SURFACEVIEW & CAMERA

2020년도 1학기 앱 프로그래밍 12주차 수업

목 차

1교시

2020년도 1학기 앱 프로그래밍

– SURFACEVIEW

1. VIEW CLASS PROBLEMS

- Android View Class
  - Canvas 에 draw(그리기) 작업 수행
  - 그리기 작업
    - onDraw() 메서드 → 그리기 과정에 자동으로 더블 버퍼링으로 깜박거림 없음
    - Main Thread(UI Thread) 사용 그리기 수행
      - 속도가 빠르지 못하고 그리는 작업 동안 사용자의 입력 처리 없음 → 반응성 나쁨!
      - 게임과 같이 화면전환 속도가 빠르거나 지도와 같이 복잡한 작업에 불리함!
        - 그리기 작업 동안 마치 스레드가 멈추는 것처럼 보여 일시적인 UI 입력 반응을 처리 못하는 경우 발생!
    - 그리기 작업을 스레드로 분리할 수 없음!
      - 메인 스레드가 아닌 스레드는 뷰 또는 캔버스에 직접적인 접근이 불가능하기 때문
      - Main Thread가 관리하는 View class의 onDraw() 호출 → invalidate() 사용
      - 다른 스레드에서 View 클래스의 onDraw() 호출 → postInvalidate() 사용
        - 지난 학기 안드로이드 기초 시간에 View 클래스와 스레드 시간에 학습!

2. SURFACEVIEW CLASS

- SurfaceView Class → View 클래스의 문제점 해결
  - 복잡한 그리기 수행
  - 게임 등의 고속 이미지 처리를 위해 제공하는 장치
  - 독립된 스레드가 캔버스에 액세스 하지 못하도록 금지
    - 이유 : 만약 두 개의 독립된 스레드가 동시에 UI 스레드에 출력을 요청한 경우 문제 발생 때문!
- SurfaceView
  - 일반 View의 Canvas가 아닌 표면 뷰 즉 SurfaceView 를 사용
  - Surface → 메모리 상에 구성한 가상 화면
    - 메모리이므로 스레드에서 작업 내용을 직접 출력 가능!
    - 독립된 스레드가 Surface(메모리)에 그리는(출력) 동안 메인 스레드 UI 작업 가능!
    - 표면은 메모리에 존재할 뿐 View와 구조가 같기 때문에 캔버스 출력과 같은 방법으로 출력!

2. SURFACEVIEW CLASS (계속)

- Surface 사용 방법
  - SurfaceView 클래스 상속 → 사용자 정의 View 클래스가 된다!
    - View 클래스 생성 이후 Surface (표면) 생성 필요!
      - View를 관리자에 등록하고 레이아웃 결합을 완료하여 크기와 위치 결정된 후 생성됨!
    - 주의 사항
      - Thread는 표면이 준비된 이후 그리기 작업을 처리해야 함!
      - 표면이 파괴되면 그리기 작업을 처리해서는 안됨!
  - SurfaceHolder.Callback 인터페이스 구현
    - 메인 스레드는 표면의 변화를 통지 받고 스레드에게 그리기 허용 여부를 알려주어야 함!
    - 인터페이스의 추상 메서드 오버라이드 수행

override methods

1. void surfaceCreated(SurfaceHolder holder)
2. void surfaceDestroyed(SurfaceHolder holder)
3. void surfaceChanged(SurfaceHolder holder, int format, int width, int height)

2. SURFACEVIEW CLASS (계속)

- surfaceCreated()
  - 표면 생성 시 자동 호출
  - 메서드 호출 이후 표면에 그리기 허용
  - 표면에 하나의 스레드로 그리기 수행 가능
    - 만약 메인 스레드에서 콜백 구현 시 별도의 그리기 스레드가 필요함
- surfaceDestroyed()
  - 표면 파괴 시 호출
  - 이 메서드의 반환시 표면이 유효하지 않음
  - 더 이상 그리기 작업을 수행하면 안됨
  - 스레드에 그리기를 즉시 종료하도록 처리해야 함
- surfaceChanged()
  - 표면의 색상 또는 포맷 변경 시 호출
  - 표면의 크기 초기화(매개변수 width, height 값으로 크기 조정)

2. SURFACEVIEW CLASS (계속)

```
public class CameraPreview extends SurfaceView implements SurfaceHolder.Callback {
    private SurfaceHolder mHolder;
    private Camera mCamera;

    public CameraPreview(Context context, Camera camera) {...}

    @Override
    public void surfaceCreated(SurfaceHolder holder) {...}

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {...}

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {...}
}
```

2. SURFACEVIEW CLASS (계속)

- SurfaceHolder 객체
  - 표면을 관리하는 주체
  - 표면의 크기와 색상 등을 관리하고 표면으로 출력 내보내기 처리

  - 내장 메서드
    - getHolder() 메서드 → holder 얻기
    - addCallback() 메서드 → 표면의 변화 감지(감지를 위한 콜백 객체 등록)
    - lockCanvas() → 출력을 바로 화면으로 보내지 않고 표면의 비트맵에 그리기 수행
    - unlockCanvasAndPost() → 표면 비트맵에 그려진 그림을 화면으로 보내어 그리기 표현

2. SURFACEVIEW CLASS (계속)

```
public CameraPreview(Context context, Camera camera) {
    super(context);
    mCamera = camera;

    // SurfaceHolder.Callback 설치 : Surface(표면)가 생성되거나 삭제될때 알림을 받기 위함
    mHolder = getHolder();
    mHolder.addCallback(this);
}
```

2교시

2020년도 1학기 앱 프로그래밍

– CAMERA

3. CAMERA CLASS

- Camera Class → 지원 중단 클래스
  - API 레벨 21 이상에서는 지원하는 Camera2 클래스 사용
    - 지원 중단은 추가 업그레이드 등의 지원이 없을 뿐 사용할 수 없는 것이 아니다!

- 학습 내용
  - Camera Class와 SurfaceView Class 그리고 SurfaceHolder.Callback Interface 관계
  - 사진 촬영을 위한 카메라 미리보기(Free View) 기능 구현
  - Permission 체크 기능 구현

3. CAMERA CLASS (계속)

- 카메라 관련 권한 작업 (AndroidManifest.xml)
  - 매니패스트에 카메라 하드웨어와 관련 기능 사용의 허용 설정 필요

- 권한 내용
  - 카메라 권한 → 애플리케이션이 기기 카메라를 사용할 권한 요청
  - 카메라 기능 → 애플리케이션이 카메라의 기능 사용 선언
  - 저장 권한 → 이미지 또는 동영상 저장을 위해 SD 저장소 접근 권한
  - 오디오 권한 → 영상과 오디오 녹음을 위한 권한
  - 위치 권한 → 위치 정보로 이미지를 태그하기 위한 경우 권한 요청

3. CAMERA CLASS (계속)

```
<uses-permission android:name="android.permission.CAMERA"/> <!-- 카메라 기능 설치시 허가권 확인 -->
<uses-feature android:name="android.hardware.camera"/> <!-- 카메라 기능 실행 시 접근 권한 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/> <!-- SD 저장 권한 확인 -->
<uses-permission android:name="android.permission.RECORD_AUDIO"/> <!-- 오디오 녹음 권한 -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/> <!-- 위치 태그를 위한 권한 -->
```

3. CAMERA CLASS (계속)

- 카메라 기능 구현을 위한 단계
  1. 카메라 권한 및 허가권 확인
     - 매니패스트 및 허가권 확인
  2. 카메라 감지
     - 카메라 유무 확인
  3. 카메라 액세스
     - 접근 요청 코드 작성
  4. 미리보기 클래스 만들기
     - SurfaceView 상속, SurfaceHolder 인터페이스 구현의 카메라 미리보기 클래스 코드 작성
  5. 미리보기 레이아웃 빌드
     - 미리보기, 사용자 인터페이스 제어를 위한 뷰 레이아웃 코드 작성
  6. 캡처를 위한 리스너 설정
     - 사용자 작업(촬영 버튼 누르기)을 감지하는 리스너 설정 코드 작성
  7. 캡처 및 파일 저장
     - 사진, 동영상 캡처 후 출력 데이터 저장을 위한 코드 작성
  8. 카메라 해제
     - 설정한 카메라 객체 및 기능 해제 코드 작성

3. CAMERA CLASS (계속)

1. 카메라 권한 및 허가권 확인 – 권한 상태 확인

```
@Override
protected void onResume() {
    super.onResume();

    //사용자의 OS 버전이 마시멜로우(23)이상인지 판별
    if(Build.VERSION.SDK_INT>= Build.VERSION_CODES.M) {
        //사용자의 단말에 권한 중 위치 가져오기(ACCESS_FINE_LOCATION)의 권한 허가 여부를 가져온다.
        //허가 -> PERMISSION_GRANTED
        //거부 -> PERMISSION_DENIED
        int permissionCheck = checkSelfPermission(Manifest.permission.CAMERA);
        int permissionCheck2 = checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION);

        //현재 어플리케이션이 권한에 대해 거부되었는지 확인
        if (checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION_DENIED ||
                checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_DENIED) {

            requestPermissions(new String[]{Manifest.permission.CAMERA, Manifest.permission.ACCESS_FINE_LOCATION}, requestCode: 1000);
        }
        else {
            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
            startActivity(intent);
            finish();
        }
    }
    else {
        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
        startActivity(intent);
        finish();
    }
}
```

3. CAMERA CLASS (계속)

1. 카메라 권한 및 허가권 확인 – 권한 요청 결과 처리

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    if(requestCode==1000){
        if(grantResults.length>0 &&
                grantResults[0]== PackageManager.PERMISSION_GRANTED &&
                grantResults[1] == PackageManager.PERMISSION_GRANTED){

            if(ActivityCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_DENIED &&
                    ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_DENIED){
                Intent intent=new Intent(getApplicationContext(),MainActivity.class);
                startActivity(intent);
                finish();
            }else{
                Toast.makeText( context: this,  text: "권한 요청을 거부하였습니다", Toast.LENGTH_SHORT).show();
            }
        }
        else {
            Toast.makeText(getApplicationContext(),  text: "권한 설정 실패!\n앱을 다시 설치하세요!\n앱 종료!", Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}
```

3. CAMERA CLASS (계속)

2. 카메라 하드웨어 감지

```
// 카메라 하드웨어 감지
private boolean checkCameraHardware(Context context){
    if(context.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA))
        return true; // 카메라 있음!
    else
        return false; // 카메라 없음
}
```

3. CAMERA CLASS (계속)

3. 카메라 액세스

```
public static Camera getCameraInstance(){
    Camera c = null;
    try {
        c = Camera.open();
    }
    catch (Exception e){}
    return c;
}
```

3. CAMERA CLASS (계속)

4. 미리보기 클래스 만들기 - 생성자

```
public CameraPreview(Context context, Camera camera) {
    super(context);
    mCamera = camera;

    // SurfaceHolder.Callback 설치 : Surface(표면)가 생성되거나 삭제될때 알림을 받기 위함
    mHolder = getHolder();
    mHolder.addCallback(this);
}
```

3. CAMERA CLASS (계속)

4. 미리보기 클래스 만들기 – surfaceCreated()

```
@Override
public void surfaceCreated(SurfaceHolder holder) {
    // Surface 생성, 미리보기를 그릴 위치 설정
    try {
        mCamera.setPreviewDisplay(holder);
        mCamera.startPreview();
    } catch (IOException e) {
        Log.d(TAG, msg: "Error setting camera preview: " + e.getMessage());
    }
}
```

3. CAMERA CLASS (계속)

4. 미리보기 클래스 만들기 – surfaceChanged()

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

    if (mHolder.getSurface() == null){
        // 미리보기를 위한 surface가 없는 경우
        return;
    }

    // 변경전 미리보기 중지!
    try {
        mCamera.stopPreview();
    } catch (Exception e){}

    // 미리보기 크기 설정, 크기 조정, 회전 또는 형식 변경 코드 위치
    mHolder.setFixedSize(width, height);

    // 새로운 설정으로 미리보기 시작
    try {
        mCamera.setPreviewDisplay(mHolder);
        mCamera.startPreview();

    } catch (Exception e){
        Log.d(TAG, msg: "Error starting camera preview: " + e.getMessage());
    }
}
```

3. CAMERA CLASS (계속)

4. 미리보기 클래스 만들기 – surfaceDestroyed()

```
@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    mCamera.release();
    mCamera = null;
}
```

3. CAMERA CLASS (계속)

5. 미리보기 레이아웃 빌드 – activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/camera_preview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1">

    <Button
        android:id="@+id/btn_capture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Capture!"
        android:layout_gravity="center|right"/>
    </FrameLayout>
</LinearLayout>
```

3. CAMERA CLASS (계속)

5. 미리보기 레이아웃 빌드 – MainActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // 카메라 인스턴스 얻고 멤버에 설정
    mCamera = getCameraInstance();

    if(checkCameraHardware( context: this)) {
        // 미리보기 뷰 생성 후 액티비티 내의 컨텐츠로 설정한다.
        mPreview = new CameraPreview( context: this, mCamera);
        FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
        preview.addView(mPreview);

        hideSystemUI();
    }
    else{
        Toast.makeText(getApplicationContext(), text: "장치에 카메라를 지원하지 않습니다.\n앱을 종료합니다!", Toast.LENGTH_SHORT).show();
    }
}
```

3교시

2020년도 1학기 앱 프로그래밍

– CAMERA FREEVIEW PROJECT

4. 앱 프로그래밍 학습을 위한 실습 (과제)

- 실습 및 예습 문제
  - 카메라 영상 미리 보기 화면의 전체 화면 모드 구현
    - App 타이틀 바 삭제 / Android 메뉴 바 삭제
  - 안드로이드 개발자 사이트에서 "full screen mode"로 검색
    - 스스로 학습하고 기능이 적용된 결과를 제출하세요!

- 제출 방법
  - http://ctl.gtec.ac.kr 의 과제 제출란에 제출
  - Java 파일과 xml 파일을 제출한다.
  - 압축파일은 자신의 "반_학번_이름.zip" 로 한다.
  - 제출 일자와 시간을 엄수하여 제출하세요!