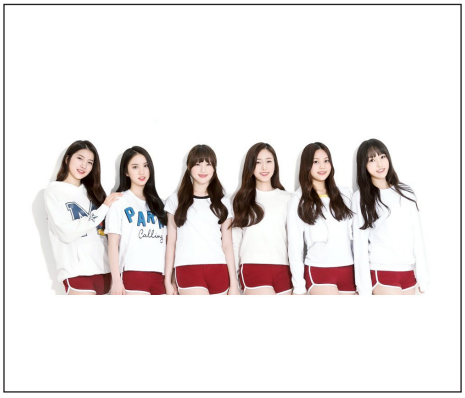
이미지 잘라내기

이전에 설명하기를 프로세싱은 여러 데이터 타입을 가지고 있고 연산되는 방식이 데이터 타입에 따라 다르다고 했습니다. 이미지 타입을 지칭하는 PImage 타입(정확하게 말하면 클래스)도 이미지를 어떻게 연산하는지 미리 정해두었습니다. PImage 타입이 어떻게 연산 혹은 동작하는지를 정한 함수(정확하게는 메서드)를 이용하면 우리가 원하는 작업을 쉽게 할 수 있습니다.

이미지를 잘라내는 작업은 get() 메서드를 사용하면 쉽게 할 수 있습니다. 설명을 이어가기 전 우선 get() 메서드를 이용한 예를 먼저 보겠습니다.

```
//myGetMethod
PImage img;
PImage cropped;

void setup() {
   size(1000, 848);
   imageMode(CENTER);
   img = loadImage("http://nktrend.diskn.com/00_gf/
gf_002.jpg");

   cropped = img.get(0, 0, width, height / 2);
   frameRate(1);
}

void draw() {
   background(255);
   translate(width / 2.0, height / 2.0);
   if (0 == frameCount % 2) {
      image(img, 0, 0);
   }
   else {
      image(cropped, 0, 0);
   }
}
```

myHelloImage



myHelloImage

get 메서드는 가지고 올 이미지의 시작과 끝, 너비와 높이를 입력값으로 받아서 이미지 파일을 잘라냅니다.

```
   get(int x, int y, int width, int height);
```

혹시 너비와 높이를 정하지 않으면 (x, y) 지점의 픽셀값 하나만 떼옵니다.

08.03
이미지 모자이크 만들기

유명한 회화의 일부를 잘라 다시 배열해 봅시다. 생각보다 재미있는 패턴이 들어나는 것을 발견할 수 있습니다. 이미지 다루는 법을 연습하는 좋은 기회입니다.

이미지 자르기

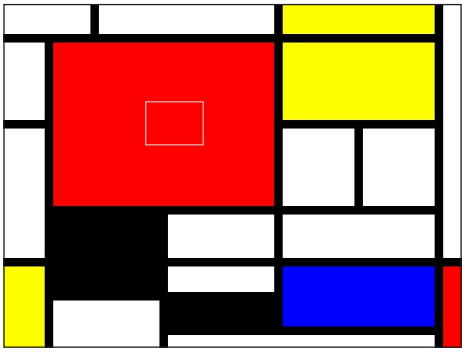
이미지를 화면에 띄우고 마우스를 따라 작은 사각형을 이미지 위에 띄우도록 합시다. 사각형 내부의 이미지가 원본에서 떼어낼 부분입니다.

이미지 띄우기와 사각형 그리기

하나씩 원하는 기능을 추가해 가는 것이 오히려 한꺼번에 모든 기능을 추가하려는 것보다 더 빨리 완성할 때가 많습니다. 일단 이미지를 띄우고 사각형을 그리는 것에 집중합시다.

```
//myCropRect

PImage img;
PImage cropped;
String URL = "http://c85c7a.medialib.glogster.com/
media/b3/b32d597d70ee28fe942e2a8e7b485684456a4b0
ffc7c955a4d6cb874ee60d894/mondrian1.jpg";

int tileWidth = 128;
```

```
int tileHeight = 96;

void setup() {
   size(1024, 768);
   img = loadImage(URL);
}

void draw() {
   image(img, 0, 0);
   int posX = constrain(mouseX, 0, width -
tileWidth);
   int posY = constrain(mouseY, 0, height -
tileHeight);

   stroke(255);
   strokeWeight(2);
   noFill();

   rect(posX, posY, tileWidth, tileHeight);
}
```

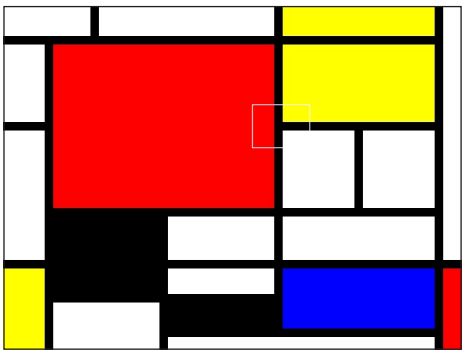


myCropRect

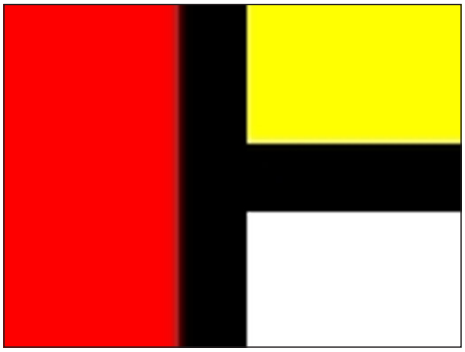
이미지 잘라내기

이미지를 잘라내는 것은 이전에 설명한 것처럼 get() 메서드를 이용합니다. 이전에 배웠던 내용을 다시 확인해 볼까요? 마우스를 클릭하는 동안 사각형 내부의 이미지가 전체 화면으로 확대되게 만들어 보겠습니다.

```
//myCropImage

PImage img;
PImage cropped;
String URL = "http://c85c7a.medialib.glogster.com/
media/b3/b32d597d70ee28fe942e2a8e7b485684456a4b0
ffc7c955a4d6cb874ee60d894/mondrian1.jpg";

int tileWidth = 128;
int tileHeight = 96;

void setup() {
   size(1024, 768);
   img = loadImage(URL);
}

void draw() {
   image(img, 0, 0);
   int posX = constrain(mouseX, 0, width -
tileWidth);
   int posY = constrain(mouseY, 0, height -
tileHeight);
```



myCropImage



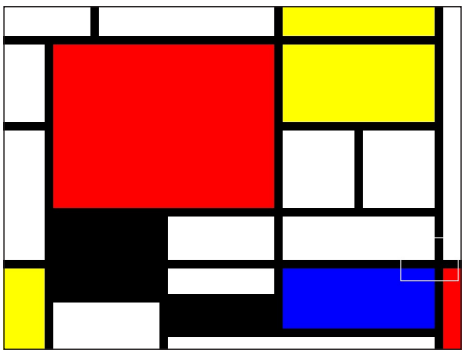
myCropImage

```
  stroke(255);
  strokeWeight(2);
  noFill();

  rect(posX, posY, tileWidth, tileHeight);

  if(mousePressed) {
    cropped = img.get(posX, posY, tileWidth,
tileHeight);
    image(cropped, 0, 0, width, height);
  }
}
```

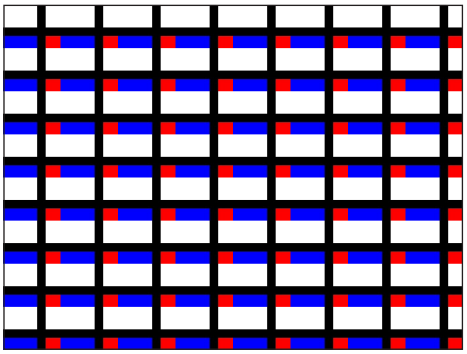
이미지 배치하기

이제 잘라낸 이미지를 전체 화면에 표시하지 않고 화면에 반복해서 표시하겠습니다. 지금까지 했던 작업과 크게 다르지 않습니다. 몇 개의 변수를 추가해서 화면에 배치하는 작업을 쉽게 하겠습니다.

```
//myCropImageDisplay
PImage img;
PImage cropped;
boolean mosaicMode = false;
String URL = "http://c85c7a.medialib.glogster.com/
media/b3/b32d597d70ee28fe942e2a8e7b485684456a4b0
ffc7c955a4d6cb874ee60d894/mondrian1.jpg";

int tileWidth = 128;
int tileHeight = 96;
int nX;
int nY;
int posX;
int posY;

void setup() {
  size(1024, 768);
  img = loadImage(URL);
  nX = width / tileWidth;
  nY = height / tileHeight;
}

void draw() {
  image(img, 0, 0);
  posX = constrain(mouseX, 0, width - tileWidth);
  posY = constrain(mouseY, 0, height - tileHeight);

  stroke(255);
  strokeWeight(2);
  noFill();
```



myCropImageDisplay



myCropImageDisplay

```
   rect(posX, posY, tileWidth, tileHeight);

   if(mosaicMode) {
      for(int y  = 0; y < nY; y++) {
         for(int x = 0; x < nX; x++) {
            image(cropped, x * tileWidth, y *
tileHeight);
         }
      }
   }
}

void mousePressed() {
   cropped = img.get(posX, posY, tileWidth,
tileHeight);
   mosaicMode = true;
}

void mouseMoved() {
   mosaicMode = false;
}
```

이미지 임의로 자르기

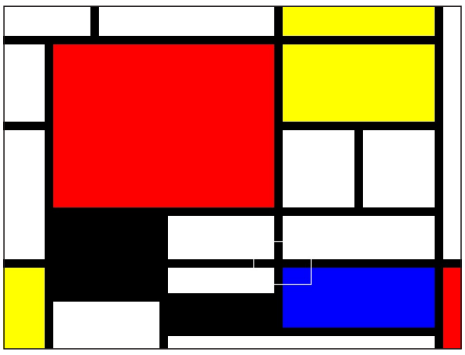
하지만 너무 정확하게 같은 조각을 자르면 재미가 없습니다. 자를 때 약간씩 오차를 허용하면 어떨까요? random() 함수를 이용해 오차를 발생합시다.

덧붙여 각각의 조각들을 담을 배열을 아래와 같이 정의해서 조각을 저장합시다.

```
PImage[][] crops = new PImage[nY][nX];
```

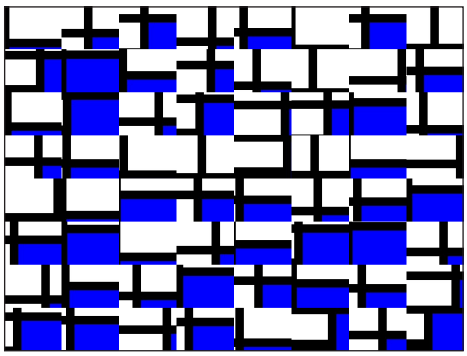
실제 코드를 써보면 아래와 같습니다.

```
//myCropImageDisplayRandom
PImage img;
PImage[][] crops;

boolean mosaicMode = false;

String URL = "http://c85c7a.medialib.glogster.com/
media/b3/b32d597d70ee28fe942e2a8e7b485684456a4b0
ffc7c955a4d6cb874ee60d894/mondrian1.jpg";

int tileWidth = 128;
int tileHeight = 96;
int nX;
```



myCropImageDisplayRandom

```
int nY;
int posX;
int posY;

void setup() {
   size(1024, 768);
   img = loadImage(URL);
   nX = width / tileWidth;
   nY = height / tileHeight;
   crops = new PImage[nY][nX];
}

void draw() {
   image(img, 0, 0);
   posX = constrain(mouseX, 0, width - tileWidth);
   posY = constrain(mouseY, 0, height -
tileHeight);

   stroke(255);
   strokeWeight(2);
   noFill();

   rect(posX, posY, tileWidth, tileHeight);

   if(mosaicMode) {
      for(int y  = 0; y < nY; y++) {
         for(int x = 0; x < nX; x++) {
            image(crops[y][x], x * tileWidth, y *
tileHeight);
         }
      }
   }
}

void mousePressed() {
   for (int y = 0; y < nY; y++) {
      for (int x = 0; x < nX; x++) {
         int posRX = int(posX + 0.5 * tileWidth *
random(-1, 1));
         int posRY = int(posY + 0.5 * tileHeight *
random(-1, 1));
         crops[y][x] = img.get(posRX, posRY,
tileWidth, tileHeight);
      }
   }
   mosaicMode = true;
}

void mouseMoved() {
   mosaicMode = false;
}
```



myCropImageDisplayRandom

다시 한 번 외쳐봅시다. 그럴 듯한 작품을 만드는 데 엄청난 지식이 필요한 것은 아닙니다. 약간만 배워도 조금만 알아도 이런 재미있는 일들을 할 수 있습니다. 여러분 두려워 마시고 함께 배워봅시다.

07.04
하나 이상의 좌표체계: 톱니바퀴 따로 돌리기

이제 도형을 회전하는 일이 어렵지 않습니다. translate()를 이용해 회전시킬 도형의 중심으로 원점을 이동시킨 후 rotate() 함수를 이용하면 도형의 회전을 쉽게 할 수 있습니다.

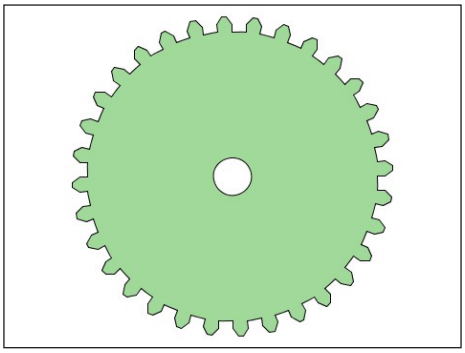
회전을 익히기 위해 하나 더 연습해보겠습니다. 이제 톱니바퀴를 돌려봅시다.

[참고 링크: https://bl.ocks.org/mbostock/1353700]

톱니바퀴 하나 그리기

톱니바퀴 SVG를 만들어서 화면에 띄운 후 회전시킵니다.

```
//myGearPlanet
PShape planet;
float theta = 0.0;

void setup() {
  size(480, 360);
  planet = loadShape("planet.svg");
  planet.disableStyle();
}

void draw() {
```



myGearPlanet

```
  background(255);
  translate(width / 2.0, height / 2.0);
  rotate(theta);
  fill(#a1d99b);
  shape(planet, 0, 0);

  theta += 0.01;
}
```

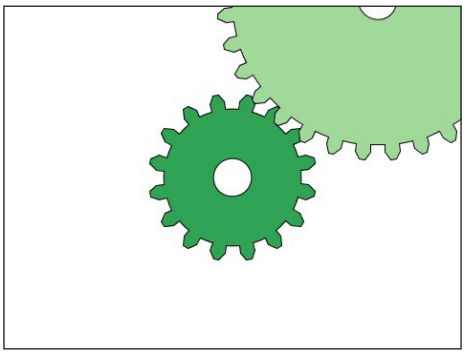
translate()와 rotate()를 결합하니 간단하게 도형을 회전할 수 있습니다.

톱니바퀴 두 개를 회전시키기

하나로는 성에 차지 않습니다. 톱니바퀴는 맞물려서 돌아야야 제대로 된 그림 아니겠습니까?

톱니 수가 적은 톱니바퀴를 하나 더 추가합니다. 작은 톱니바퀴는 16개의 톱니를 가지고 큰 톱니바퀴는 32개의 톱니바퀴를 가지고 있습니다. 따라서 큰 톱니바퀴는 더 천천히, 정확하게는 1/2의 속도로 회전해야 합니다. 이 부분을 추가로 수정했습니다.

```
//myGearPlanetAndSun
PShape sun;
PShape planet;
PShape annulus;
PShape solarSystem;
float theta = 0.0;

void setup() {
  size(480, 360);
  solarSystem = loadShape("solarSystem.svg");
  solarSystem.disableStyle();
  planet = solarSystem.getChild("planet");
  sun = solarSystem.getChild("sun");
}

void draw() {
  background(255);
  translate(width / 2.0, height / 2.0);
  rotate(theta);
  fill(#31a354);
  shape(sun, 0, 0);
  translate(240, 0);
  fill(#a1d99b);
  rotate(-theta * 16 / 32);
  shape(planet, 0, 0);

  theta += 0.01;
```



myGearPlanetAndSun

```
}
```

무언가 이상합니다. 톱니바퀴는 각자 회전해야 하는데 화면 가운데를 중심으로 두 톱니바퀴가 함께 돌고 있습니다. 이 문제를 어떻게 해결해야 할까요?

현재 사용하는 좌표체계를 저장할 수 있으면 좋겠습니다. 좌표체계가 바뀔 때마다 좌표체계를 저장하고 translate(), rotate()를 적용한 다음 더이상 쓸모없어지면 이전 체계로 돌아가면 문제가 해결될 것 같습니다.

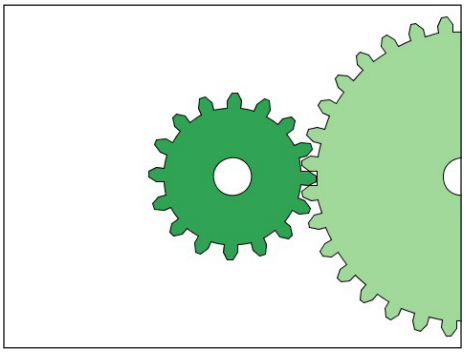
현재 좌표체계를 저장하는 함수가 pushMatrix()입니다. 반대로 현재 좌표체계를 버리고 이전 체계로 돌아가는 함수가 popMatrix()입니다.

좌표체계를 기억하는 것은 빨래통에 빨래를 집어넣는 것과 비슷합니다. 빨래통에 세탁물을 넣었다면(push) 세탁기에 빨래를 넣을 때(pop) 빨래통의 가장 위에 있는 의류부터 빼내야 합니다.

이 시스템은 편리한 점이 있습니다. 지금 어떤 좌표체계를 사용하고 있는지 추적할 필요가 없습니다. 가장 위에 있는 좌표체계만 확인하면 됩니다. 단지 지금 사용하는 좌표체계를 집어넣고(pushMatrix()) 사용이 끝나면 popMatrix()를 실행하면 됩니다.

실제 예제를 봅시다.

```
//myGearPlanetAndSunDualSystem
PShape sun;
PShape planet;
PShape annulus;
PShape solarSystem;
float theta = 0.0;

void setup() {
   size(480, 360);
   solarSystem = loadShape("solarSystem.svg");
   solarSystem.disableStyle();
   planet = solarSystem.getChild("planet");
   sun = solarSystem.getChild("sun");
}

void draw() {
   background(255);
   translate(width / 2.0, height / 2.0);
   pushMatrix();
      rotate(theta);
      fill(#31a354);
      shape(sun, 0, 0);
   popMatrix();
```



myGearPlanetAndSunDualSystem

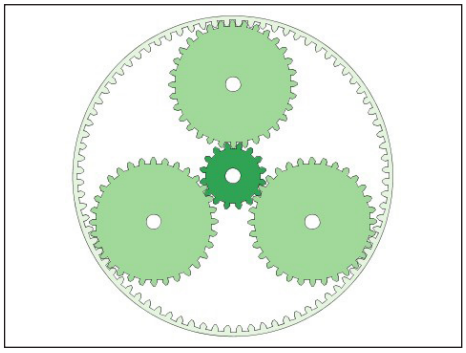
```
   translate(240, 0);
   pushMatrix();
      fill(#a1d99b);
      rotate(-theta * 16 / 32);
      shape(planet, 0, 0);
   popMatrix();

   theta += 0.01;
}
```

톱니바퀴 여러 개를 회전시키기

그냥 지나칠 수 없습니다. 조금 더 추가해 더 그럴듯한 그림을 그려봅시다. 여러분들은 이미 많은 것을 알고 있습니다. 약간의 상상력만 가미해 봅시다.

이제 톱니바퀴 여러 개를 동시에 회전시킵시다.

```
//myGearSolarSystem
PShape sun;
PShape planet;
PShape annulus;
PShape solarSystem;
float theta = 0.0;

void setup() {
   size(480, 360);
   solarSystem = loadShape("solarSystem.svg");
   solarSystem.disableStyle();
   planet = solarSystem.getChild("planet");
   sun = solarSystem.getChild("sun");
   annulus = solarSystem.getChild("annulus");
}

void draw() {
   background(255);
   translate(width / 2.0, height / 2.0);
   scale(0.4);

   //sun
   pushMatrix();
      rotate(theta);
      fill(#31a354);
      shape(sun, 0, 0);
   popMatrix();

   //planet01
   pushMatrix();
      translate(240 * cos(PI / 6), 240 * sin(PI / 6));
```



myGearSolarSystem

```
    fill(#a1d99b);
    rotate(-theta * 16 / 32);
    shape(planet, 0, 0);
  popMatrix();

  //planet02
  pushMatrix();
    translate(240 * cos(PI * 5 / 6), 240 * sin(PI * 5
/ 6));
    fill(#a1d99b);
    rotate(-theta * 16 / 32);
    shape(planet, 0, 0);
  popMatrix();

  //planet03
  pushMatrix();
    translate(240 * cos(PI * 9 / 6), 240 * sin(PI * 9
/ 6));
    fill(#a1d99b);
    rotate(-theta * 16 / 32);
    shape(planet, 0, 0);
  popMatrix();

  //annulus
  pushMatrix();
    fill(#e5f5e0);
    rotate(-theta * 16 / 80);
    shape(annulus, 0, 0);
  popMatrix();

  theta += 0.01;
}
```

popMatrix()와 pushMatrix()는 편리한 도구입니다. 손에 익도록 자주 사용합시다.