헬로, 안드로이드

5주차 – 2D 그래픽 배우기 (1)

강대기
동서대학교 컴퓨터정보공학부

학습 목표

- 2D 그래픽에 대해 배운다.
- Color, Paint, Canvas, Path, Drawable 클래스를 배운다.
- 스도쿠의 게임 시작하기를 구현하고, Game 클래스, PuzzleView 클래스를 정의한다.
- 정의된 클래스 내에서 보드를 그리고, 숫자를 그린다.
- 숫자 입력, 힌트 추가, 화면 흔들기를 구현해 본다.

차례

- Color 클래스
- Paint 클래스
- Canvas 클래스
- Path 클래스
- Drawable 클래스
- 스도쿠의 게임 시작하기
- Game 클래스
- PuzzleView 클래스
- 보드 그리기
- 숫자 그리기
- 셀렉션 정의 및 업데이트
- 숫자 입력하기
- 힌트 추가하기
- 화면 흔들기

Color 클래스

- 색 – 알파(alpha), 레드(red), 그린(green), 블루(blue) – 32 비트 정수
- 알파 – 투명도
  - int color = Color.BLUE; // 파란 색
  - color = Color.argb(127,255,0,255); // 반투명 보라색
- XML 리소스에서 색상 정의
  - <resources>
  - <color name="mycolor">#3500ffff</color>
  - </resources>
- 자바 코드 내에서 사용
  - color = getResources().getColor(R.color.mycolor);

Paint 클래스

- 비트맵, 텍스트, 기하학적 모양의 그래픽을 그리는 데 필요한 스타일과 색상 정보를 가짐
- 화면에 칠을 할 때 필요한 색상 설정
  - cPaint.setColor(Color.LTGRAY);

Canvas 클래스

- 그림이 그려지는 표면
- Activity > View > Canvas
- 캔버스에 그리려면, View.onDraw() 를 오버라이드

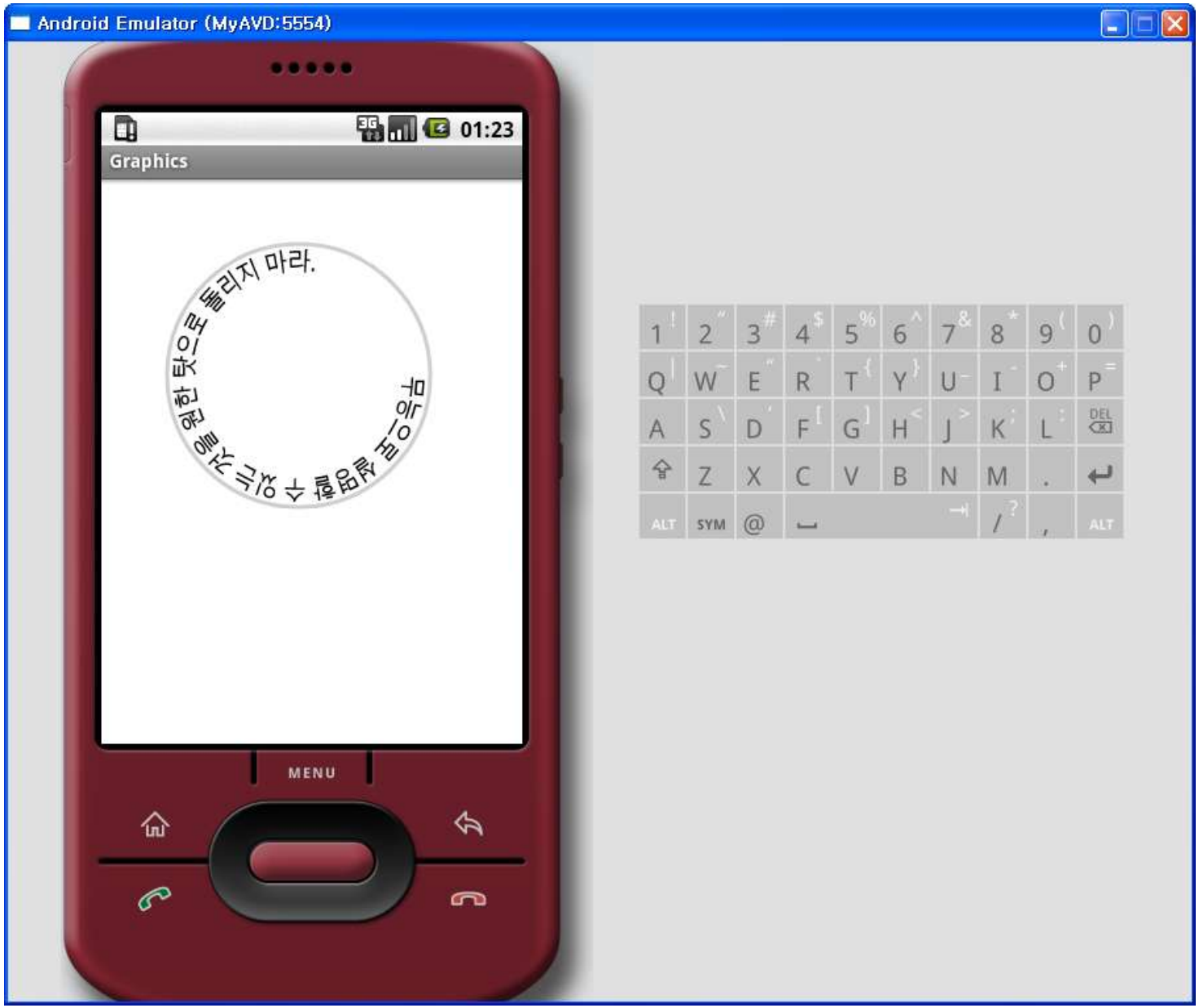
```
public class Graphics extends Activity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
}

static public class GraphicsView extends View {
  public GraphicsView(Context context) {
    super(context);
  }

  @Override
  protected void onDraw(Canvas canvas) {
    // 사용자의 그리기 코드가 여기에 추가됨
  }
}
```

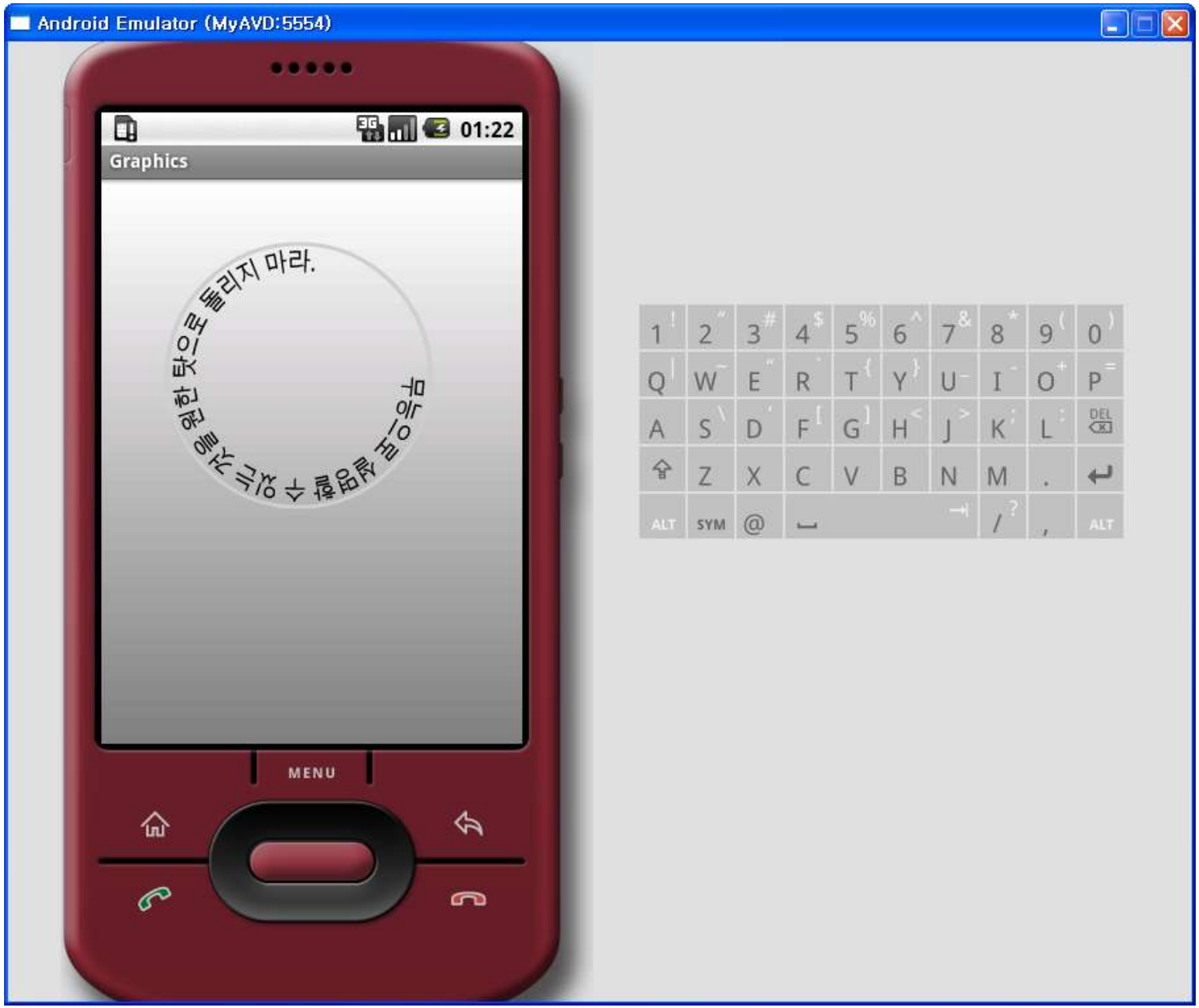
Path 클래스

- 선, 사각형, 커브 등 벡터 그리기 명령어를 포함
- 원형 경로 정의 – CW는 시계 방향
  - circle = new Path();
  - circle.addCircle(150,150,100,Direction.CW);
- 텍스트 추가
  - private static final String QUOTE = “무능으로 설명할 수 있는 것을 원한 탓으로 돌리지 마라.”;
  - canvas.drawPath(circle, cPaint);
  - canvas.drawTextOnPath(QUOTE, circle, 0, 20, cPaint);


Android Emulator (MyAVD:5554)
01:23
Graphics
원한 탓으로 돌리지 마라.
무능을 변명할 수 있는 것은
MENU

Drawable 클래스

- 비트맵, 단색 등과 같이 화면 표시용 시각적 요소에 사용됨
- 다른 그래픽에 결합시킬 수 있으며, 사용자 인터페이스 위젯(버튼이나 뷰의 배경)에 사용됨
- Drawable 의 형태
  - 비트맵(Bitmap) – PNG 또는 JPG 이미지
  - 나인패치(NinePatch) – 늘어나는 PNG 이미지
  - 모양(Shape) – 경로에 기초한 벡터 그리기 명령어, Scalable Vector Graphics (SVG) 의 축소판
  - 레이어 (Layer) – Z-order 순서에 따라 서로 덮어 그리는 하위 drawable을 포함하는 컨테이너
  - 상태 (State) – 상태(비트 마스크; bit mask)에 맞는 하나의 하위 drawable을 보여주는 컨테이너
  - 레벨 (Level) – 컨테이너
  - 스케일 (Scale) – 컨테이너


Android Emulator (MyAVD:5554)
01:22
Graphics
MENU

스도쿠의 게임 시작하기

- 이제 게임 구현 자체를 보도록 하자.
- startGame() – Game 이라는 액티비티를 위해 새 인텐트를 만들고 extraData 영역에 난이도 수준 숫자를 넣고 startActivity() 호출로 액티비티 시작

```
/** Start a new game with the given difficulty level */
 private void startGame(int i) {
   Log.d(TAG, "clicked on " + i);
   Intent intent = new Intent(Sudoku.this, Game.class);
   intent.putExtra(Game.KEY_DIFFICULTY, i);
   startActivity(intent);
 }
```

Game 클래스

- Sudoku4/src/org/example/sudoku/Sudoku.java
- 인텐트에서 난이도의 숫자를 가져와 이에 적합한 퍼즐을 고름
- PuzzleView 클래스 인스턴스 생성하여 뷰의 새 콘텐츠로 설정 (XML에서 하지 않고 코드에서 함)
- calculateUsedTiles() 메서드 – 수도쿠 규칙을 사용해서 각 타일에 더 이상 사용될 수 없는 숫자를 알아냄
- Game 클래스는 액티비티이므로 AndroidManifest.xml 에 등록해 놓음

PuzzleView 클래스

- 생성자 안에서 뷰의 폭과 높이를 사용하지 말 것
  - onSizeChanged() 메서드에서 값이 정해지면 받음
  - onDraw() 메서드에서 getWidth(), getHeight() 메서드 사용
- 생성자 안에서 Game 클래스의 참조를 저장
- 뷰에서 사용자 입력이 허용되도록 옵션 설정
- onSizeChanged() 메서드는 안드로이드 운영체제가 모든 것의 크기를 파악한 후 호출됨 – 여기서 화면의 각 타일의 크기를 계산함

그래픽 처리 전략

- 안드로이드 운영체제는 뷰가 수정될 때마다 onDraw() 메서드를 호출함 – 화면 전체를 다시 만든다고 가정하는 것이나, 실은 작은 부분만 다시 그림 – 안드로이드 운영체제가 부분 선택(clipping)을 대신 해줌
- 다음과 같이 여러 방법을 시도해 보았음
  - 버튼을 각 타일에 사용해 봄
  - ImageView 클래스의 그리드를 XML 안에 선언해 봄
  - 결국, 전체 퍼즐을 뷰 하나에 만들고, 그 안에 선과 숫자를 그려넣는 방식을 선택함
  - 다만, 선택 영역(selection)을 그려야 함과 키보드와 터치 이벤트를 명시적으로 처리해야 하는 등의 결점이 있음
  - 가능하다면 표준 위젯과 뷰를 사용해 보고, 불가피할 경우에 사용자 정의 방식을 선택하는 게 좋음

보드 그리기

- /res/values/colors.xml 에서 색상들 정의
- onDraw() 내에서 보드, 숫자, 힌트, 선택 영역 등을 순차적으로 그린다
- canvas.drawLine() 으로 보조 그리드 라인과 주 그리드 라인을 그리면 됨

숫자 그리기

- 각 숫자들이 타일에 정확히 중앙에 오도록 퍼즐 숫자를 집어 넣음
- 글자 높이를 타일 높이의 ¾로 설정하고, 가로 세로 비율을 동일하게 맞춤
- 가로 방향은 타일의 폭을 이등분함
- 세로 방향은 시작점을 약간 아래로 조정해야 함
  - 이를 위해 FontMetrics 클래스를 사용하여 한 글자가 얼마만큼의 세로 공간을 차지하는지를 알아내어, 그것을 이등분함
- canvas.drawText() 메서드를 사용하여 숫자를 넣음

입력 다루기

- 아이폰 프로그래밍의 장점은 폰이 하나로 고정되어 있어서, 화면의 가로 세로 크기를 계산하여 그래픽을 처리할 필요가 없다는 것 – 따라서, 아예 애플 측에서는 직접 고정된 상수로 입력하는 것을 선호한다는 말조차도 있음
- 안드로이드의 경우, 폰의 크기가 다양하고, 넷북일 수도 있으며, 디패드(키패드), 터치 스크린, 트랙볼 등의 입력도 있을 수 있음 – 따라서, 화면의 다양한 해상도는 물론 여러 입력 장치까지 지원해야 함

셀렉션 정의 및 업데이트

- 어떤 타일이 선택되었는지를 보여주는 작은 커서를 만들어 본다
  - 커서란 단순히 마우스 커서나 커맨드 프롬프트 커서가 아닌, 뭔가 초점의 대상인 것을 커서라고 부름
  - 이는 데이터베이스에서 레코드들을 가져오는 경우나 객체지향 프로그래밍에서 컨테이너를 다루는 경우에도 마찬가지임
- 선택된 타일 – 사용자가 숫자를 입력할 때, 변경되는 타일
- onDraw() 메서드에서 선택된 타일 표시
- onSizeChanged() 에서 미리 정의된 선택영역 사각형(selRect)을 사용하여 선택된 타일 위에 알파 값이 있는 색을 입힘 (selected)

셀렉션 정의 및 업데이트

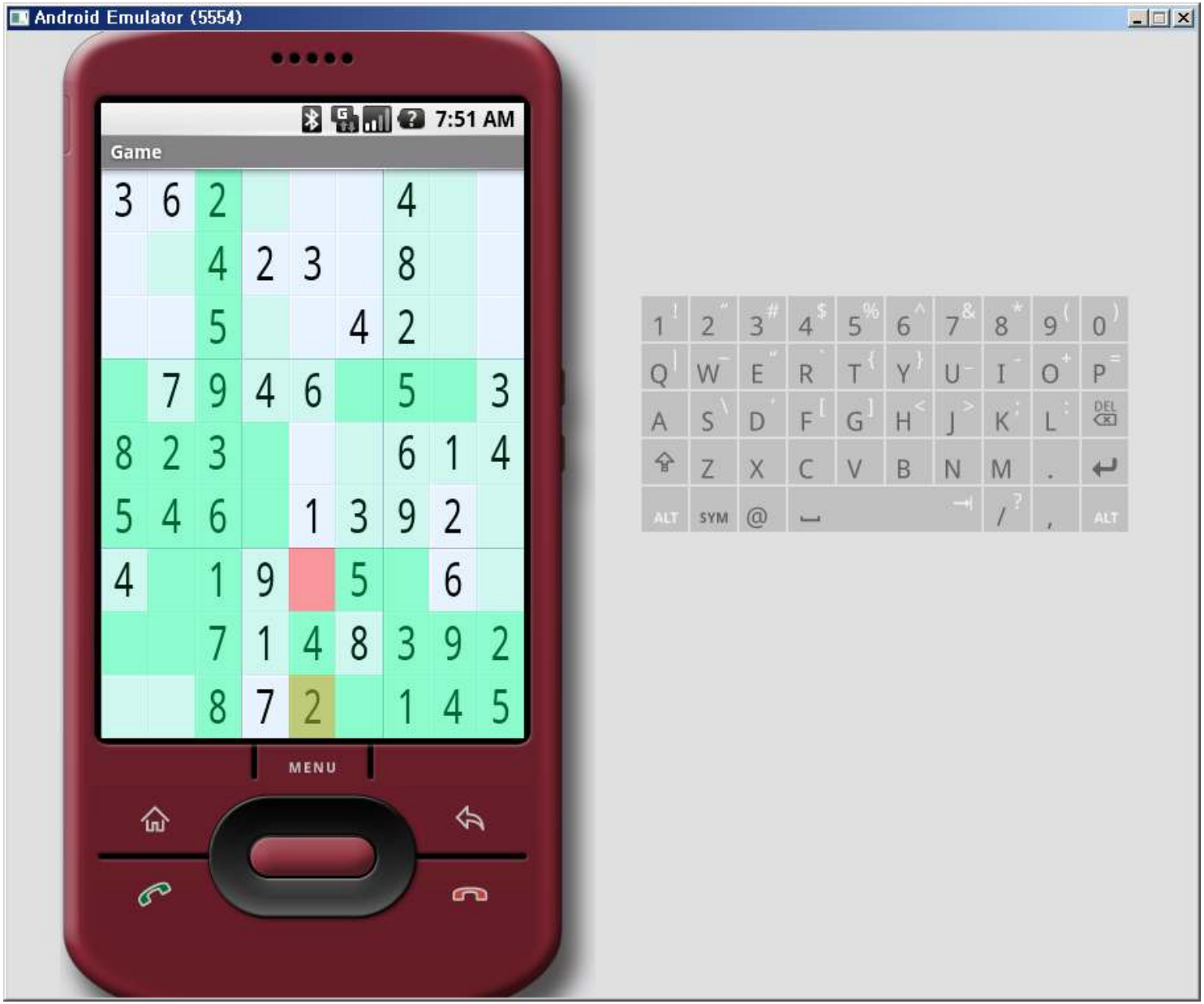
- onKeyDown() 메서드에서 선택 영역을 옮김 – 이를 통해 디패드의 키 이벤트 처리
- 트랙 볼의 경우? → onTrackBallEvent() 를 오버라이드하면 되지만, 만일 오버라이드 안하면 안드로이드는 자동으로 디패드 이벤트로 해석함
  - 대부분 이런 경우가 많은 데, 예를 들어, 대부분의 컴퓨터에서 바코드 리더기를 통한 입력의 경우에도, 드라이버를 통해 자동으로 키보드 입력으로 해석됨
- onDraw() 밖에서는 드로잉 함수를 절대 호출할 수 없음 – 대신 invalidate() 호출을 사용하여 다시 그려져야 됨을 알림 (더티 영역 – dirty area)
  - select() 함수에서 invalidate() 는 이전 선택 영역 사각형과 새로운 선택 영역 사각형에 대해 두 번 호출됨
- 안드로이드의 윈도 매니저는 특정 시점에서 더티한 사각형들을 모두 합해서 onDraw() 를 다시 호출함

숫자 입력하기

- 숫자 입력을 위해 0부터 9까지의 숫자를 다루는 case 문을 추가함
- 0과 스페이스는 숫자를 지우는 것을 의미하게 함
- 사용자가 디패드를 사용하는 경우, onKeyDown() 을 오버라이드하여 디패드 중앙 버튼이나 엔터 버튼의 입력을 확인하고 키패드 창을 띄워 숫자를 고르게 함
- 터치 스크린에서는 onTouchEvent() 메서드를 오버라이드하여 동일한 키패드를 사용자에게 보여줌
- 어떤 방식을 쓰든 타일 위의 숫자를 바꾸려면, setSelectedTile() 메서드 호출
- setSelectedTile() 메서드에서는 invalidate()를 호출하되 매개 변수는 없음 – 즉 화면 전체를 다시 그림 – 이유는 새로 추가된 숫자나 새로 지워진 숫자가 사용자에게 주어질 힌트를 변경하게 되어 다른 타일들의 배경색이 달라질 수 있기 때문임

힌트 추가하기

- 퍼즐을 풀어주지는 않지만, 사용자를 도와주기 위해 배경색을 다르게 함
  - 빈 타일에 들어갈 수 있는 숫자들의 경우의 수에 따라 타일 배경색을 다르게 함
- 만일 사용자가 실수를 저지른 경우, 그에 따라 해당 타일에 다른 배경색을 설정함


Android Emulator (5554)
7:51 AM
Game
3 6 2 4
4 2 3 8
5 4 2
7 9 4 6 5 3
8 2 3 6 1 4
5 4 6 1 3 9 2
4 1 9 5 6
7 1 4 8 3 9 2
8 7 2 1 4 5
MENU

화면 흔들기

- 사용자가 이미 사용된 숫자를 집어넣는 것과 같은 명백한 실수를 하는 경우, 화면을 앞뒤로 흔들어 줌
- /res/anim/shake.xml 에 정의된 R.anim.shake 리소스를 불러와 실행해서 화면을 1 초에 10픽셀씩 좌우로 흔들어 줌
- 이 애니메이션의 실행 횟수와 속력, 가속도 등은 XML 내에 정의된 애니메이션 인터폴레이터(interpolator)에 의해 조정됨
  - 인터폴레이터 – /res/anim/cycle_7.xml

요약

- 2D 그래픽에 대해 배웠다.
- Color, Paint, Canvas, Path, Drawable 클래스를 배웠다.
- 스도쿠의 게임 시작하기를 구현하고, Game 클래스, PuzzleView 클래스를 정의해 보았다.
- 정의된 클래스 내에서 보드를 그리고, 숫자를 그려 보았다.
- 숫자 입력, 힌트 추가, 화면 흔들기를 구현하였다.

퀴즈

- 화면에 칠할 때 필요한 색상을 설정하려면 어떤 클래스들을 이용해야 하는가?
- 화면에 그림이 그려지는 표면은 어떤 클래스인가?
- Path 클래스의 역할을 설명하라.
- Drawable 한 것들을 아는 대로 열거하라.
- XML로 콘텐츠를 설정하지 않고, 코드로 설정하는 경우의 장점은 무엇인가?
- 생성자에서 뷰의 폭과 높이를 사용하지 말아야 할 이유는 무엇인가?
- Activity 클래스의 onSizeChanged() 메서드에서 할 수 있는 일들은 무엇인가? 아는대로 쓰라.
- Activity 클래스의 onDraw() 메서드는 어떤 일을 하는가?
- Canvas 클래스의 drawLine() 메서드를 설명하라.
- Canvas 클래스의 drawText() 메서드를 설명하라.
- FontMetrics 클래스에 대해 설명하라.
- Activity 클래스의 onKeyDown() 메서드는 어떤 일을 하는가?
- Activity 클래스의 onTrackBallEvent() 메서드는 어떤 일을 하는가?
- invalidate() 메서드에서 매개 변수가 있을 때와 없을 때의 차이는 무엇인가?
- 애니메이션 인터폴레이터란 무엇인가?

연습문제

- 힌트로 들어갈 배경색을 임의로 바꾸어 보자
- 수도쿠 프로그램에서 화면 흔들기를 1초에 20 픽셀씩 좌우로 흔드는 것을 10회 반복하도록 고쳐보자
- 자신의 Hello Android 프로그램에서 화면 흔들기를 1초에 10 픽셀씩 좌우로 흔드는 것을 7회 반복하도록 해보자
  - 힌트: XML 파일에서 최상위 Layout에 대해 ID를 부여한 뒤, findViewById()로 해당 Layout의 View 참조를 가져와 멤버 변수로 저장한다
- 버튼을 타일 하나로 설정하는 방법으로 수도쿠 화면을 구현해 보라
- ImageView 클래스를 통해 수도쿠 화면을 구현해 보라

프로젝트 아이디어

- Nonomino Sudoku를 구현하는 프로젝트를 고려해 보자. Nonomino Sudoku는 다음에 소개되어 있다.
  - http://www.boldts.net/Sudoku/No/