Labs for Graphics

Mobile Software

2019 Fall

Lab. 준비

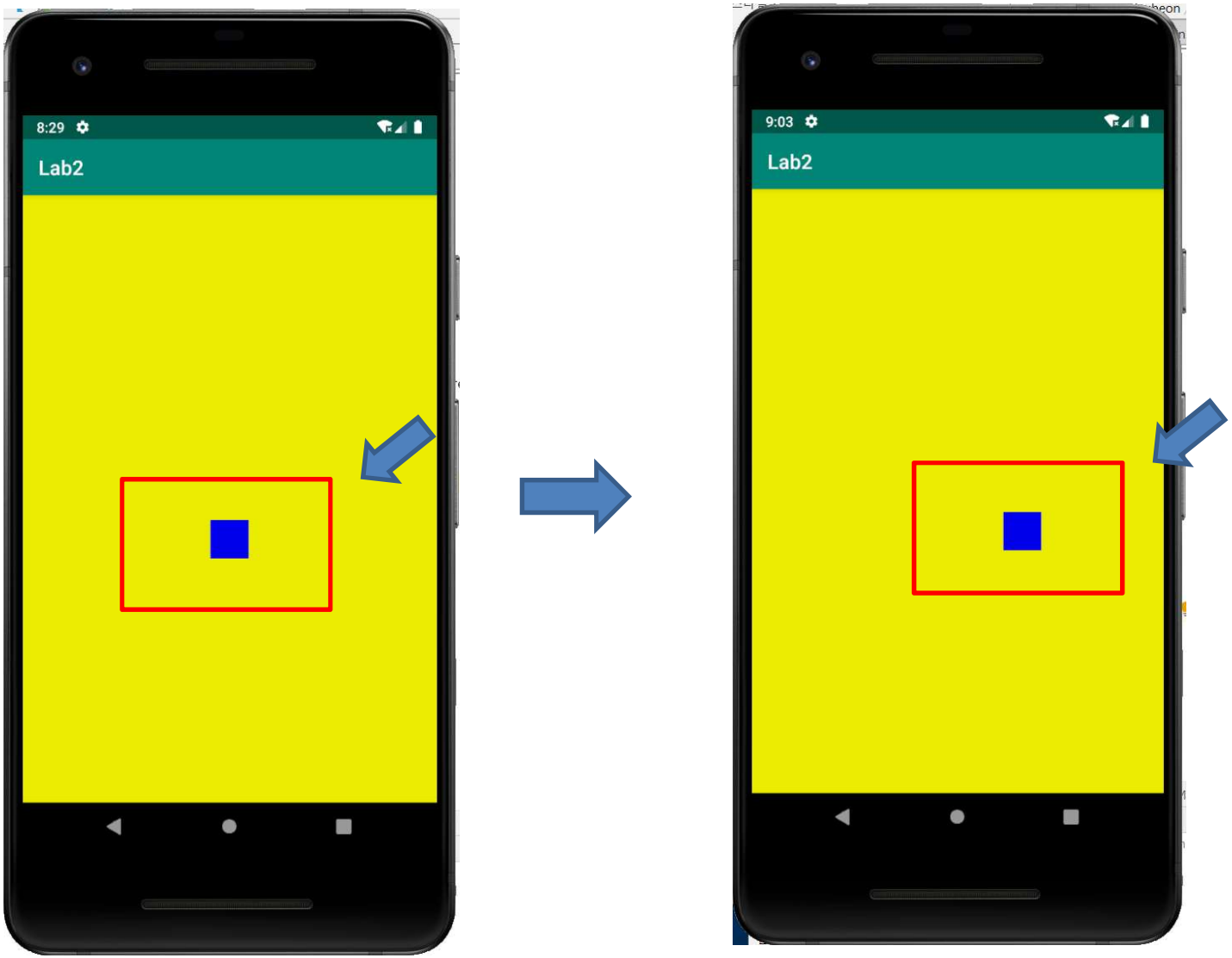
- 새 프로젝트 생성
  - **Application name**
    - **Lab2-학번**
  - **Target Android Devices**
    - **Phone and Tablet**
      - minimum SDK **– API 26** 이상
  - **Activity**
    - **Empty Activity**

- 자동 생성된 layout은 **ConstraintLayout**

What to do next?

- **Lab 1. key를 눌러 사각형 이동**
- Lab 2. 자유 곡선 그리기
- Lab 3. 도형 크기 조절
- Lab 4. 색상 선택이 가능한 곡선 그리기
- Lab 5. 볼륨 컨트롤 View 만들기

Lab 1: key를 눌러 사각형 이동



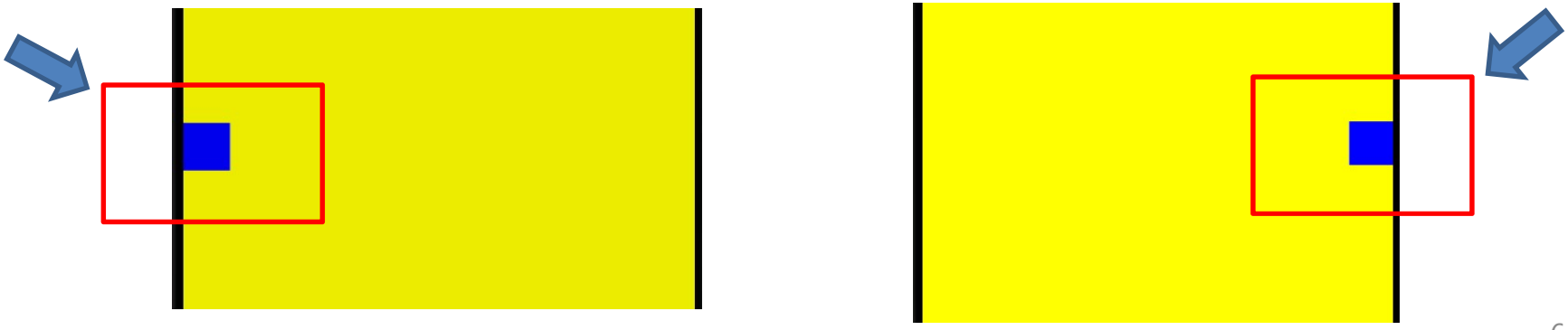
Lab 1. key를 눌러 사각형 이동

- XML Layout은 사용하지 않음
- 키를 눌러 사각형을 좌우로 이동시킴
  - J를 누르면 왼쪽으로 이동
  - K를 누르면 오른쪽으로 이동
- 구현 과정: 동영상(**Move Rectangle App**.)

- **이를 참고해서 Practice 1을 구현하세요.**

Practice 1: 화면 바깥 이동 방지

- 상하 이동 기능 추가
  - M를 누르면 아래로 이동
  - N을 누르면 위로 이동
- 화면 바깥 이동 방지
  - 상하 또는 좌우로 이동할 때 화면 바깥으로 나가지 않아야 함



Practice 1: Hint

- 화면의 size를 알아야 함
  - widthView, heightView
- 한 가지 방법

```
val dm = resources.displayMetrics
widthView = dm.heightPixels
heightView = dm.widthPixels
```

- 그러나 위 방법은 문제가 있음.
  - **좌우로 이동할 때는 문제가 없지만 상하로 이동할 때 문제 발생**
    - 실제 그래픽 공간의 높이는 ActionBar 높이 등을 빼야 하지만, 위에서 구한 높이는 이를 고려하지 않은 화면의 높이임
  - 이 문제를 어떻게 해결할 수 있을까?
    - **Hint 동영상을 끝까지 보고 해결책에 관한 설명을 들으세요!!**

What to do next?

- Lab 1. key를 눌러 사각형 이동
- **Lab 2. 자유 곡선 그리기**
- Lab 3. 도형 크기 조절
- Lab 4. 색상 선택이 가능한 곡선 그리기
- Lab 5. 볼륨 컨트롤 View 만들기

Lab 2: 자유 곡선 그리기

MyView – 다양한 코딩 스타일

```
class MyView(context:Context) : View(context) {

    private val mPaint = Paint()
    private val mPath = Path()

    init {
        initialize()
    }
```

Constructor 한 개만 사용할 때는 이 코딩 스타일이 편하지만,

constructor를 여러 개 정의 할 때는 아래 코딩 스타일을 사용

```
class MyView : View {

    private val mPaint = Paint()
    private val mPath = Path()

    constructor(context: Context) : super(context) {
        initialize()
    }
```

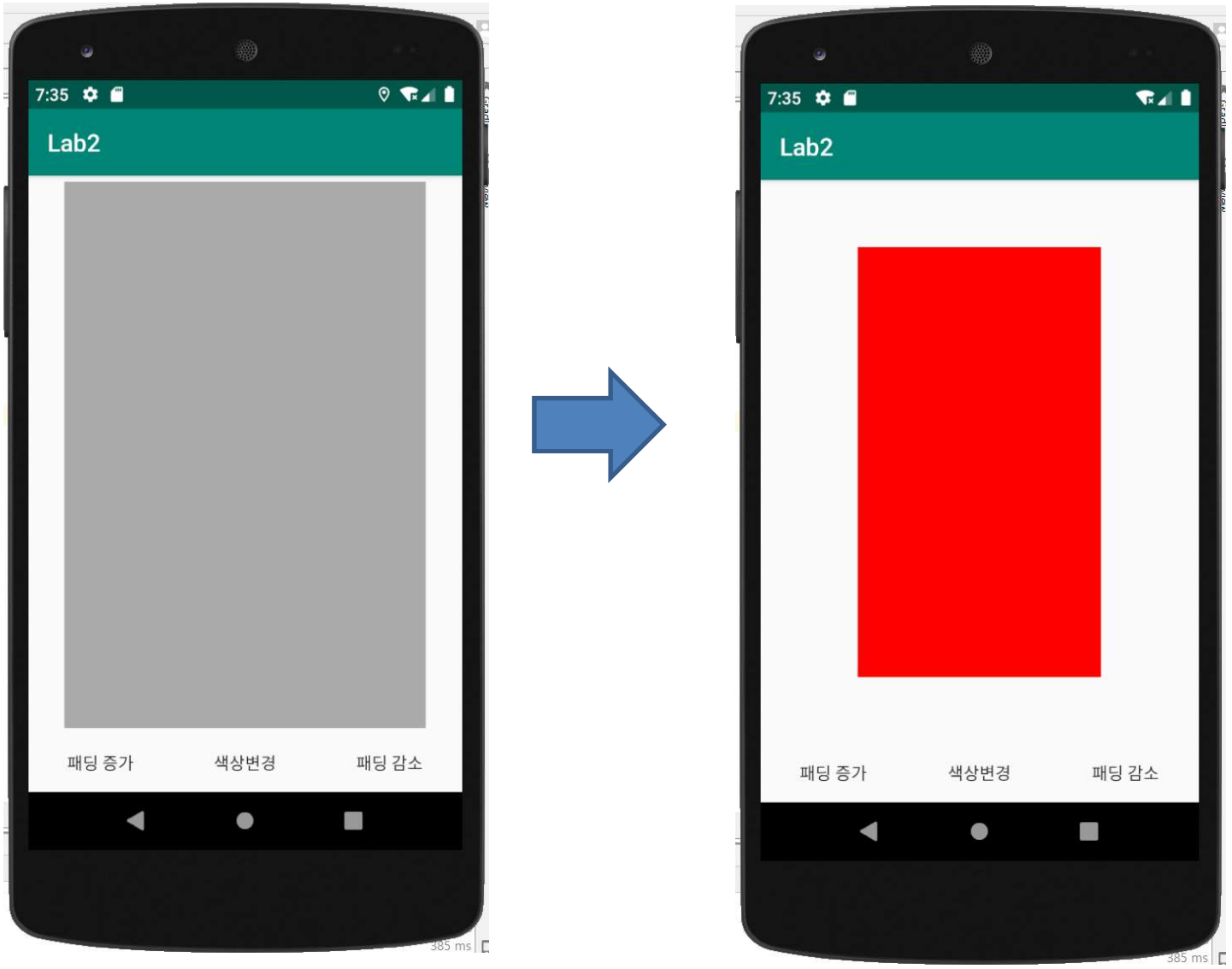
Lab 2: 자유 곡선 그리기

- XML Layout은 사용하지 않음
- 화면 터치를 통해 자유롭게 선을 그림
- 구현 과정: 동영상(**Free Curve App**.)

What to do next?

- Lab 1. key를 눌러 사각형 이동
- Lab 2. 자유 곡선 그리기
- **Lab 3. 도형 크기 조절**
- Lab 4. 색상 선택이 가능한 곡선 그리기
- Lab 5. 볼륨 컨트롤 View 만들기

Lab 3: 도형 크기 조절



Lab 3: 도형 크기 조절

- XML Layout 사용
  - XML Layout에 custom view를 포함
  - 3개 버튼을 배치 : 기능 구현
  - "패딩 증가", "패딩 감소", "색상 변경" 버튼
    - 사각형 크기 및 색상 변경

- 구현 과정: 동영상(**Resize Rectangle App**.)

Practice 2: 패딩 크기 조절 기능 구현

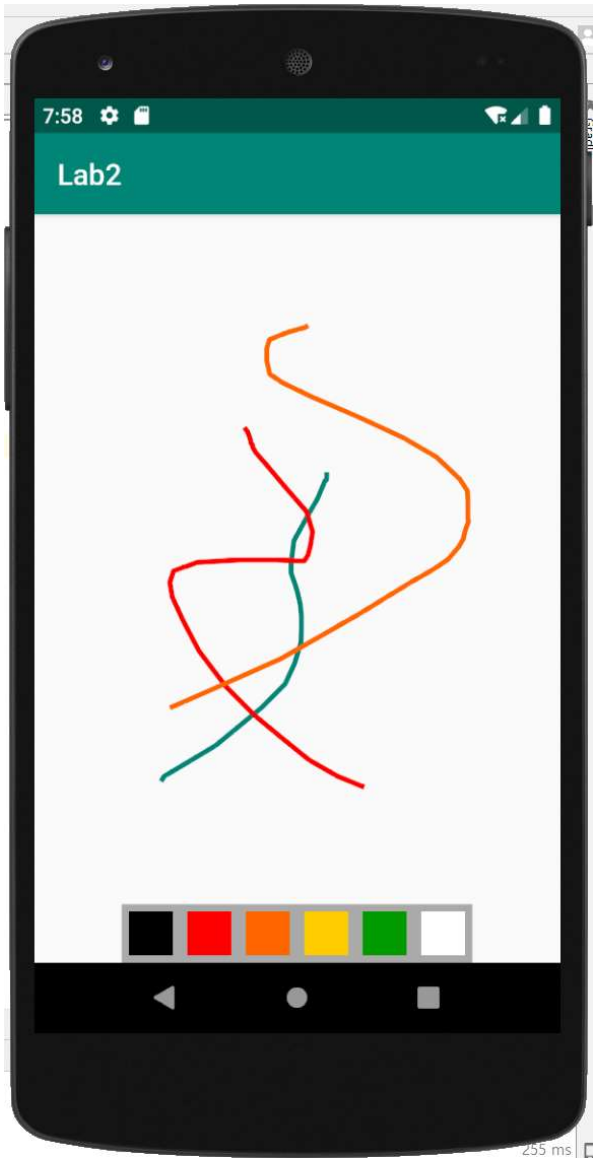
- 색상 변경 기능은 구현되어 있으나, 패딩 증가 및 패딩 감소에 따른 도형 크기 조절은 미완성.
  - onDraw 메소드 코드를 수정 하시오.

```
override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    var mRect = Rect()
    mRect.left = 0
    mRect.right = width
    mRect.top = 0
    mRect.bottom = height
    canvas.drawRect(mRect, mPaint)
}
```

What to do next?

- Lab 1. key를 눌러 사각형 이동
- Lab 2. 자유 곡선 그리기
- Lab 3. 도형 크기 조절
- **Lab 4. 색상 선택이 가능한 곡선 그리기**
- Lab 5. 볼륨 컨트롤 View 만들기

Lab 4: 색상 선택이 가능한 곡선 그리기



Lab 4: 색상 선택이 가능한 곡선 그리기

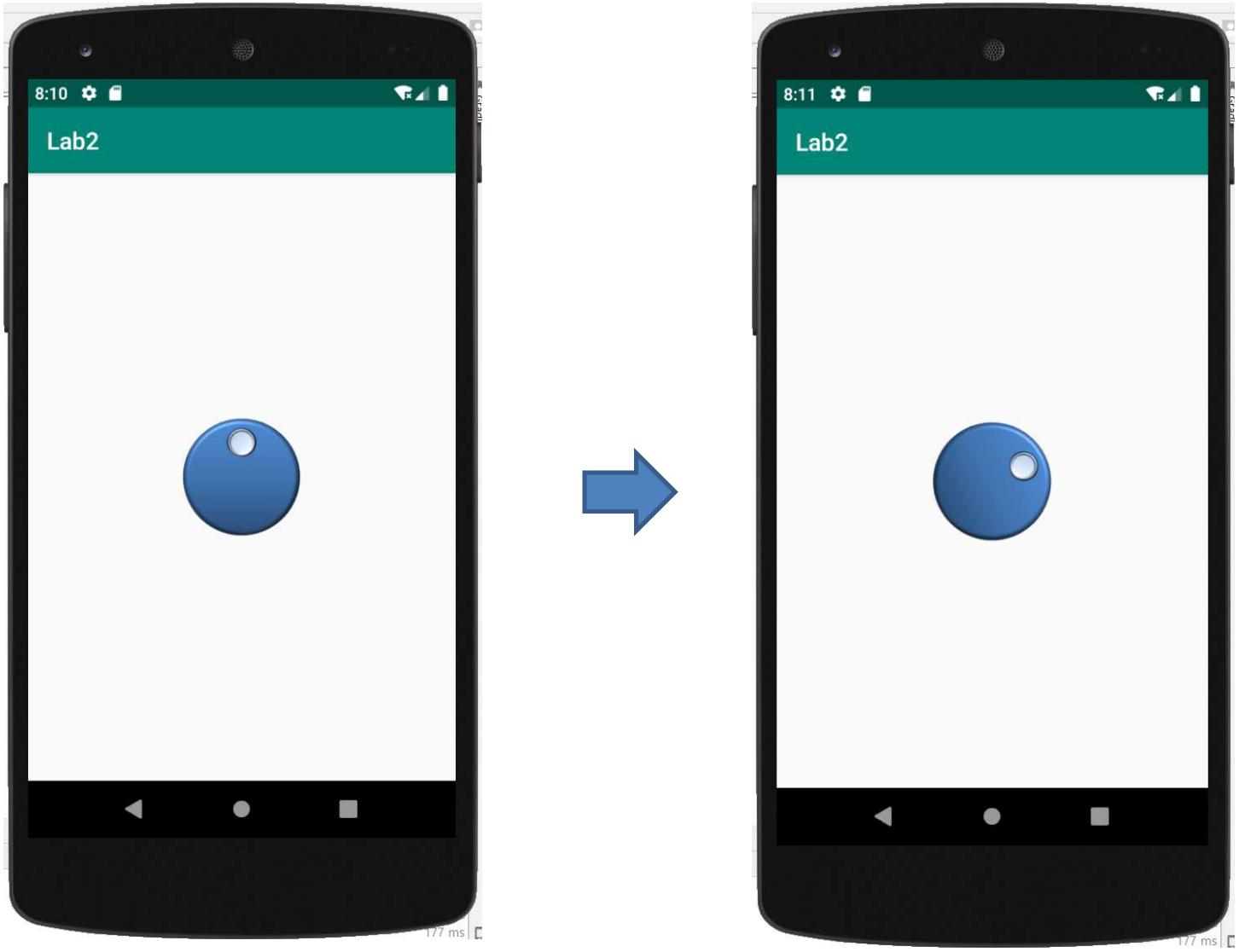
- XML Layout 사용
  - XML Layout에 custom view를 포함
  - 6개 ImageButton 배치 : 기능 구현
    - 6개 색상 선택 가능
  - Lab.2 를 확장시킨 예
    - Lab.3를 코드를 참고해 보자!

- 구현 과정: 동영상(**Simple Paint App**.)

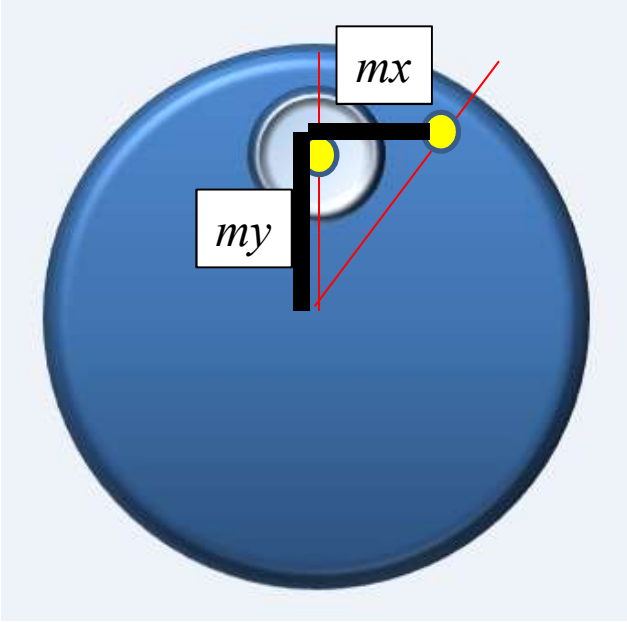
What to do next?

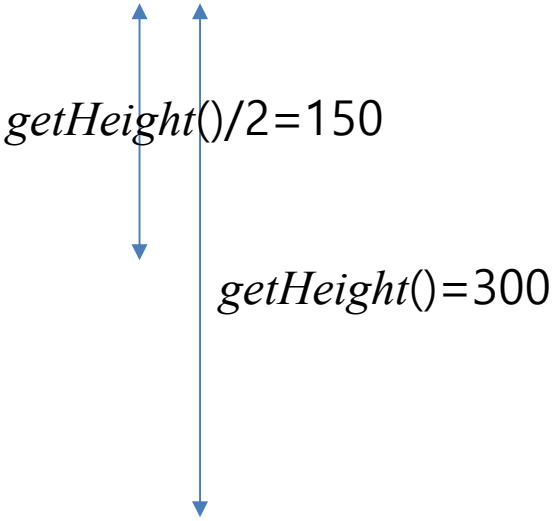
- Lab 1. key를 눌러 사각형 이동
- Lab 2. 자유 곡선 그리기
- Lab 3. 도형 크기 조절
- Lab 4. 색상 선택이 가능한 곡선 그리기
- **Lab 5. 볼륨 컨트롤 View 만들기**

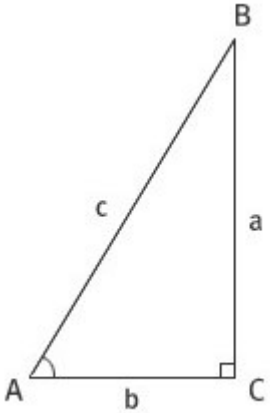
Lab 5: 볼륨 컨트롤 View 만들기



참고 : 코드 설명

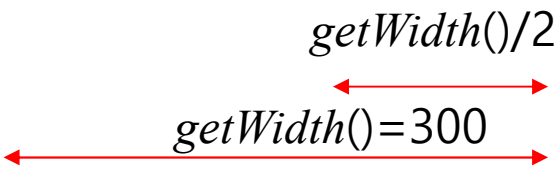






$$tan\theta = \frac{a}{b}$$

$$\theta = \tan^{-1}\frac{a}{b}$$



$$degree = \frac{180}{\pi} \times \theta[\text{rad}]$$

Lab 5: 볼륨 컨트롤 View 만들기

- XML Layout에 custom view 추가
- VolumeControlView.kt 파일
  - 이미지 파일 knob.png
  - 회전 각도에 따라 Knob 이미지 회전

- 구현 과정: 동영상(**Volume Control App**.)