01
프로그래밍과 프로세싱

01.01
프로그래밍 이해하기와 프로세싱

프로그래밍이란?

생각해보세요. 여러분은 마트의 관리자입니다. 매출을 올리기 위해 묶음상품을 만들어 보려고 합니다. 매장 사원들에게 어떤 상품이 함께 팔리는지 의견을 들었지만, 여러분 스스로를 설득하기에 그들의 단순한 의견은 충분하지 않습니다. 그래서 매출데이트를 분석해보려고 합니다. 이런 저런 크고 작은 문제를 해결하고 데이터를 성공적으로 확보했습니다. 그런데 막상 데이터 파일을 열어보니 아무 생각이 나지 않습니다. 엑셀에서 열리지도 않는 이 커다란 텍스트 파일을 막상 어떻게 분석할지 생각해보지 않았습니다. 한참을 고민하다 "그래 이건 내 길이 아닌 것 같아. 나에게는 직감이 있잖아!"라고 말하며 애써 받아온 파일을 닫아버립니다. 프로그래밍을 할 수 있었다면 이 문제를 좀 더 자신있게 풀 수 있지 않았을까요?

우리는 일상에서 여러 문제를 접하며, 문제를 풀기 위해 정보를 수집하고 분석합니다. 간단한 분석과 직관으로 충분한 경우도 있지만, 늘 이렇게 운이 좋은 것은 아닙니다. 복잡하고 어려운 문제는 더 많은 자료와 다양한 분석을 요구합니다. 더 많은 자료를 분석하는 데 기계의 도움은 필수적입니다. 손으로 혹은 엑셀로 자료를 분석하는데는 한계가 있습니다 . 더 많은 자료를 다양하게 분석하려는 여러분에게 프로그래밍 스킬은 필수요소가 될 것입니다. 물고기가 물 속에서 숨쉬듯 우리는 일상에서 프로그래밍을 할 것입니다.

하지만 이런 실용적인 이유가 프로그래밍을 배워야 할 이유의 전부는 아닙니다. 밥벌이가 프로그래밍을 배워야 할 유일한 이유일 리 없습니다. 오히려 여러분의 인생에 더 의미있는 이유가 있습니다.

무엇보다도 프로그래밍은 재미있습니다. 지금도 수많은 사람들이 밤을 지새며 컴퓨터 스크린 앞에서 머리를 쥐어뜯으며 프로그래밍에 시간을 보내는 이유는 프로그래밍이 그토록 재미있기 때문입니다. 마치 어려운 퍼

즐을 풀고 느끼는 희열처럼, 마음을 옭죄던 프로그램밍 문제를 해결했을 때의 기쁨이란 그동안의 피로를 잊게 해주고, 순수한 성취감을 맛보게 할 정도로 대단한 것입니다. 그런 기쁨을 알기에 오늘도 프로그래머들은 모니터 앞에서 밤을 지새고 있을 겁니다.

왜 프로세싱인가?

하지만 프로그래밍을 익히는 일은 만만하지 않습니다. C++나 Java같은 기존 언어를 배우는 것은 많은 것을 요구합니다. 가장 간단한 프로그램, 예를 들어 "Hello, World!"를 화면에 출력하려고 해도 알아야 할 개념과 프로그램 언어 상세가 몇십페이지가 될 것 같습니다. 하물며 사각형 그리기, 마우스의 움직임에 따라 다른 색을 칠하기, MS 키넥트 연결해 동작인식하기는 얼마나 복잡하겠습니까? 하지만 프로세싱에서는 쉽게 구현할 수 있습니다. 여전히 약간의 부가지식이 필요하지만 다른 언어에서 구현하는데 필요한 지식에 비한다면 거의 공짜로 사용하는 수준입니다.

물론 프로세싱이 모든 종류의 문제를 푸는 데 적합하다는 이야기는 아닙니다. 구글 시스템을 만드는 데 프로세싱을 사용하고 싶지는 않을 겁니다. 그렇지만 프로세싱은 여러분, 프로그래밍을 처음 배우는 여러분들이 프로그래밍의 기본을 배우는 데 좋은 환경을 제공합니다. 문법은 간단하고 라이브러리는 풍부합니다. 공식 사이트의 레퍼런스는 친절하고 쉽게 이해할 수 있습니다. 많은 사용자들이 존재하고 적당한 예제코드를 인터넷에서 검색하는 것도 어렵지 않습니다.

자 이제 프로세싱을 배워볼 준비가 되었나요?

프로세싱은 어떤 언어인가?

프로세싱은 예술가와 디자이너들이 컴퓨터와 프로그래밍을 친숙하게 접할 수 있도록 고안된 프로그래밍 언어입니다. C#, Java 그리고 python과 같은 범용언어와는 다르게 인터랙티브 프로그래밍에 방점을 두고 있는 언어입니다. 프로세싱의 디자이너이자 지금도 프로세싱 개발의 중추를 맡고 있는 케이시 리아스와 벤 프라이의 말을 들어보겠습니다.

> 우리는 인터랙티브 그래픽 프로그래밍을 쉽게 할 수 있도록 프로세싱을 만들었다. 당시 우리가 일반적으로 사용하는 프로그래밍 언어(C++와 Java)는 이러한 유형의 소프트 웨어를 작성하기가 매우 어려웠기 때문에 불만족스러웠는데, 그에 비해 어린 시절 썼던 언어들(Logo와 BASIC)로는 흥미로운 프로그램을 매우 간단하게 작성할 수 있다는 사실에서 영감을 받았다.
>
> 우리의 목표는 당시 우리가 작업하고 있던 소프트웨어들을 스케치(프로토타입)할 수 있는 방법을 찾는 것이었다.
>
> 우리의 또다른 목표는 디자인이나 예술을 전공하는 학생들에게는 프로그램하는 방법을 가르칠 수 있고, ... 데이터 구조와 텍스트 콘솔 출력보다는 '그래픽'과 '인터랙션'에 초점을 맞추기 시작했다.

프로세싱의 이러한 특성은 프로그래밍을 처음 배우는 여러분과 같은 사람에게 여러 이점을 줍니다. 모든 이점을 이 자리에서 설명할 수는 없습니다. 프로그래밍에 대한 관심이 계속되어 다른 프로그래밍 언어를 접할 때 여러분은 프로세싱을 그리워할 것입니다. 그 순간은 여러분의 생각보다 빨리 다가올 겁니다.

프로세싱으로 무엇을 할 수 있나?

프로세싱의 시작은 인터랙티브 프로그래밍으로 미약하였으나 그 끝은 창대하여 프로세싱의 줄기가 어디까지 뻗어갈 지 알기 힘듭니다.

데스크탑 환경에서 시작한 이후 여러 개발자의 노력으로 다양한 환경에서 프로세싱을 구현할 수 있게되었습니다. processing.js가 도입된 이후, 프로세싱 스케

치를 수정업이 웹페이지에 삽입하는 것이 가능해졌습니다. 이제 한단계 더 나가, 단순히 웹페이지의 한 부분이 아니라 웹 전체에서 프로세싱을 사용할 수 있는 p5.js 프로젝트도 좋은 성과를 보이고 있습니다.피지컬 프로그래밍에서도 프로세싱이 자주 보이고 있습니다. 관심이 없으신 분이라도 한번쯤 들어보셨을 법할 정도로 유명해진 아두이노 프로젝트의 IDE가 프로세싱에 기반을 두고 있습니다. 프로세싱에 익숙하신 분이라면 별 어려움없이 아두이노 프로젝트를 진행하실 수 있습니다.

여러분! 프로세싱을 선택하신 것은 좋은 결정입니다. 프로세싱 배웁시다. 열심히!

01.02
프로세싱, 어떻게 배울 것인가?

실수를 즐기세요

처음부터 끝까지 이 책을 읽는 것으로 충분하지 않습니다. 프로그래밍 실력은 눈이 아니라 손 끝에서 자랍니다. 우선 여기에 소개된 프로젝트를 따라하세요. 프로세싱 IDE에 코드를 손으로 입력하고 실행하는 것만으로도 얻을 수 있는 것이 많습니다. 그리고 스케치를 실행하여 스케치가 의도대로 움직이는지 살펴보세요. 처음 시도에 제대로 돌아간다면 정말 축하드립니다. 여러분의 프로그래머 인생에 흔치 않은 순간일 것입니다. 제대로 동작하지 않아도 축하드립니다. 여러분은 실수에서 더 많은 것을 배울 수 있습니다.

결국 여러분의 실수와 고민이야말로 여러분이 새로운 것을 배울 수 있는 기회입니다. 문제를 해결할 수 있다는 믿음을 가지고 스스로를 가시밭길로 몰아가세요. 여러분은 할 수 있습니다.

도전하세요

코드의 각 부분이 무슨 역할을 하는지 이해할 수 있을 즈음이 되면 빈 종이에 프로세싱 스케치의 개요를 적어보세요. 그리고 여러분이 작성한 개요를 바탕으로 책을 덮고 스케치를 다시 작성해 보세요. 막히는 부분이 있으면 다시 책을 참고할 수 있지만 혼자 완성할 수 있도록 노력하세요. 고민할 수 있는 시간을 충분히 가지세요. 새로운 배움은 오직 고민하는 순간에만 얻어지는 것입니다. 여러분의 스케치 코드가 제대로 돌아가면 이제 교재의 코드와 자신의 코드를 비교할 시간입니다. 교재의 코드가 언제나 옳은 것은 아닙니다. 두 코드를 비교하고 더 나은 코드를 구상해보세요.

반복하세요

교재 상의 코드 이해하기부터 내가 짠 코드 회고하기까지가 프로그래밍 연습의 한 주기입니다. 이제 이 주기를 반복하세요. 회고가 끝난 코드를 바탕으로 개요를 작성

하고 다시 여러분의 코드를 작성하는 겁니다. 그리고 다시 회고. 이 주기를 반복하고 다시 반복하세요.

많은 프로젝트를 연습할 필요는 없습니다. 여러 문제를 푸는 것보다 엄선된 소수의 문제를 반복해서 푸는 것이 프로그래밍 실력을 향상시키는 데 도움이 더 되기 마련입니다.

함께 하세요

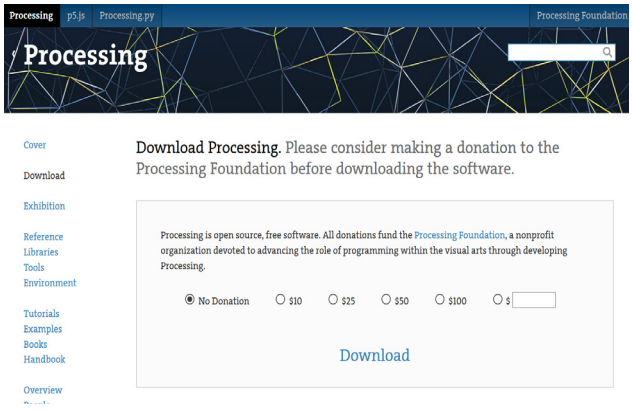
친구들과 같이 공부하세요. 게임을 같이 하듯 프로세싱 스케치를 함께 써보세요. 각자 컴퓨터를 준비하는 것도 좋지만, 짝을 지어 한 컴퓨터에서 연습하는 것(https://en.wikipedia.org/wiki/Pair_programming)도 좋은 방법입니다. 함께 가면 멀리 갈 수 있습니다.

01.03
프로세싱 설치와 내가 만든 첫번째 프로그램

프로세싱을 설치하는 데는 인터넷에 연결된 컴퓨터 한 대와 새로운 시도하는 용기가 조금 필요할 뿐입니다.

웹사이트 방문하기

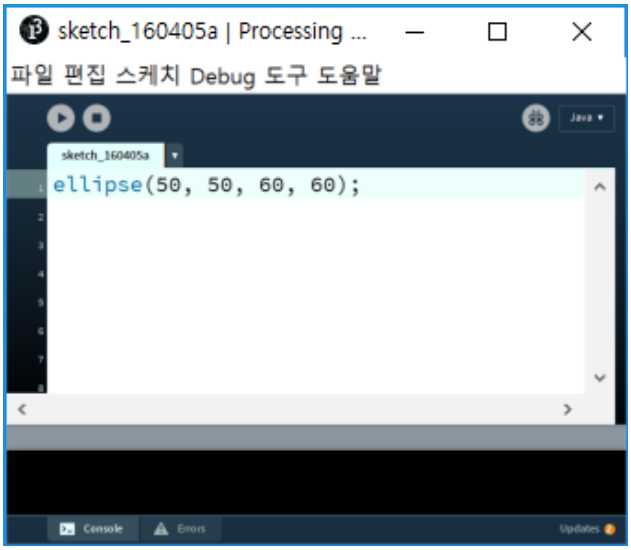
www.processing.org/download/에 접속하세요. 프로세싱을 다운받기 전 기부여부를 묻는 화면으로 이동할 것입니다. 기부는 여러분이 선택할 수 있습니다. Donate & Download 링크 혹은 Download(No Donation을 선택한 경우)를 클릭하세요. 운영체제와 버전을 선택할 수 있는 페이지로 이동하여 자신의 컴퓨터에 맡는 버전을 고르세요. 윈도우의 경우 32비트와 64비트 두가지 버전이 있습니다. 자신의 컴퓨터에 맞는 버전을 선택하세요.



파일을 다운로드 받고 적당한 위치에 압축을 푸세요. 저는 보통 바탕화면에 복사해둡니다.

윈도우의 경우 다른 윈도우 응용프로그램과 다르

게 프로세싱은 별도의 설치과정이 필요없습니다. processing.exe파일을 더블클릭하는 것으로 첫번째 스케치를 실행할 준비가 끝납니다.

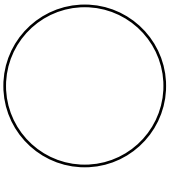


첫번째 실행

처음 실행할 경우 환영 메세지 창이 뜰 것입니다. 살펴보시고 메세지 창을 닫읍시다. 그러면 왼쪽 그림의 편집창을 볼 수 있습니다. 아래의 코드를 입력하고 실행시키면 스케치 창이 등장합니다.

마지막 ; 을 잊지마세요. 프로세싱에서 모든 명령은 ;로 마칩니다.

```
ellipse(50, 50, 60, 60);
```



별 것 아니지 않습니까? 벌써 프로그래머가 된 것같은 기분입니다.

ellipse()와 같은 것을 함수라고 부릅니다. 괄호 안에 타원의 위치와 모양을 특정하는 값을 입력받습니다. ellipse()의 경우 입력값은 차례대로

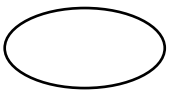
```
ellipse(x_position, y_position, width, height)
```

입니다. 더 자세한 내용은 기본도형에서 살펴보겠습니다.

두번째 프로그램

ellipse() 안에 들어가는 값을 바꿔 다른 모양의 원을 그려봅시다. 예를 들어 width값을 height값보다 크게 하면 위아래로 찌그러진 타원을 그릴 수 있습니다.

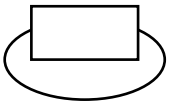
```
ellipse(50, 50, 60, 30);
```



ellipse만 그릴 필요도 없습니다. 이번에는 사각형을 추가해봅시다.

```
ellipse(50, 50, 60, 30);
```

```
rect(30, 30, 40, 20);
```



사각형이 타원 위에 겹쳐집니다. 이 예에서 알 수 있듯 프로세싱 코드는 위에서 아래로 차례로 실행되며 나중에 그려지는 도형은 앞서 그려진 도형 위에 겹쳐져 그려집니다.

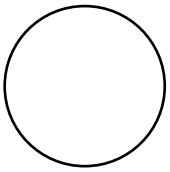
마우스를 따라다니는 도형

정지되어 있는 도형은 이제 시시합니다. 새로운 것에 도전해보고 싶습니다. 마우스를 따라다니는 도형은 어떨까요? 마우스의 현재 위치에 도형을 매번 그리는 것이 가능할까요?

프로세싱은 이런 작업을 위해 mouseX와 mouseY라는 내장변수를 가지고 있습니다. 각각 마우스의 가로와 세로 위치를 픽셀단위로 알려줍니다. 마우스의 위치에 숫자 대신 (mouseX, mouseY)를 사용합니다.

한가지 추가할 사항이 있습니다. 지금까지는 바로 함수를 사용했지만 이번에는 도형을 그리는 함수를 다른 함수 안에 집어넣습니다. 왜 이렇게 하는지는 다음 시간에 알게 됩니다.

```
void draw() {
    ellipse(mouseX, mouseY, 60, 60);
}
```

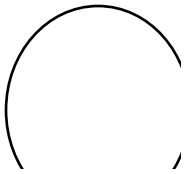


마우스를 따라다니는 원을 관찰할 수 있습니다.

마우스를 따라 크기가 바뀌는 도형

마우스를 따라 도형을 그려봤으니 이제 다음 단계로 위치에 따라 크기가 바뀌는 도형을 그려봅시다. 처음에는 막막하겠지만 한 번 해봅시다. ellipse()의 입력값 중 세번째와 네번째가 가로/세로 크기를 나타냅니다. 여기에 차례로 mouseX와 mouseY를 집어넣읍시다.

```
void draw() {
    ellipse(mouseX, mouseY, mouseX,
```



```
  mouseY);
}
```

왼쪽 위로 움직일수록 원이 작아지고, 오른쪽 아래로 갈수록 원이 커집니다.

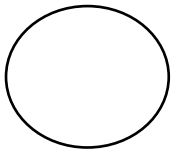
왼쪽 위로 갈수록 커지는 도형

위의 코드는 왼쪽 위로 갈수록 크기가 작아집니다. 반대로 왼쪽 위로 갈수록 크기가 커지고 오른쪽 아래로 갈수록 작아지는 도형을 그릴 수 있을까요?

이런 경우를 다룰 때는 극단적인 예를 생각해보면 실마리가 풀리는 경우가 많습니다. 왼쪽 위에 있을 때 너비 100px, 높이 100px이고 오른쪽 아래에 위치할 때 너비 0px, 높이 0px이 되려면 어떻게 해야 할까요?

기본 화면크기는 100px by 100px입니다. 그렇다면 아래와 같이 실행해보는 건 어떤가요?

```
void draw() {
    ellipse(mouseX, mouseY, 100 -
mouseX, 100 - mouseY);
}
```



프로그래밍이 점점 재미있어집니다.