SURFACEVIEW 활용하기

기초 원리 학습

OneKick

- SurfaceView
  - 표면이라는 뜻
  - 더블 버퍼링과 사용자 스레드를 이용해 그림 그리기 가능
  - 게임이나 카메라 미리 보기와 같은 연속적인 그리기가 필요한 작업에 사용
  - Android.view.View 패캐지에 포함
    - 주로 게임, 카메라 촬영 앱 구현 시 활용

- View와 SurfaceView의 차이점
  - View
    - Single Buffering
  - SurfaceView
    - Double Buffering
    - 고속 처리가 가능하고 높은 성능 발휘

기초 원리 학습

OneKick

- SurfaceView를 사용하기 위해 필요한 클래스
  - SurfaceView
    - 현재의화면으로 Android.view.SurfaceView를 임포트 해야 함
  - SurfaceHolder:
    - SurfaceView를관리하는클래스
    - SurfaceHolder.Callback()할 수 있으며, SurfaceView를 관리

기초 원리 학습

OneKick

- SurfaceView를 사용
  - Activity에SurfaceView를 상속받은 클래스로 화면 설정
  - SurfaceView클래스를 상속받아 새로운 클래스 생성

- SurfaceHolder클래스
  - SurfaceView를 관리하는 클래스
  - SurfaceHolder.Callback() 구현
    - surfaceCreated()
      - SurfaceView만들어질때호출
    - surfaceChanged()
      - SurfaceView변경될 때 호출
    - surfaceDestoryed()
      - SurfaceView제거될 때 호출

기초실습

- SurfaceView를 이용한 벽돌깨기 게임 만들기
  - 새 프로젝트 만들기
  - 게임뷰 만들기
    - 이름: GameView
    - 부모: SurfaceView
    - 인터페이스: SurfaceHolder.Callback

```
public class GameView extends SurfaceView
        implements  SurfaceHolder.Callback
```

  - 움직이는 공 Ball 클래스 구현하기
  - 게임을 진행 시킬 GameView 구현하기
    - 쓰레드를 이용한 게임의 진행
    - 충돌을 처리하는 물리엔진 구현하기

액티비티 수정하기

OneKick

```
package com.iot.surfaceviewtest;
import android.app.Activity;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Window;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);          // 타이틀바제거

        setContentView(new GameView(this));                 // XML은 필요 없음
    }
}
```

Ball 클래스 구현

- 화면을 돌아다니는 빨간 공 구현하기 (Ball.java -1/3)

```
package com.iot.surfaceviewtest;

import android.graphics.Canvas;
import android.graphics.ColorFilter;
import android.graphics.Point;
import android.graphics.Rect;
import android.graphics.drawable.Drawable;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;

public class Ball {
        private Drawable        image = null;
        private Point           point = new Point();
        private Point           size = new Point();
        private Point           delta;

        public Drawable getImage() {
                return image;
        }
```

Ball.java -2/3

OneKick

```
public void setImage(Drawable image) {
        this.image = image;
}

public Point getPoint() {
        return point;
}

public void setPoint(Point point) {
        this.point = point;
}

public Point getSize() {
        return size;
}

public void setSize(Point size) {
        this.size = size;
}
```

Ball.java -3/3

OneKick

```
	public void draw(Canvas canvas) {
		image.setBounds(point.x, point.y, point.x+size.x, point.y+size.y);
		image.draw(canvas);
	}

	public void setDelta(int dx, int dy) {
		delta = new Point(dx,dy);
	}

	public void move(Rect surfaceFrame) {
		// X axis collision
		if(point.x + delta.x <0 ||
				point.x + delta.x +size.x>surfaceFrame.right) delta.x*=-1;
		else point.x +=delta.x;

		// Y axis collision
		if(point.y + delta.y <0 ||
				point.y + delta.y +size.y>surfaceFrame.bottom) delta.y*=-1;
		else point.y +=delta.y;
	}
}
```

GameView 구현하기 1/4

```
package com.iot.surfaceviewtest;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Point;
import android.graphics.drawable.Drawable;
import android.util.Log; // 로그 출력 목적
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class GameView extends SurfaceView implements
                                    SurfaceHolder.Callback {

    private static final String TAG = GameView.class.getName();
    private final SurfaceHolder holder;
    private boolean goOnPlay = true;
```

GameView 구현하기 2/4

OneKick

```
private Thread renderer = new Thread() {

        @Override // 부모 메소드 재정의
        public void run() {
                super.run();

                Drawable bg = getResources().getDrawable(R.drawable.bg_space);
                bg.setBounds(holder.getSurfaceFrame());
                ball.setDelta(15,30);

                while(goOnPlay) {
                        ball.move(holder.getSurfaceFrame());
                        Canvas canvas = holder.lockCanvas();
                        bg.draw(canvas);
                        ball.draw(canvas);
                        holder.unlockCanvasAndPost(canvas);
                }
        }
};
```

GameView 구현하기 3/4

OneKick

```
private Ball ball;

public GameView(Context context) {
        super(context);

        Log.i(TAG,"GameView created");
        holder = getHolder();
        holder.addCallback(this);
}

@Override // 부모 메소드 재정의
public void surfaceCreated(SurfaceHolder holder) {
        renderer.start();
        ball = new Ball();

        ball.setImage(getResources().getDrawable(R.drawable.red_ball));
        ball.setSize(new Point(100, 100));
        ball.setPoint(new Point(0,0));
}
```

GameView 구현하기 4/4

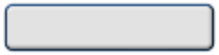
```
        @Override // 부모 메소드 재정의
        public void surfaceChanged(SurfaceHolder holder, int format, int width, int
        height) {

        }
        @Override // 부모 메소드 재정의
        public void surfaceDestroyed(SurfaceHolder holder) {
                goOnPlay = false; // kill thread
        }
}
```

동작 테스트

- 공이 경계선에 부딪힐 때 마다 반대 방향으로 튕기며 돌아다니는 것을 볼 수 있다.
  - SurfaceView는 물리적으로 화면에 접근해서 이미지를 그리기 때문에 움직임이 상당히 부드럽게 나타난다.
  - 만약 AVD 에서 가상화 설정이 되이 있지 않다면 실제 스마트폰 처럼 매끄러운 움직임을 보여주지 못한다

심화 학습

- 사각형 블록을 하나 추가하고 부딪혔을 때 벽돌이 사라지도록 구현해 보자
  - 이때는 Rect 클래스의 contains() 메소드를 이용하면 쉽게 충돌검사 가능



contains() ? **false**

contains() ? **true**