4주차 개발일지

Processes and Thread

Processes and threads overview | Android Developers

When an application component starts and the application does not have any other components running, the Android system starts a new Linux process for the application with a single thread of execution. By default, all components of the same application run in the same process and thread (called the "main" thread).

https://developer.android.com/guide/components/processes-and-threads

application component가 시작을 하면 android system은 해당 component을 위한 single thread, Linux 프로세스를 시작한다.

▷android: process

하나의 application 안에 있는 component(activity, service, receiver, provider)는 고유한 공통 main process에서 구동한다. manifest 파일의 android: process 속성을 통해 해당 component를 전담하는 process를 지정할 수 있다.

Process이란

Process는 현재 실행 중인 program이다.

- program은 명령어 리스트를 내용으로 가진 디스크에 저장된 파일이다.
- process는 다음에 실행할 명령어를 지정하는 프로그램 카운터와 관련 자원의 집합을 자인 능동적인 존재이다.

Thread이란

애플리케이션이 시작되면 시스템이 애플리케이션에 대한 실행의 main thread를 생성한다. main thread는 드로어블 이벤트를 포함하여 적절한 사용자 인터페이스 위젯에 이벤트를 발송하는 역할을 맡기 때문에 main thread는 기본적으로 UI thread이다.

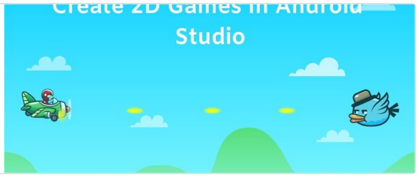
앱이 사용자 상호작용에 응답하여 리소스를 많이 소모하는 작업을 수행하는 경우, 이 단일 스레드 모델은 애플리케이션을 제대로 구현하지 않으면 낮은 성능을 보일 수 있습니다. 특히 모든 것이 UI 스레드에서 발생할 경우 네트워크 액세스나 데이터베이스 쿼리 등의 긴 작업을 수행할 때 전체 UI가 차단됩니다.

공부

How To Make 2D Games In Android Studio | Part 1

In this video series, we will learn to make 2d games in Android Studio
We will be creating a game where a flight has to shoot some evil birds and if the flight misses shooting any bird then the game will be over or if the bird hits the flight then also the game will be over

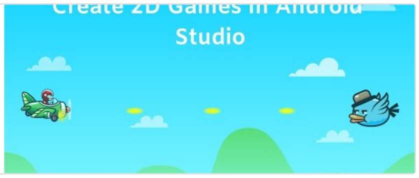
https://www.youtube.com/watch?v=aTT4GfojkHA



How To Make 2D Games In Android Studio | Part 2

In this video, we will create the background of the game Github Repo: https://github.com/rathour-mihir/2D-Game-In-Android-Studio

https://www.youtube.com/watch?v=RQoT9BUsl1Q&t=169s



surface view: 스킨의 content를 빠르게 바꿔야 할 때 사용한다.

오류

▷drawable에서 image 가져오

```
val image = org.texchtown.flightgame.R.drawable.background
background = BitmapFactory.decodeResource(res, image)
```

▷full screen

```
//set full screen
window.setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
    WindowManager.LayoutParams.FLAG_FULLSCREEN)
```

▷canvas를 이용해서 그리고 screen에 보이기

```
private fun draw() {
        //this is where the image for the screen is made
        // this is for ensuring that surfaceView is
        //successfully initialize
        if (holder.surface.isValid) {
            //getting the current screen
            val canvas: Canvas = holder.lockCanvas()

            //drawing the canvas to display on a screen
            //using drawBitmap
            //image, x axis, y axis, paint class is needed
            val rectangle1 = Rect(background1.x-screenX/2, 0, background1.x+screenX/2, screenY)
            val rectangle2 =Rect(background2.x-screenX/2, 0, background2.x+screenX/2, screenY)
            canvas.drawBitmap(background1.background, null, rectangle1,paint)
            canvas.drawBitmap(background2.background, null, rectangle2, paint)
            // show a canvas on a screen
            holder.unlockCanvasAndPost(canvas)
        }
    }
```

▷how to use drawBitmap

`https://developer.android.com/reference/android/graphics/Canvas#drawBitmap(android.graphics.Bitmap,%20android.graphics.Rect,%20android.graphi`

주의!! left < right , top < bottom

```
private val paint: Paint
paint = Paint()
val rectangle = Rect(left,top,right,bottom)
canvas.drawBitmap(Bitmap, null, rectangle, paint)
```

▷Runnable과 Thread 사용

Runnables Threads and Handlers in Android

In this video you will learn what is a #handler #thread and #runnable, and how to use each one of them.

⭐ Kite is a free AI-powered coding assistant that will help you code faster and smarter. The Kite plugin

https://www.youtube.com/watch?v=lsmJezQIOtI

```
public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";
    TextView textView;

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

        textView = findViewById(R.id.textView);
    }

    public void buttonClicked(View view) {

        final Handler handler = new Handler();

        Runnable runnable = new Runnable() {
            @Override
            public void run() {
                Log.i(TAG, "Thread Name 2: " + Thread.currentThread().getName());
                synchronized (this) {
                    try {
                        wait(5000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }

                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(MainActivity.this, "Download finished...", Toast.LENGTH_SHORT).show();
                    }
                });

                handler.postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(MainActivity.this, "10 seconds passed since download was finished...", Toast.LENGTH_SHORT).show();
                    }
                }, 10000);

                Log.i(TAG, "run: Download finished.");

            }
        };
//        runnable.run();
        Log.i(TAG, "Thread Name 1: " + Thread.currentThread().getName());
        Thread thread = new Thread(runnable);
        thread.start();

    }
}
```

▷Thread를 안전하게 멈추는 법

Java Multithreading Tutorial for Beginners #7: How to safely stop a thread ?

In this chapter, we will talk about properly closing a thread. The built-in method Thread.stop() is deprecated and we should implement our own method for closing a thread properly.

https://www.youtube.com/watch?v=w-qd_mQ3s1g

▷화면 터치를 인식하는 방법

MotionEvent | Android Developers

https://developer.android.com/reference/android/view/MotionEvent

```
//this is for moving the flight by touching the screen
override fun onTouchEvent(event: MotionEvent?): Boolean {
    when (event?.action) {
        //the screen is pressed
        MotionEvent.ACTION_DOWN-> if (event.x< screenX / 2) {
            //when left screen is touched make flight up
            flight.isGoingUp = true
        }
        //the screen press is released
        MotionEvent.ACTION_UP-> {
            flight.isGoingUp = false
            //when right screen is touched make the flight shoot
            if (event.x> screenX / 2) {
                flight.toShoot++
            }
        }
    }
    return true     //it has to return true to know just after the user touch the screen
}
```