Effectuer des traitements longs

Adrian Bracigliano

Exemples de traitements longs

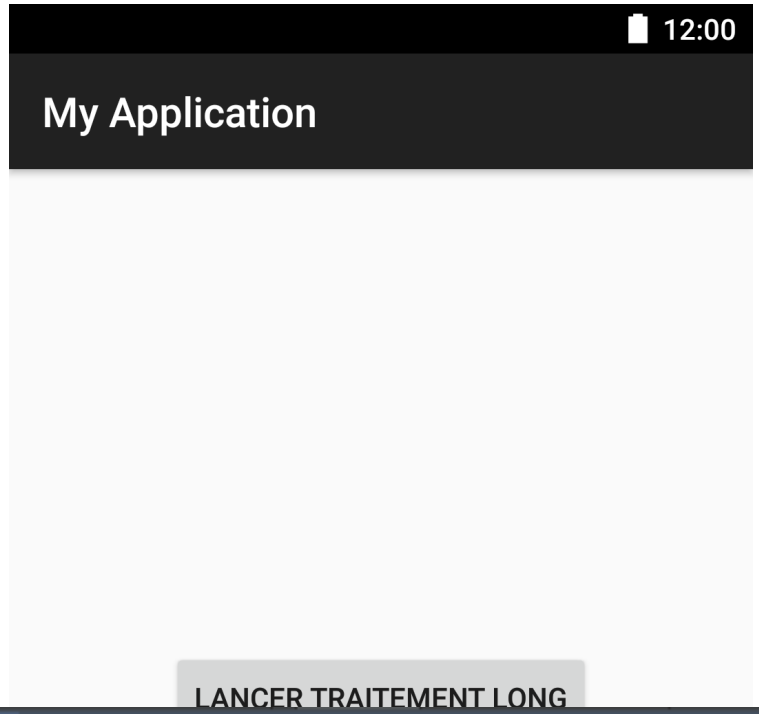
- Appels réseaux
- Ecritures/lectures dans base de données
- Traitements sur une image
- Calculs complexes
- etc…


12:00
My Application
LANCER TRAITEMENT LONG

Exécution d'un traitement long

```
@Override
public void onClick(View v) {
    button.setVisibility(View.GONE);
    progressBar.setVisibility(View.VISIBLE);

    longWorkToDo();

    button.setVisibility(View.VISIBLE);
    progressBar.setVisibility(View.GONE);
}
```


12:00
My Application
LANCER TRAITEMENT LONG



Run: AVD: Nexus_6_API_24 app

```
Connected to process 2957 on device emulator-5554
W/System: ClassLoader referenced unknown path: /data/app/tp3.ensiie.com.myapplication-2/lib/x86
W/art: Before Android 4.1, method android.graphics.PorterDuffColorFilter android.support.graphics.draw
W/gralloc_ranchu: Gralloc pipe failed

                  [ 11-21 06:06:04.877  2957: 2957 D/         ]
                  HostConnection::get() New Host Connection established 0xa343df00, tid 2957


                  [ 11-21 06:06:04.919  2957: 2972 D/         ]
                  HostConnection::get() New Host Connection established 0xa2a500c0, tid 2972
I/OpenGLRenderer: Initialized EGL, version 1.4
D/OpenGLRenderer: Swap behavior 1
I/Choreographer: Skipped 600 frames!  The application may be doing too much work on its main thread.
```

0: Messages | Terminal | 6: Android Monitor | 4: Run | TODO

Frame rate

- Nombre de fois qu'une image va être rafraichit par seconde
- Permet de donner une notion de mouvement

Frame rate

- Flipbook : 10 frames par seconde (fps)

Frame rate

- Film : 24 fps
- Application Android : 60 fps

  (1 frame ≈ 16 ms)

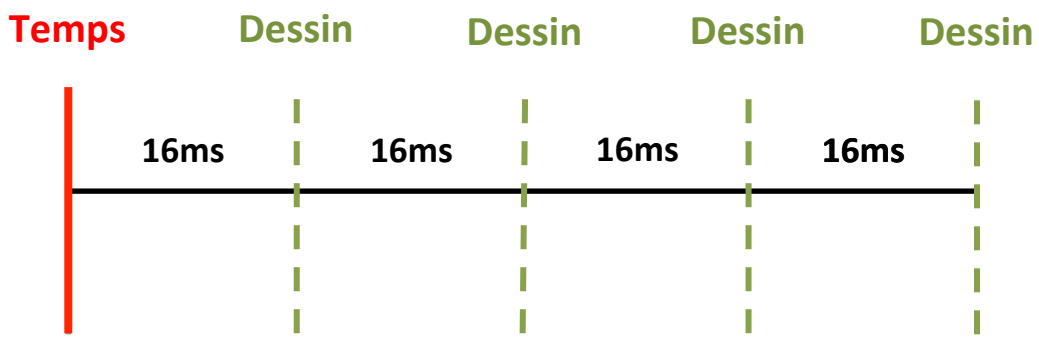
Rendu GPU du profil

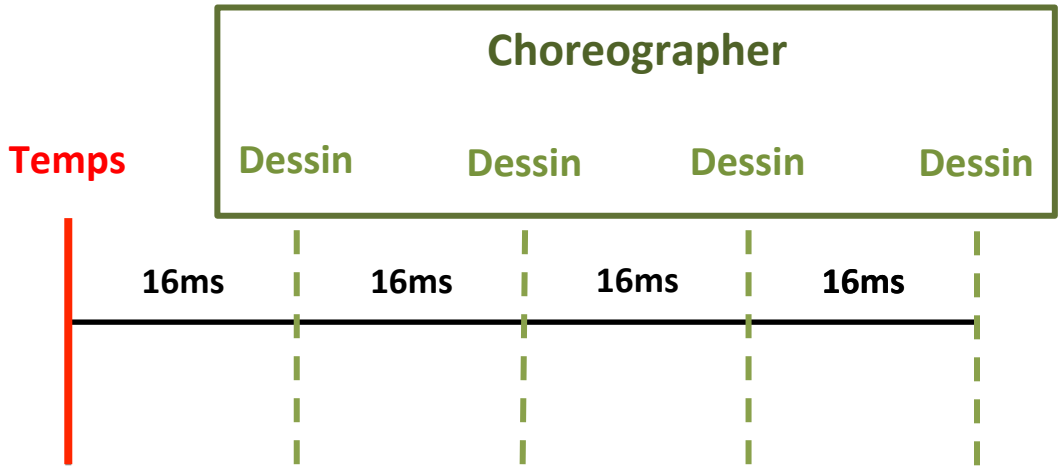
Options pour les développeurs →
Rendu GPU du profil → A l'écran sous forme de barres

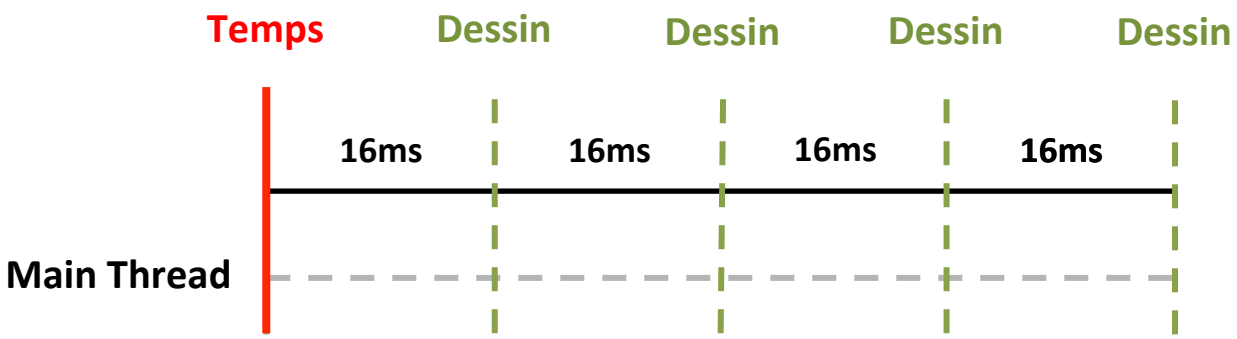
Frame dessinée en 16 ms ⟶

Main Thread

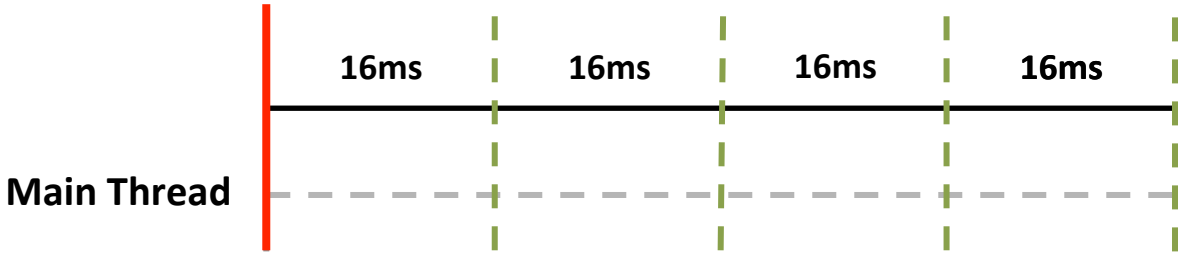
- Thread crée au lancement d'une application
- Par défaut, thread dans lequel s'exécute tout le code

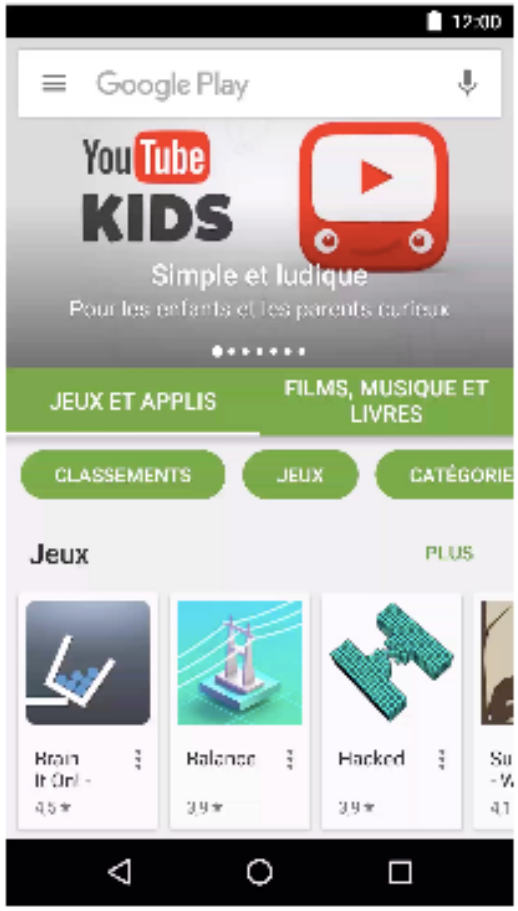

Temps
Dessin
Dessin
Dessin
Dessin
16ms
16ms
16ms
16ms


Choreographer
Temps
Dessin
Dessin
Dessin
Dessin
16ms
16ms
16ms
16ms

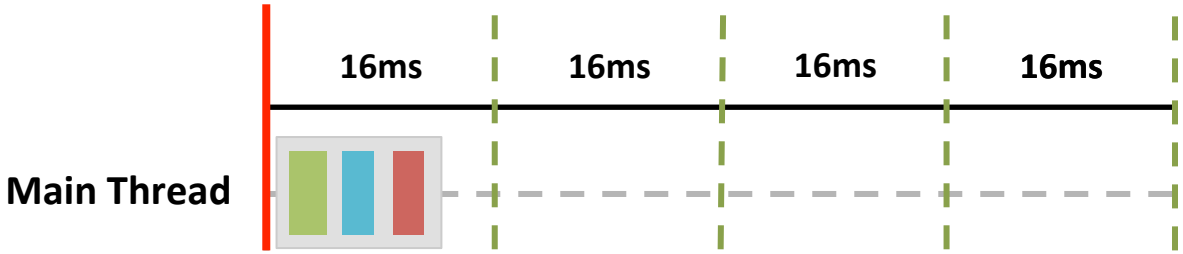

Temps
Dessin
Dessin
Dessin
Dessin
16ms
16ms
16ms
16ms
Main Thread

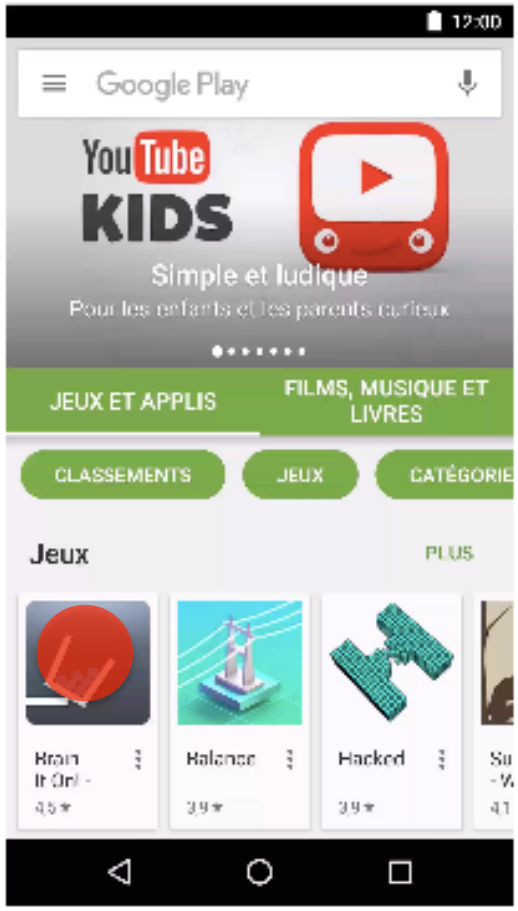
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
}
```



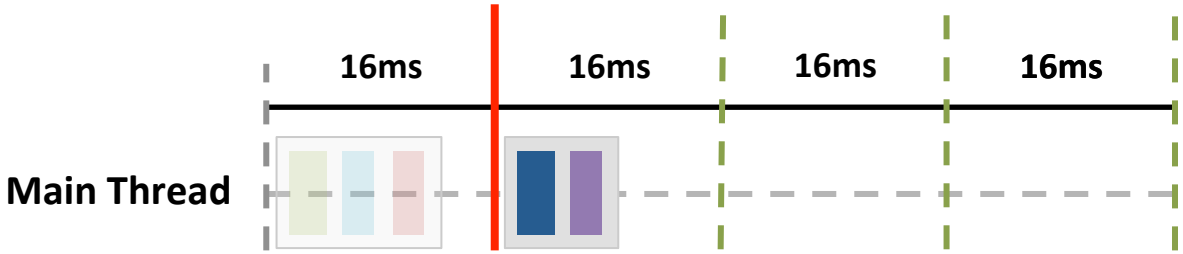


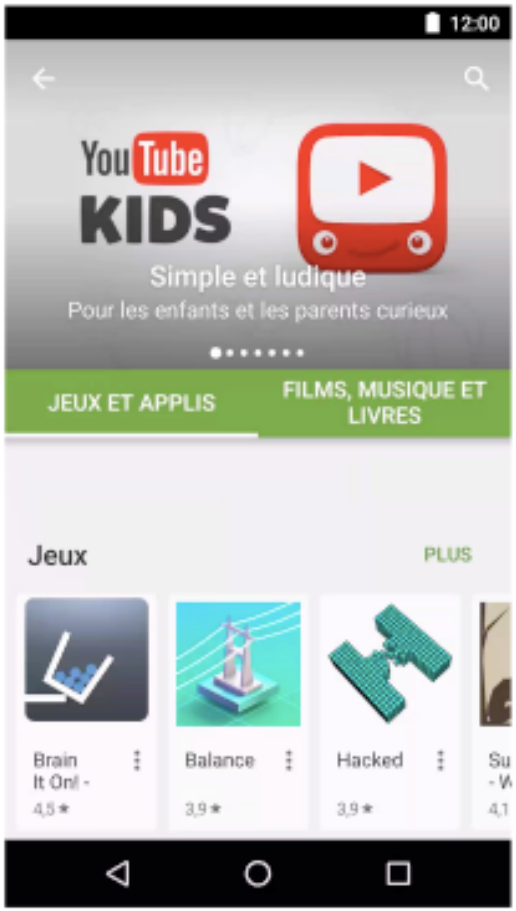
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
}
```



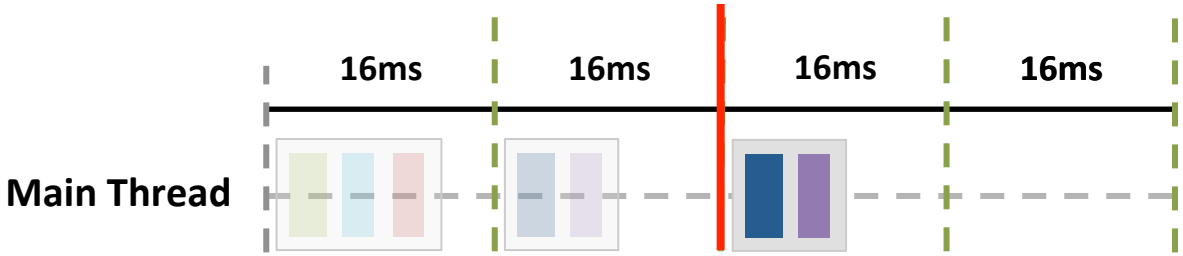


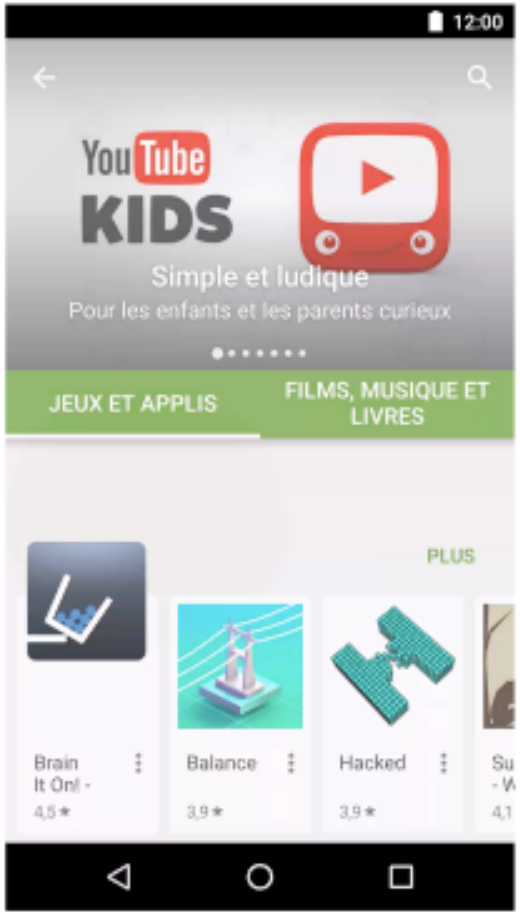
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
}
```



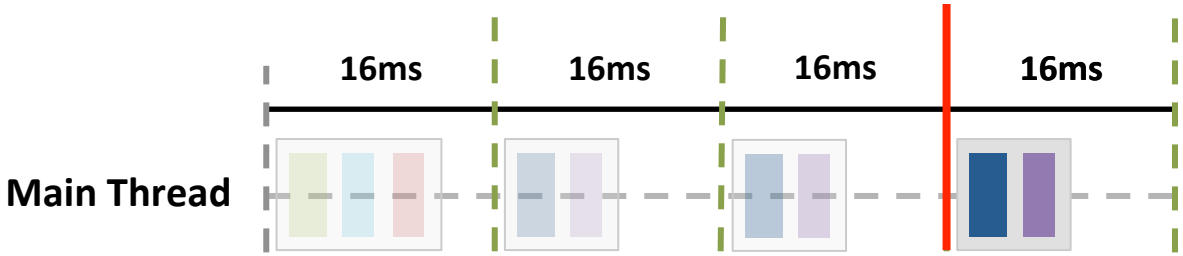


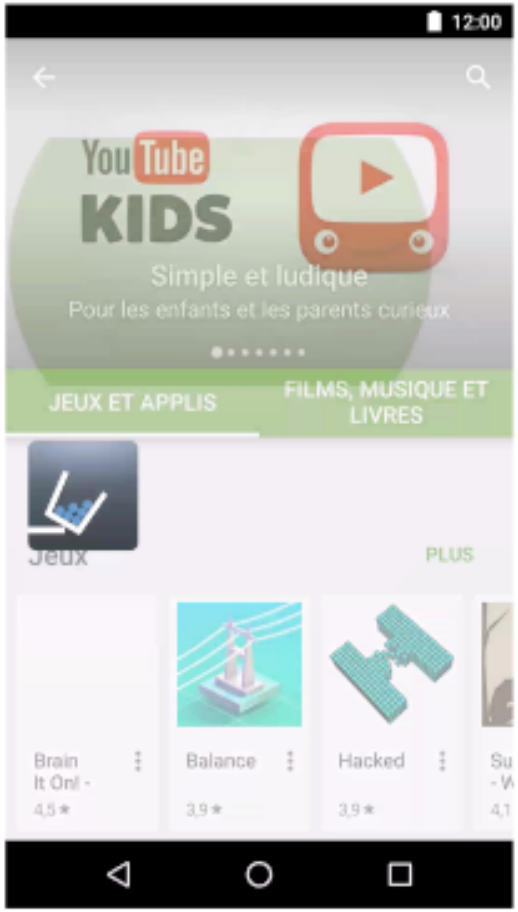
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
}
```



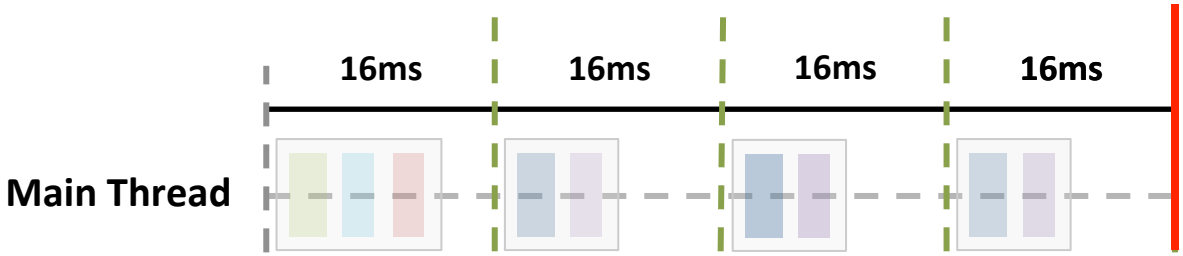


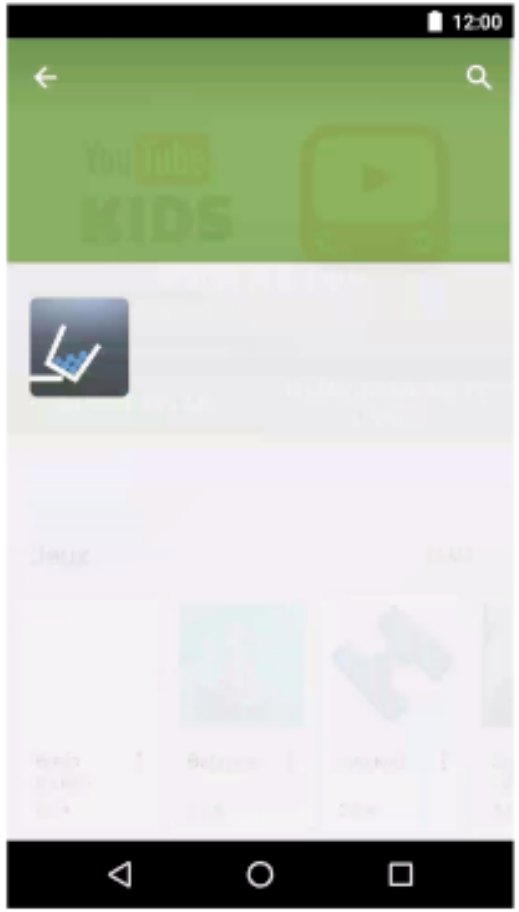
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
}
```



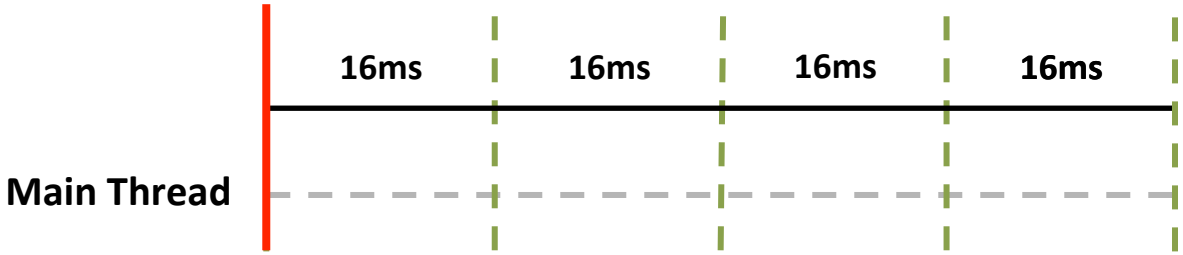


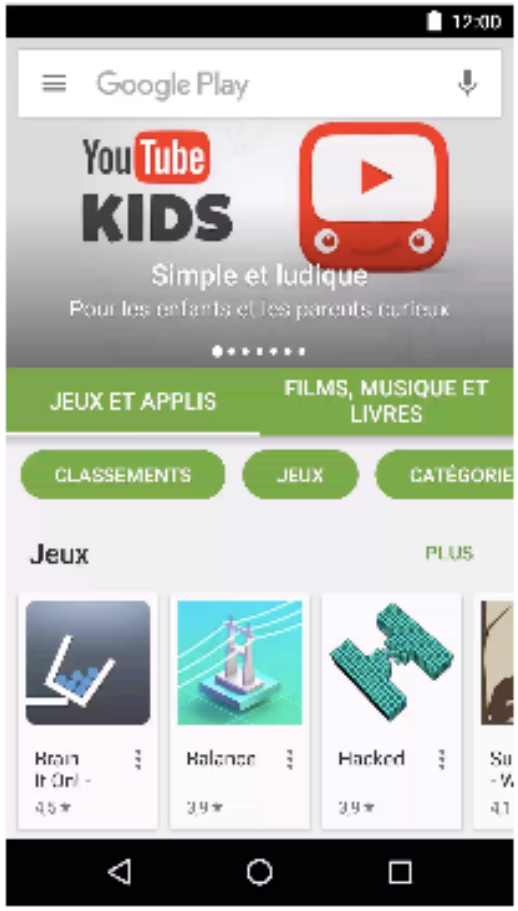
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
}
```



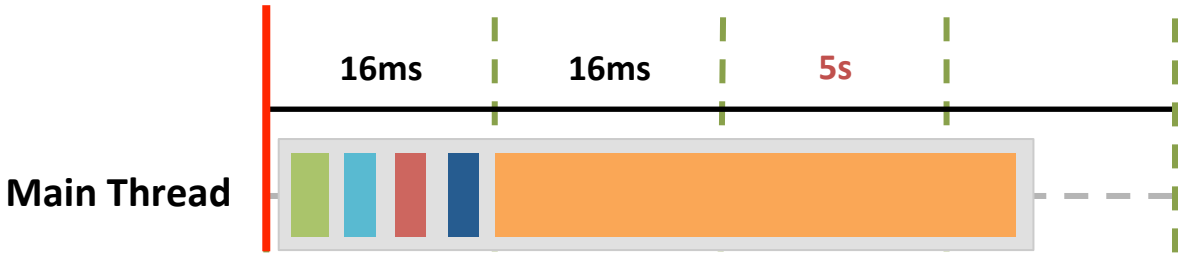


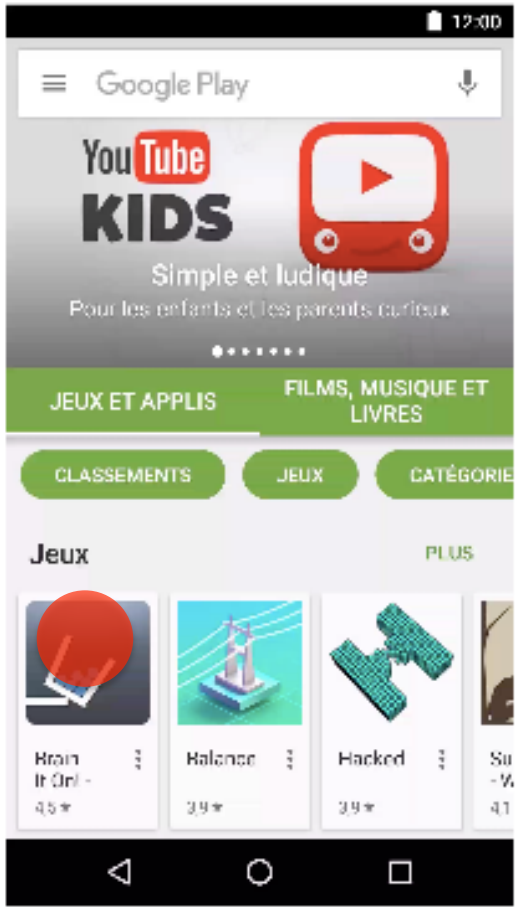
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
    longWorkToDo();
}
```



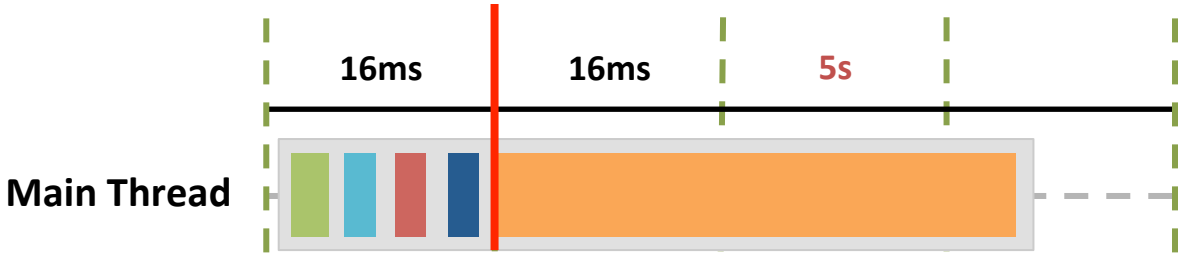


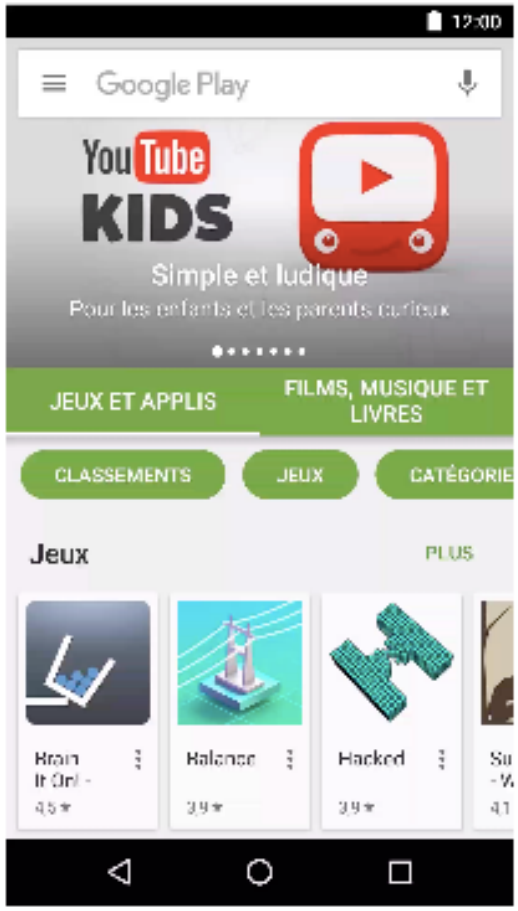
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
    longWorkToDo();
}
```



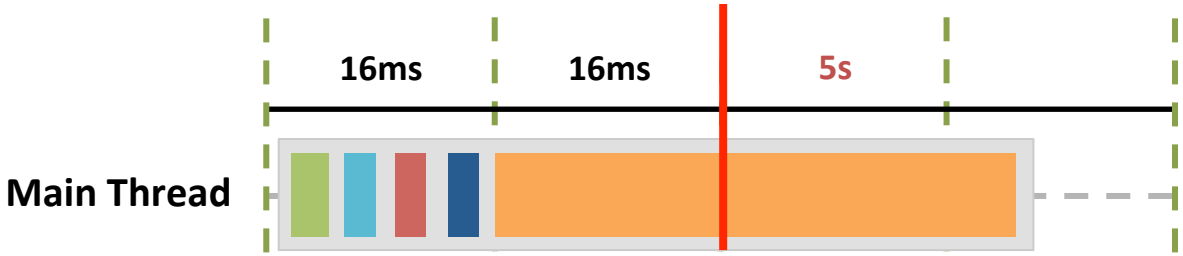


```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
    longWorkToDo();
}
```

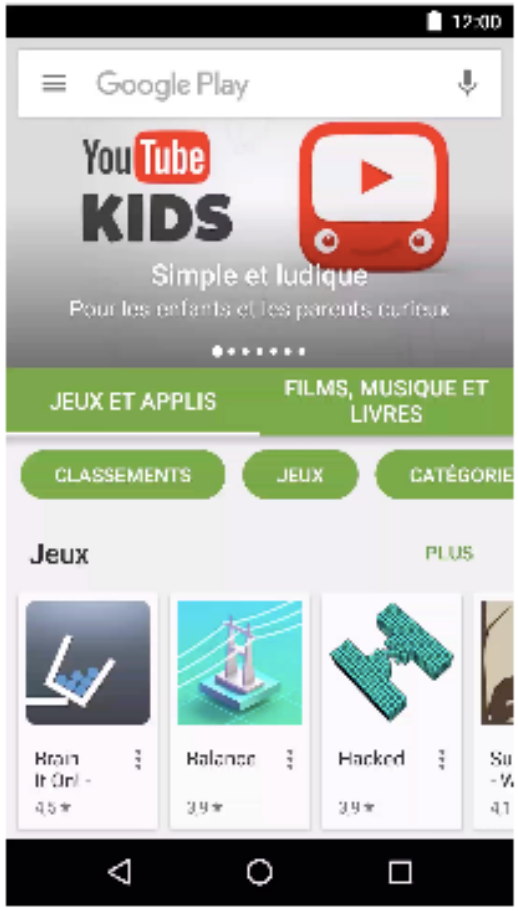




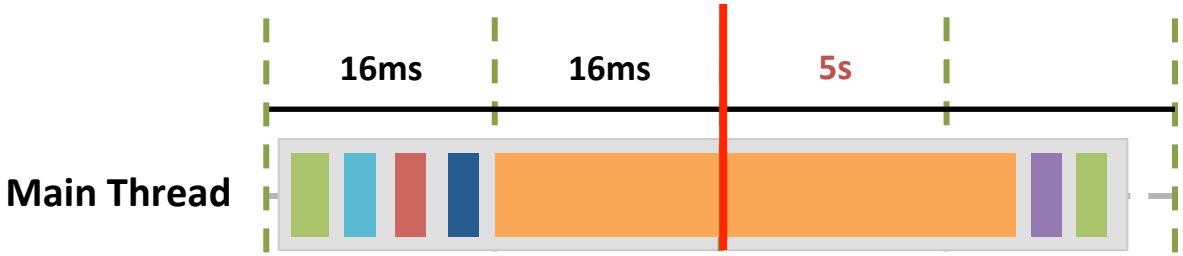
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
    longWorkToDo();
}
```

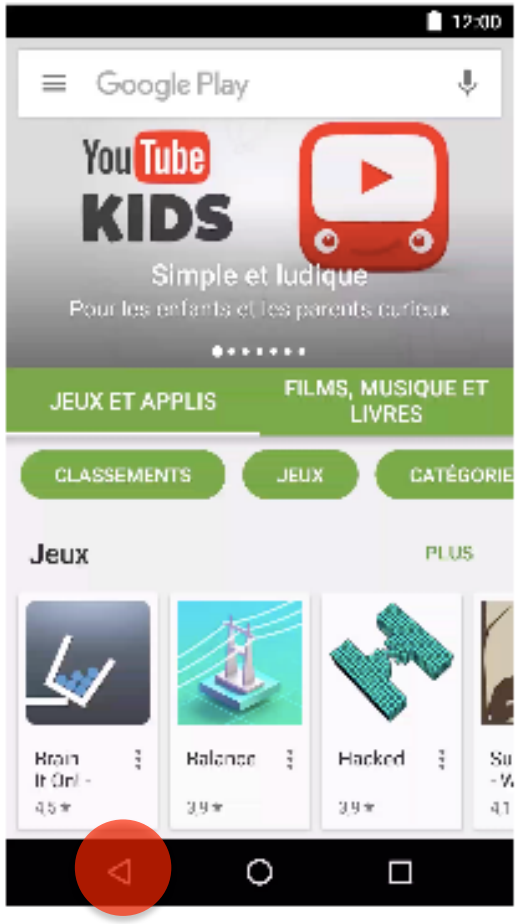


Choreographer skipped 2 frames !

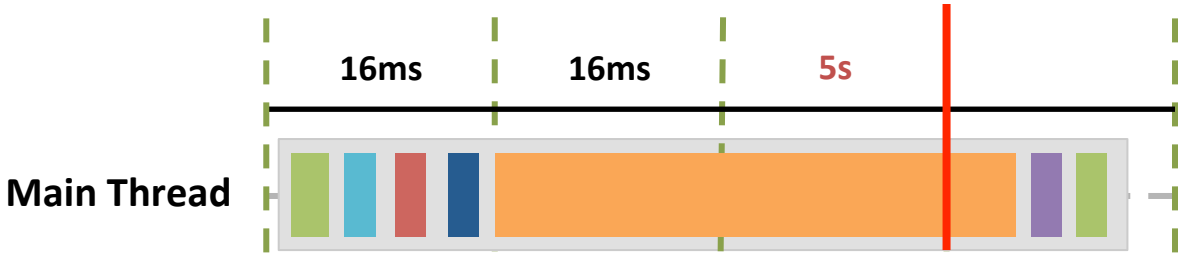


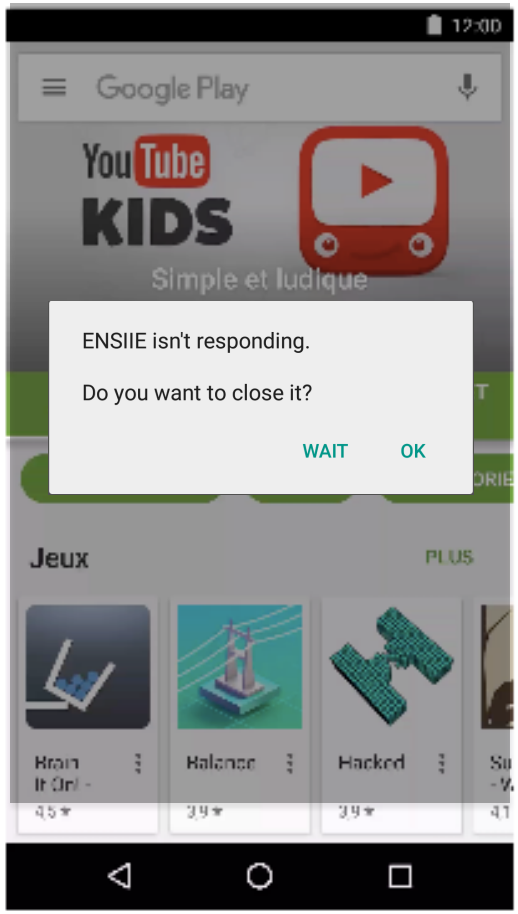
```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
    longWorkToDo();
}
```





```java
@Override
public void onClick(View v) {
    animateImage();
    animateText();
    startActivity(new Intent(this, Activity.class));
    longWorkToDo();
}
```





Création d'un background Thread

- Exécuter tous les traitements longs dans un background thread
- Exécuter tous les traitements sur la vue dans le Main Thread

Création d'un background Thread

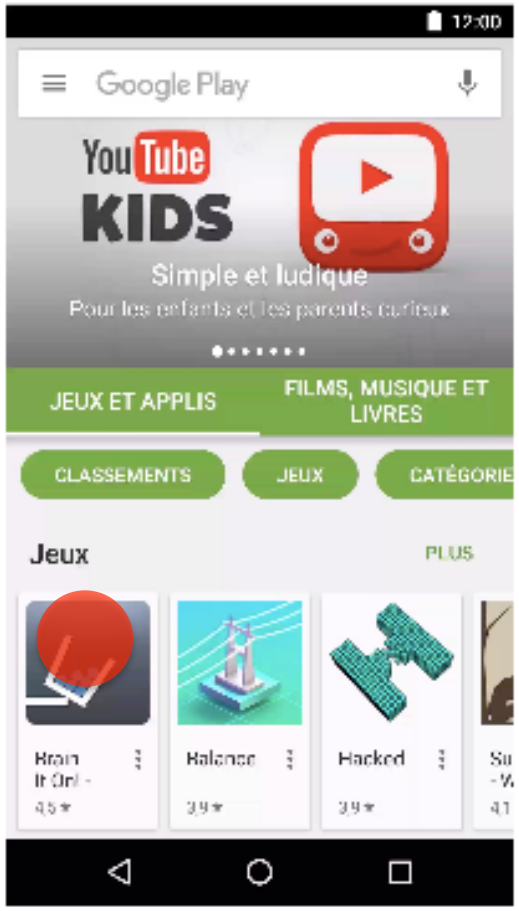
- Exécuter tous les traitements longs dans un background thread
- Exécuter tous les traitements sur la vue dans le Main Thread
- **Important** : la vue ne peut être uniquement manipulée à partir du Main Thread (UI Thread).

Création d'un background Thread

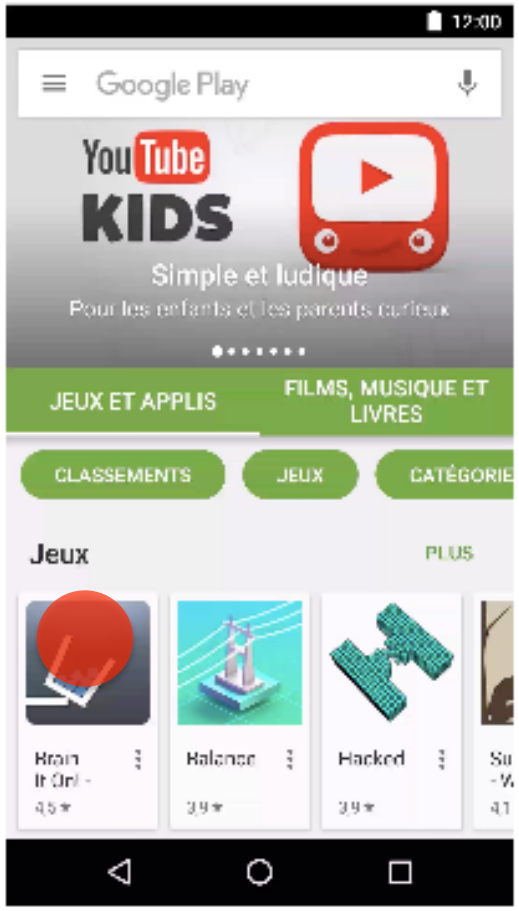
- Exécuter tous les traitements longs dans un background thread
- Exécuter tous les traitements sur la vue dans le Main Thread
- **Important** : la vue ne peut être uniquement manipulée à partir du Main Thread (UI Thread).

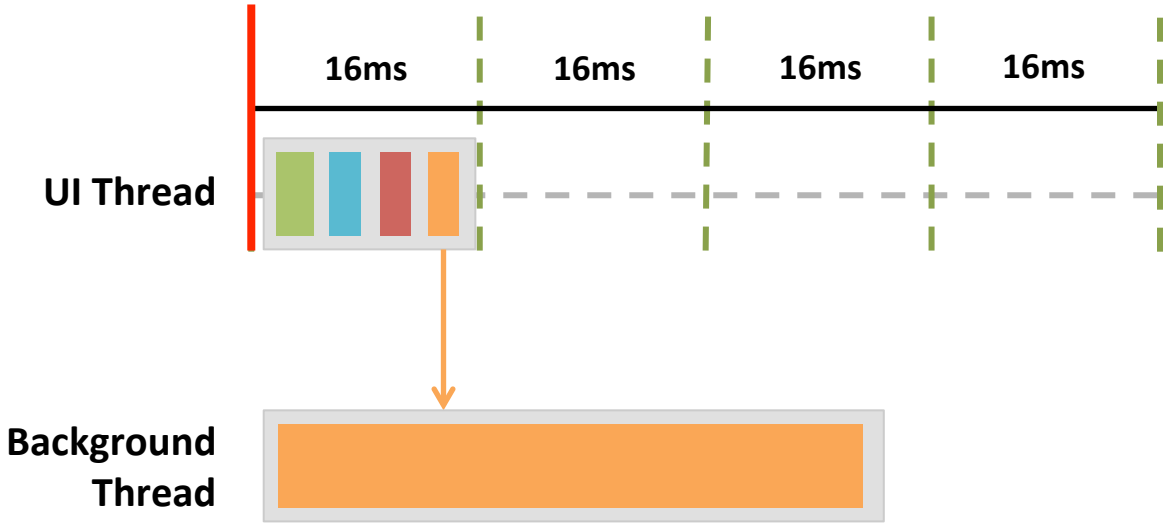
CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.

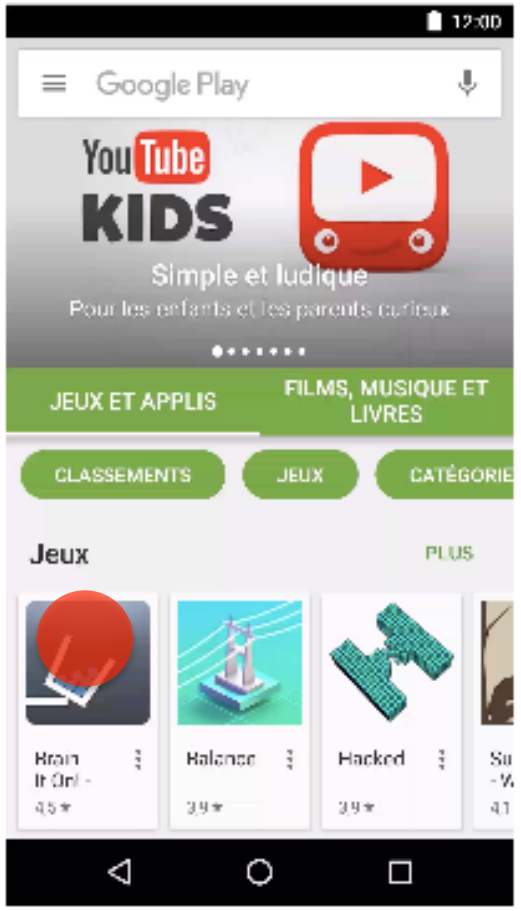

16ms
16ms
5s
UI Thread

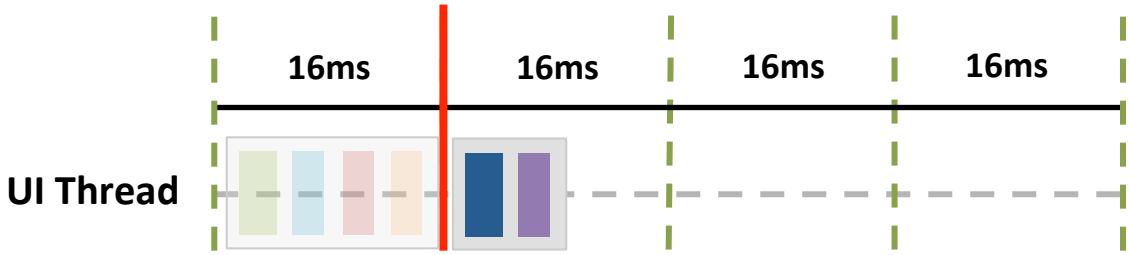

Google Play
YouTube
KIDS
Simple et ludique
JEUX ET APPLIS
FILMS, MUSIQUE ET LIVRES
CLASSEMENTS
JEUX
Jeux
PLUS
Balance
Hacked

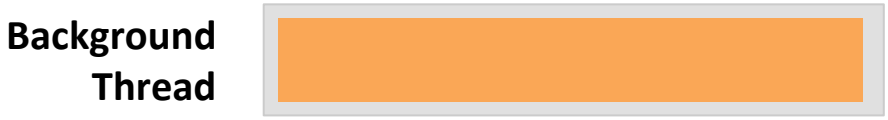
Background Thread

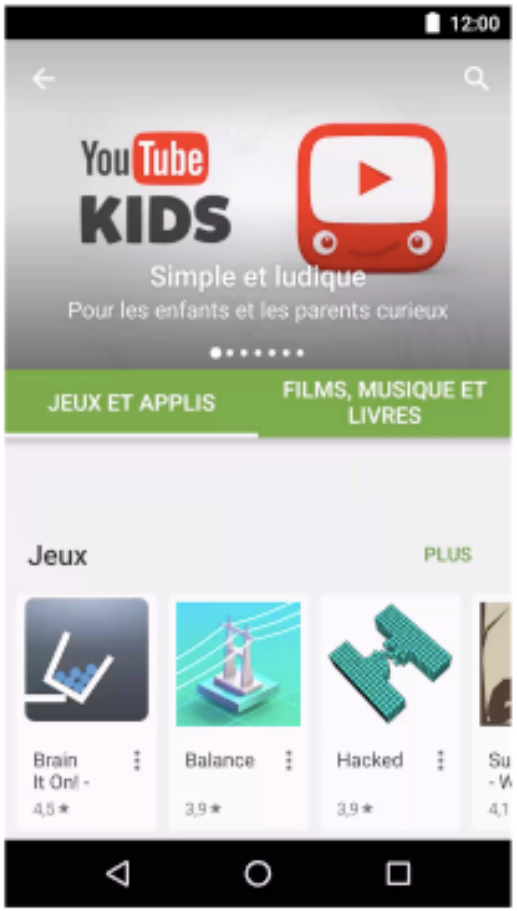


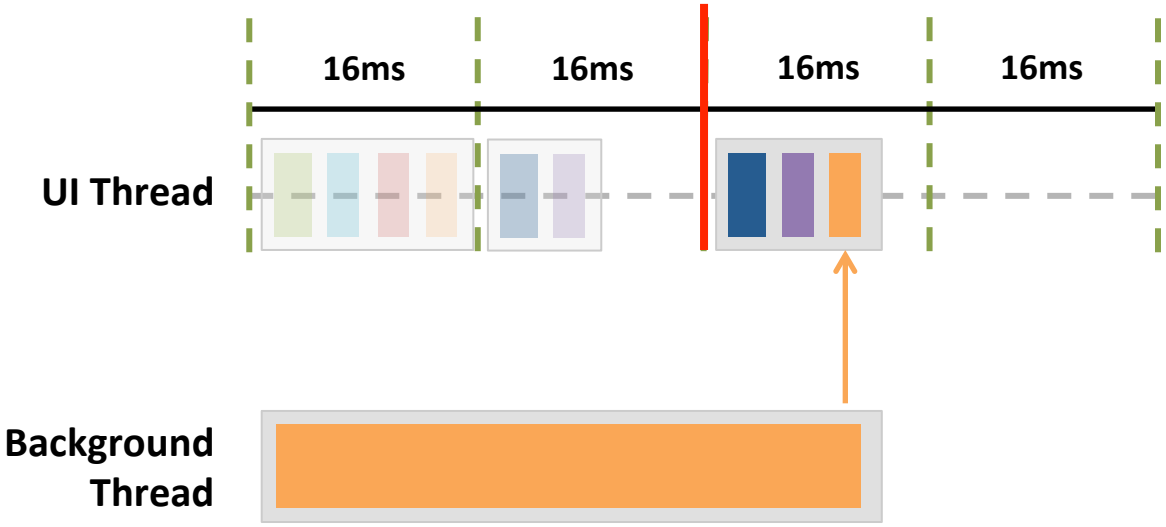

16ms
16ms
16ms
16ms
UI Thread
Background Thread

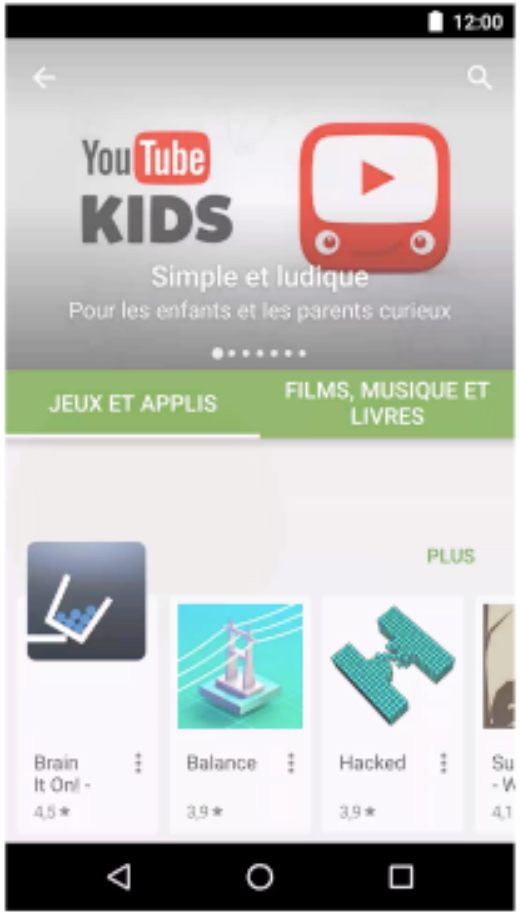

Google Play
YouTube KIDS
Simple et ludique
JEUX ET APPLIS
FILMS, MUSIQUE ET LIVRES
CLASSEMENTS
JEUX
Jeux
PLUS
Balance
Hacked

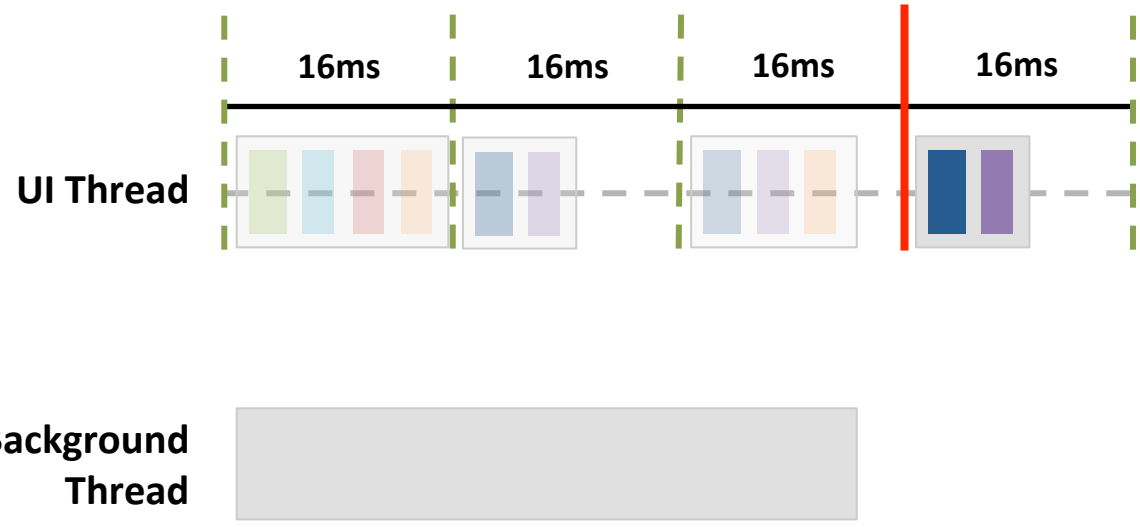

16ms
16ms
16ms
16ms
UI Thread

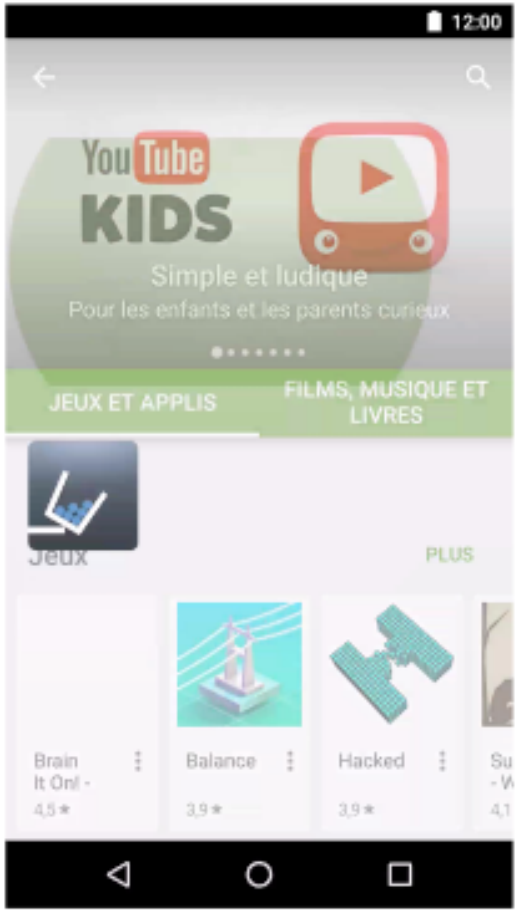

Background
Thread

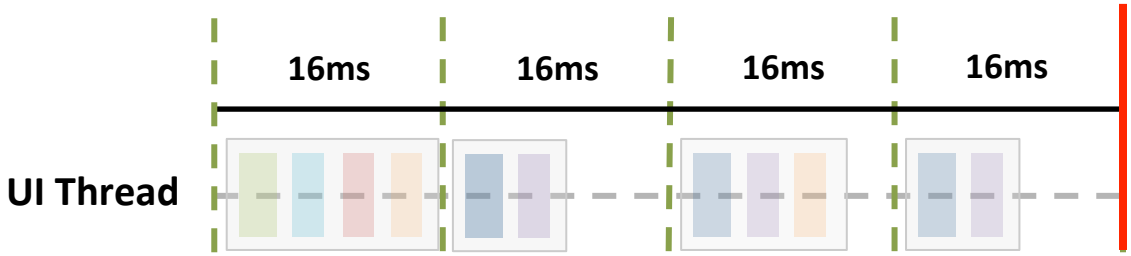

12:00
YouTube
KIDS
Simple et ludique
Pour les enfants et les parents curieux
JEUX ET APPLIS
FILMS, MUSIQUE ET LIVRES
Jeux
PLUS
Brain
Balance
Hacked

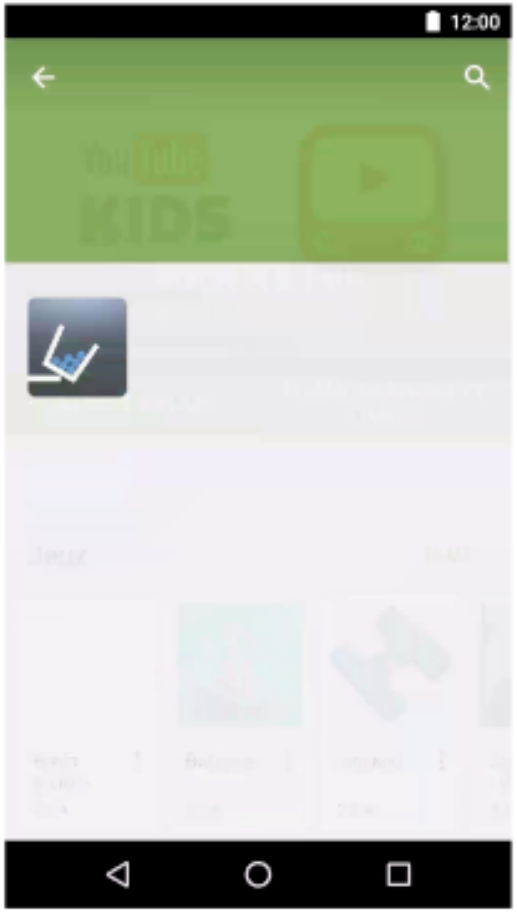

16ms
16ms
16ms
16ms
UI Thread
Background Thread


12:00
YouTube KIDS
Simple et ludique
Pour les enfants et les parents curieux
JEUX ET APPLIS
FILMS, MUSIQUE ET LIVRES
PLUS
Brain It On! -
Balance
Hacked


16ms
16ms
16ms
16ms
UI Thread
Background Thread


12:00
YouTube
KIDS
Simple et ludique
Pour les enfants et les parents curieux
JEUX ET APPLIS
FILMS, MUSIQUE ET LIVRES
Jeux
PLUS
Brain
Balance
Hacked


16ms
16ms
16ms
16ms
UI Thread


12:00

Classes helper pour Background Thread

- Handler/Thread
- AsyncTask
- IntentService
- HandlerThread
- ThreadPoolExecutor
- …

Handler/Thread

- Gestion manuelle du background Thread et de la communication avec l'UI Thread

Thread

- Permet d’exécuter du code
- Exécution du code séquentielle

Thread

Thread

- Permet d'exécuter du code
- Exécution du code séquentielle

```
class MyThread extends Thread {
    @Override
    public void run() {
        // Code to execute
    }
}

Thread thread = new MyThread();
thread.start();
```

Thread

Création et exécution d'un Thread

Thread

- Permet d'exécuter du code
- Exécution du code séquentielle

```
class MyRunnable implements Runnable {
    @Override
    public void run() {
        // Code to execute
    }
}

Thread thread = new Thread(new MyRunnable());
thread.start();
```

Thread

Création d'un runnable et exécution dans un Thread

Thread

- Permet d’exécuter du code
- Exécution du code séquentielle
- Exécute la méthode `run()` et se termine

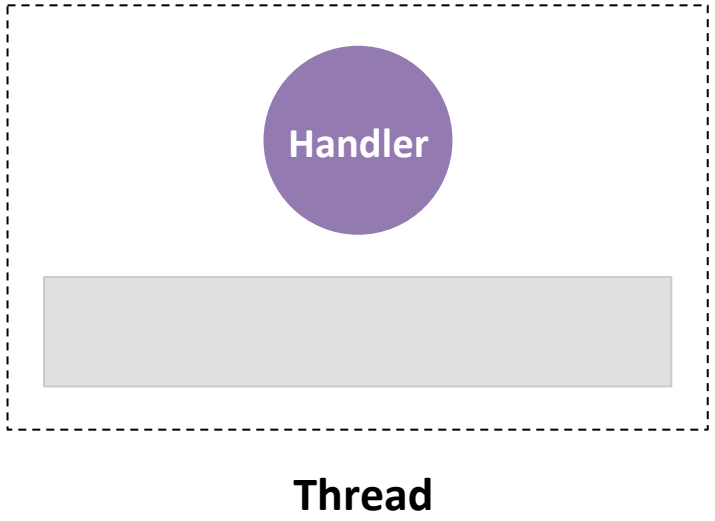
Thread

Handler

- Lié au Thread qui le crée
- Permet de communiquer avec ce Thread et lui envoyer des blocs de code à exécuter

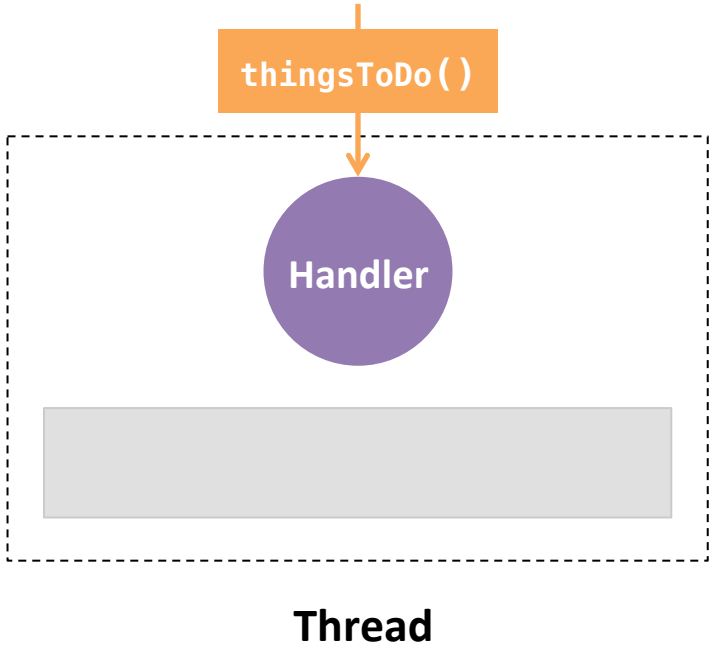
Handler

- Lié au Thread qui le crée
- Permet de communiquer avec ce Thread et lui envoyer des blocs de code à exécuter



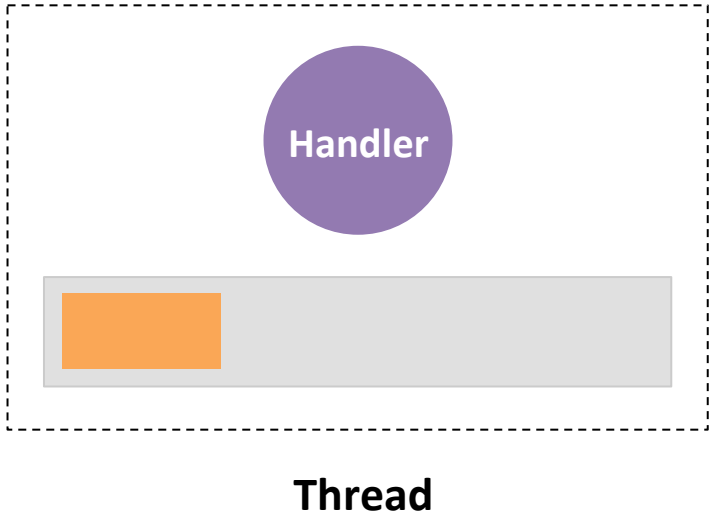
Handler

- Lié au Thread qui le crée
- Permet de communiquer avec ce Thread et lui envoyer des blocs de code à exécuter



Handler

- Lié au Thread qui le crée
- Permet de communiquer avec ce Thread et lui envoyer des blocs de code à exécuter

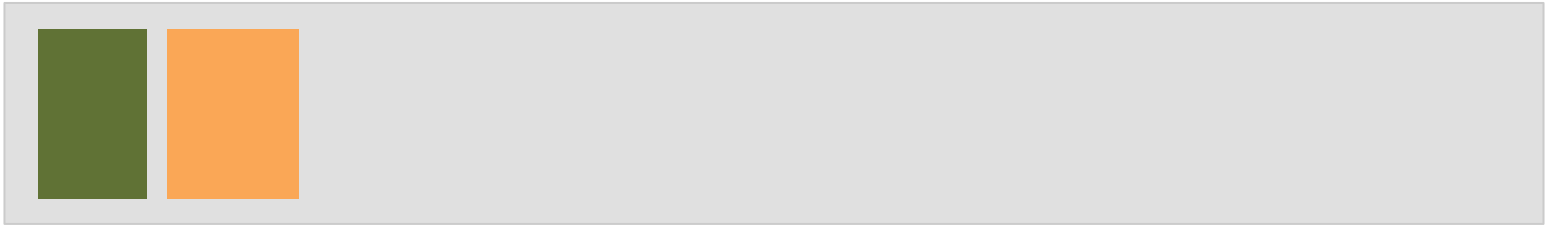


Handler

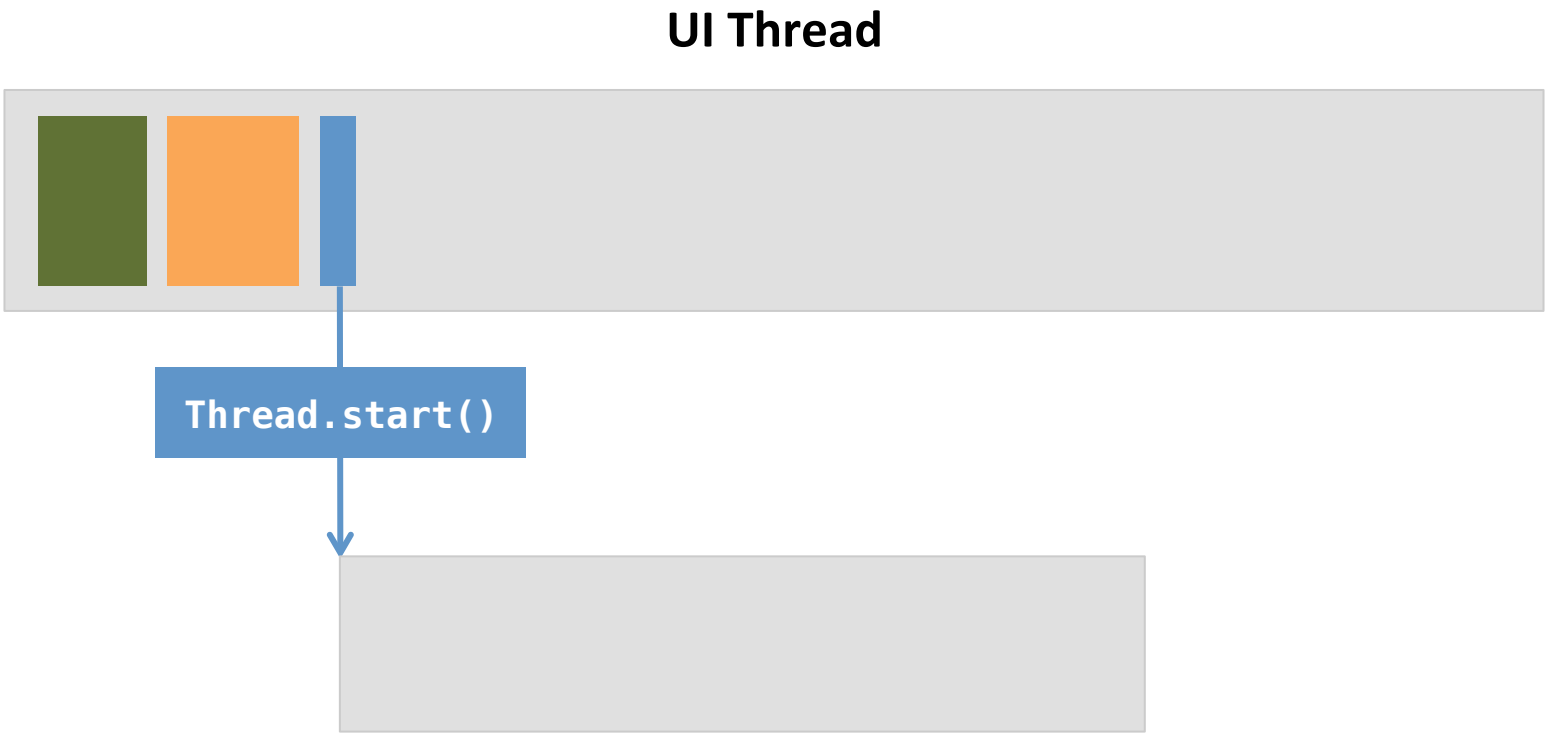
- Lié au Thread qui le crée
- Permet de communiquer avec ce Thread et lui envoyer des blocs de code à exécuter

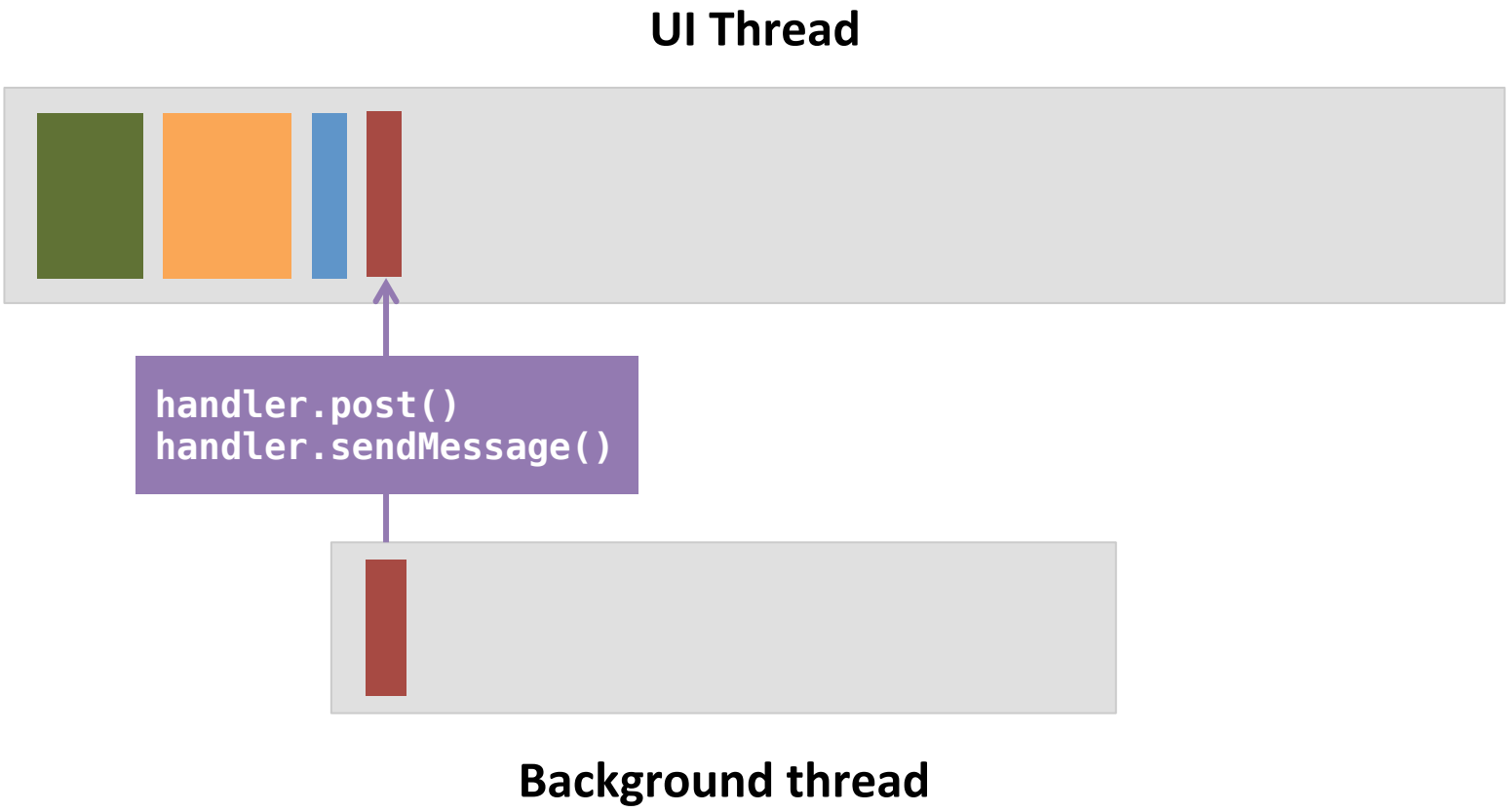
```
// Bind handler to the UI Thread
private Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {} // process incoming messages
};

// Background Thread
class MyThread extends Thread {
    @Override
    public void run() {
        // Execute a runnable on the UI Thread
        handler.post(new Runnable() {});
        // Send a message to the handler
        handler.sendMessage(new Message());
    }
}
```

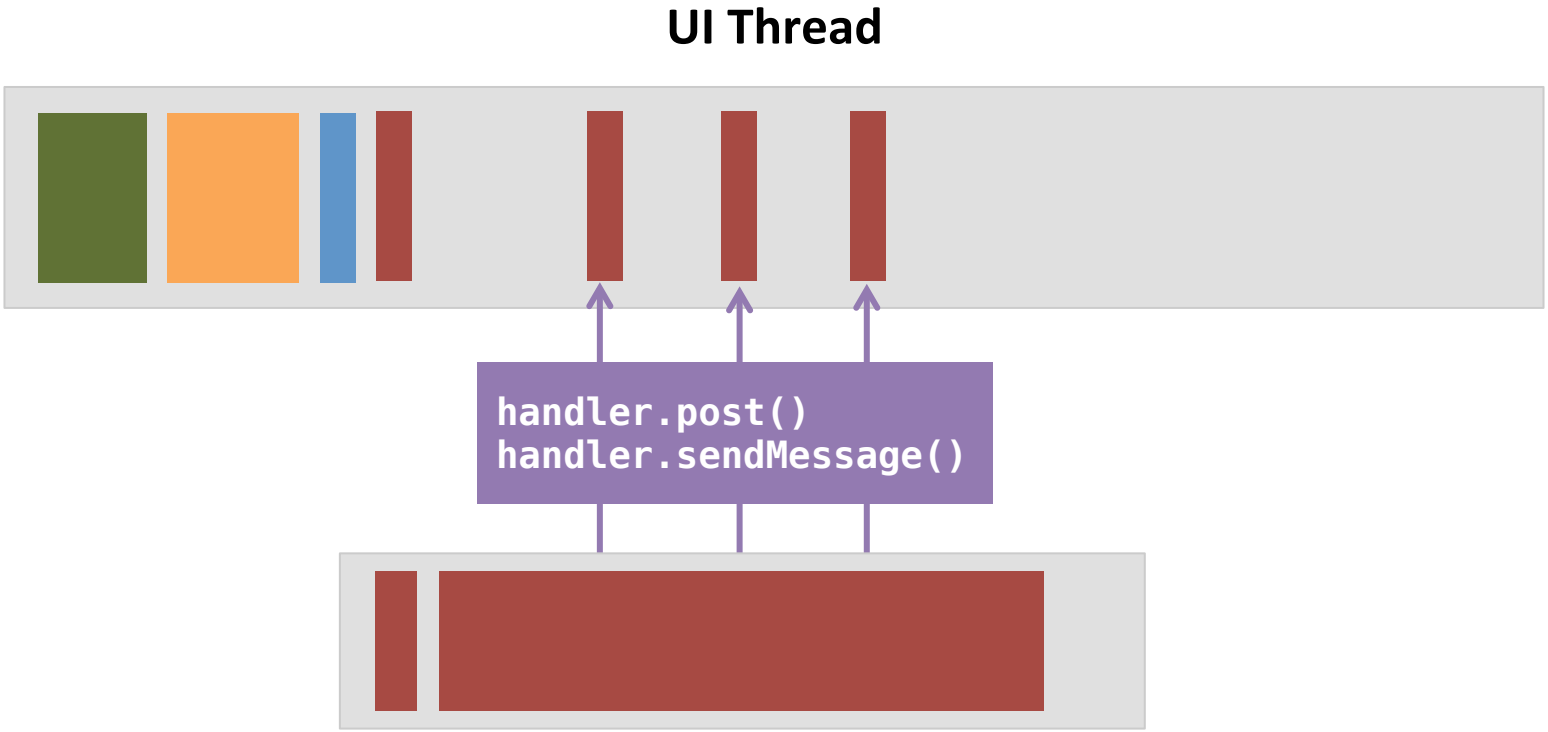
UI Thread

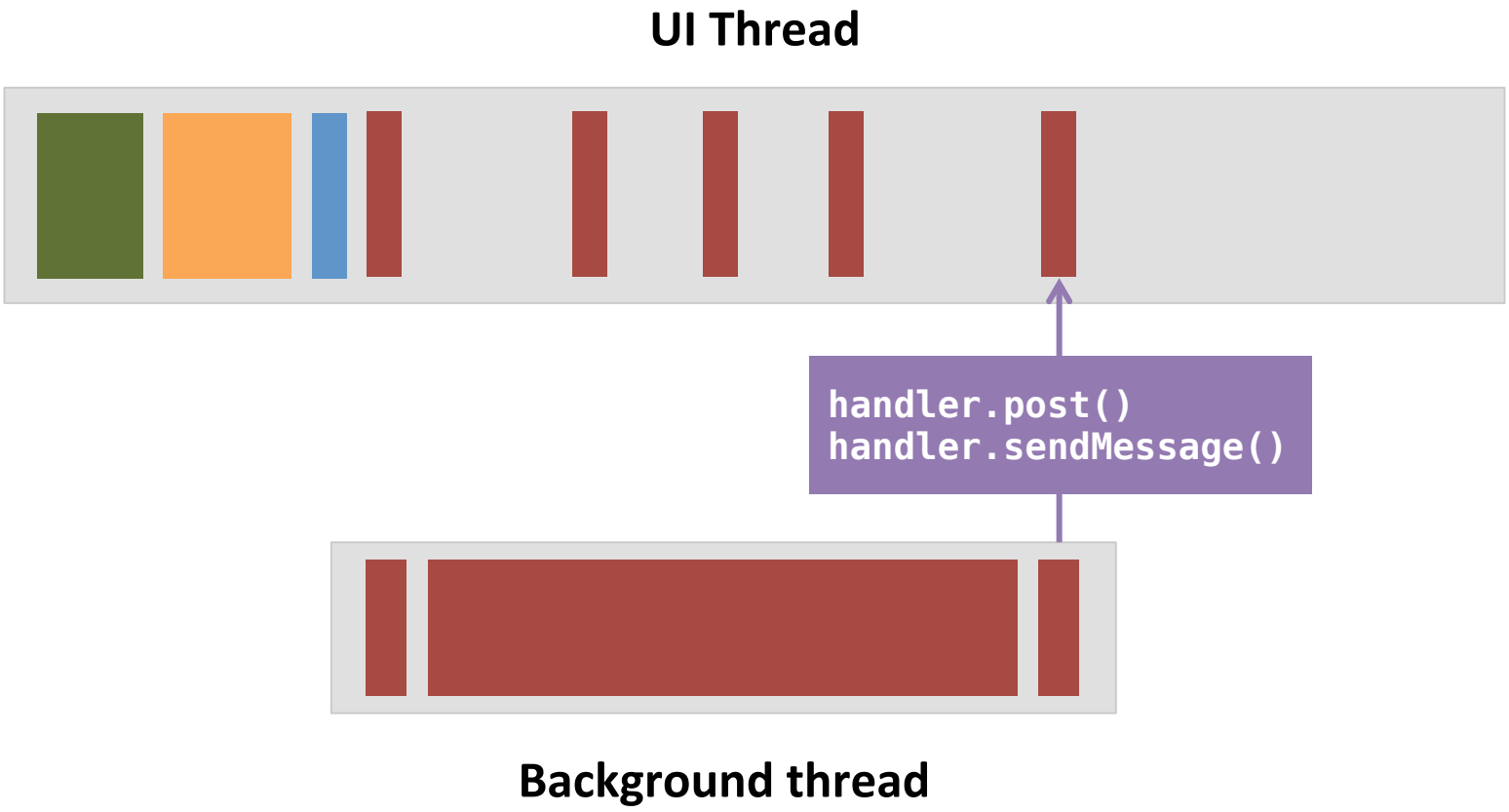


Background thread


UI Thread
Thread.start()
Background thread


UI Thread
handler.post()
handler.sendMessage()
Background thread


UI Thread
handler.post()
handler.sendMessage()
Background thread


UI Thread
handler.post()
handler.sendMessage()
Background thread

```
Handler handler = new Handler();

public void onClick(View v) {
    Thread thread = new Thread() {
        @Override
        public void run() {
            handler.post(new Runnable() {
                @Override
                public void run() {
                    button.setVisibility(View.GONE);
                    progressBar.setVisibility(View.VISIBLE);
                }
            });
            longWorkToDo();
            handler.post(new Runnable() {
                @Override
                public void run() {
                    button.setVisibility(View.VISIBLE);
                    progressBar.setVisibility(View.GONE);
                }
            });
        }
    };
    thread.start();
}
```

```
public void onClick(View v) {
    Thread thread = new Thread() {
        @Override
        public void run() {
            // Use Activity default Handler
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    button.setVisibility(View.GONE);
                    progressBar.setVisibility(View.VISIBLE);
                }
            });
            longWorkToDo();
            // Use Activity default Handler
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    button.setVisibility(View.VISIBLE);
                    progressBar.setVisibility(View.GONE);
                }
            });
        }
    };
    thread.start();
}
```

```
private static final int START = 0;
private static final int DONE = 1;


private Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        if (msg.what == START) {
            button.setVisibility(View.GONE);
            progressBar.setVisibility(View.VISIBLE);
        } else {
            button.setVisibility(View.VISIBLE);
            progressBar.setVisibility(View.GONE);
        }
    }
};

public void onClick(View v) {
    final Thread thread = new Thread() {
        @Override
        public void run() {
            handler.sendEmptyMessage(START);
            Utils.workTodo();
            handler.sendEmptyMessage(DONE);
        }
    };
    thread.start();
}
```

AsyncTask

- Classe helper utilisant en interne Handler / Thread
- Utilisé pour l'exécution de tâches courtes ne nécessitant pas de survivre à la destruction d'une Activity

AsyncTask

```
class MyAsyncTask extends AsyncTask<Param, Param, Param> {

    // Called on the UI Thread before executing the task
    @Override
    protected void onPreExecute() {}
}
```

AsyncTask

```
class MyAsyncTask extends AsyncTask<Param, Param, Param> {

    // Called on the UI Thread before executing the task
    @Override
    protected void onPreExecute() {}

    // Task to execute. Happen in a background Thread
    @Override
    protected Param doInBackground(Param... params) {
        publishProgress(progress)
        return res;
    }

    // Called on the UI Thread to publish progress on task execution
    @Override
    protected void onProgressUpdate(Param... values) {}
}
```

AsyncTask

```
class MyAsyncTask extends AsyncTask<Param, Param, Param> {

    // Called on the UI Thread before executing the task
    @Override
    protected void onPreExecute() {}

    // Task to execute. Happen in a background Thread
    @Override
    protected Param doInBackground(Param... params) {
        publishProgress(progress)
        return res;
    }

    // Called on the UI Thread to publish progress on task execution
    @Override
    protected void onProgressUpdate(Param... values) {}
}
```

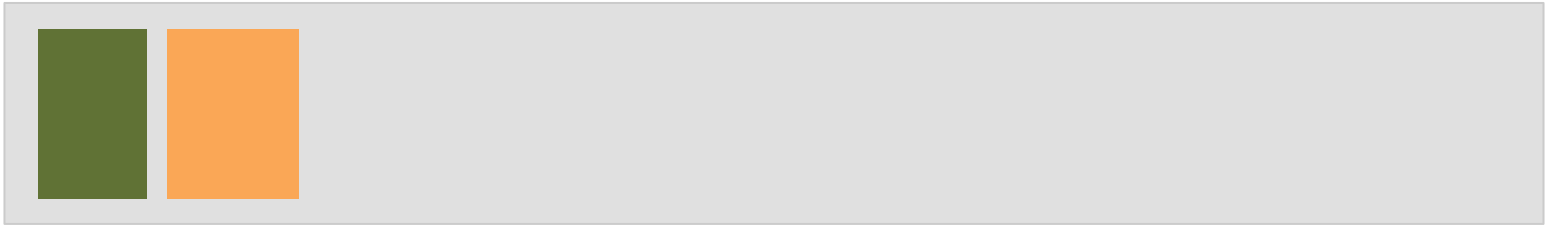
AsyncTask

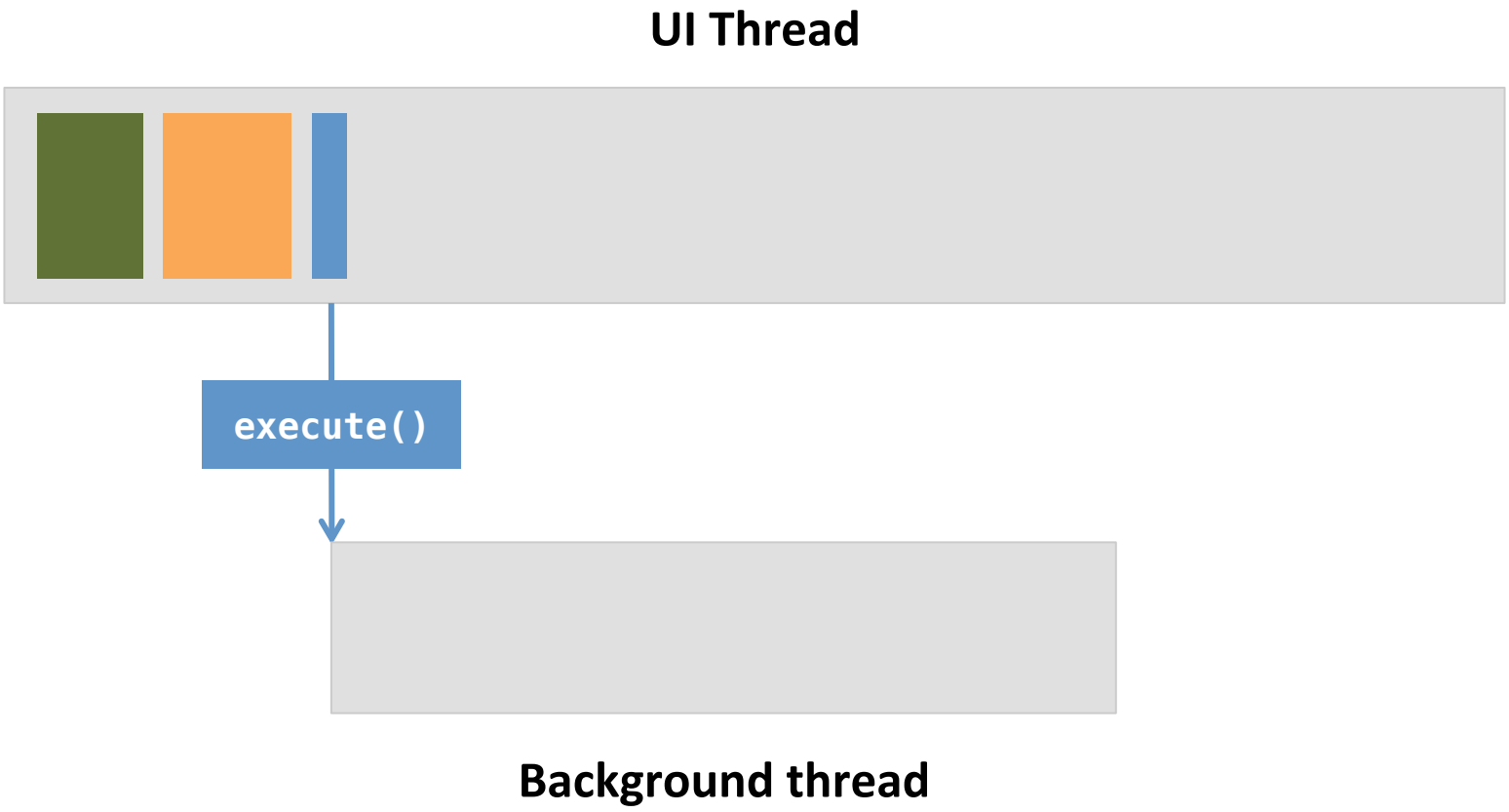
```
class MyAsyncTask extends AsyncTask<Param, Param, Param> {

    // Called on the UI Thread before executing the task
    @Override
    protected void onPreExecute() {}

    // Task to execute. Happen in a background Thread
    @Override
    protected Param doInBackground(Param... params) {
        publishProgress(progress)
        return res;
    }

    // Called on the UI Thread to publish progress on task execution
    @Override
    protected void onProgressUpdate(Param... values) {}

    // Called on the UI Thread when the task is done
    @Override
    protected void onPostExecute(Param res) {}
}
```

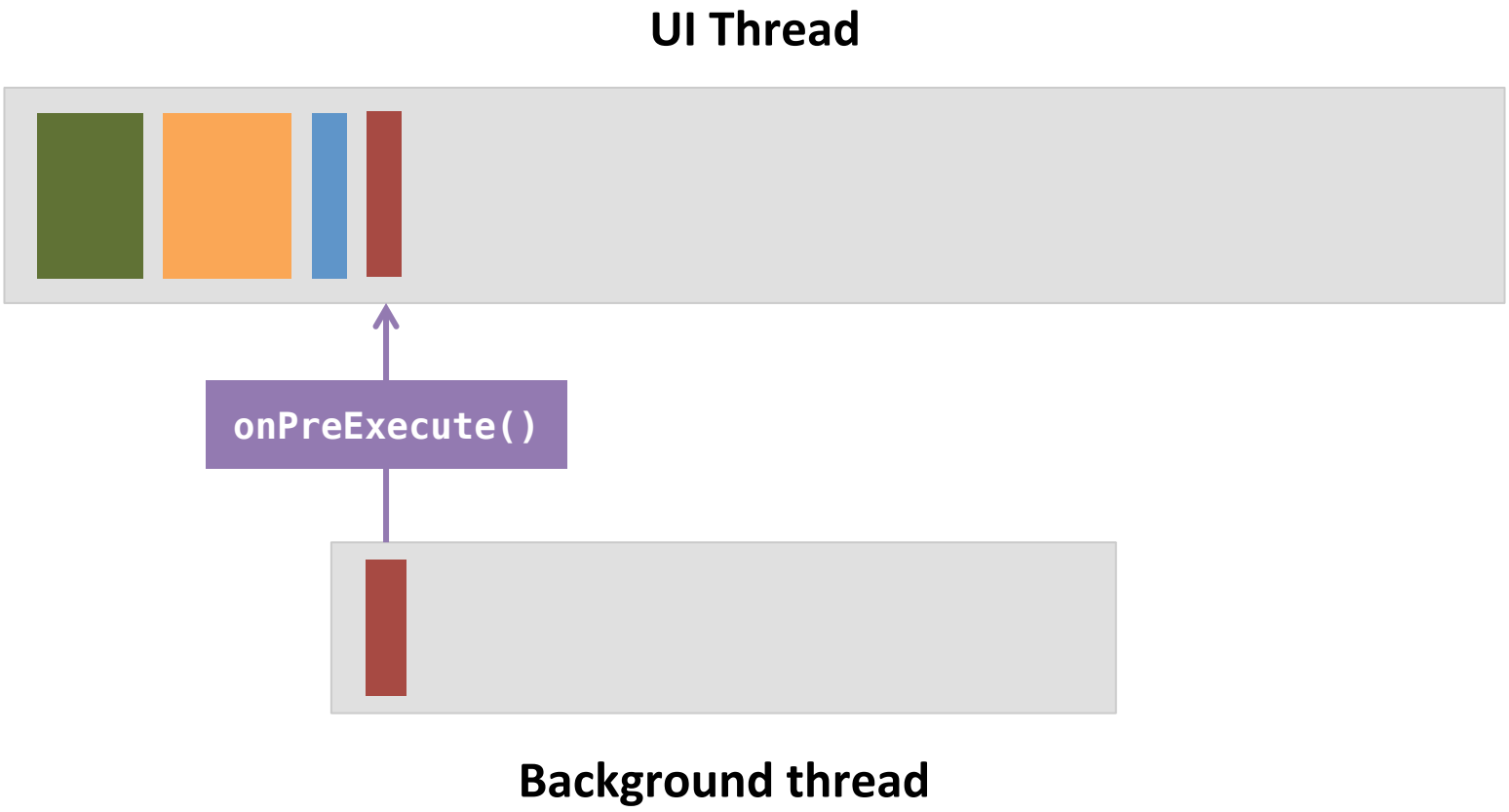
AsyncTask

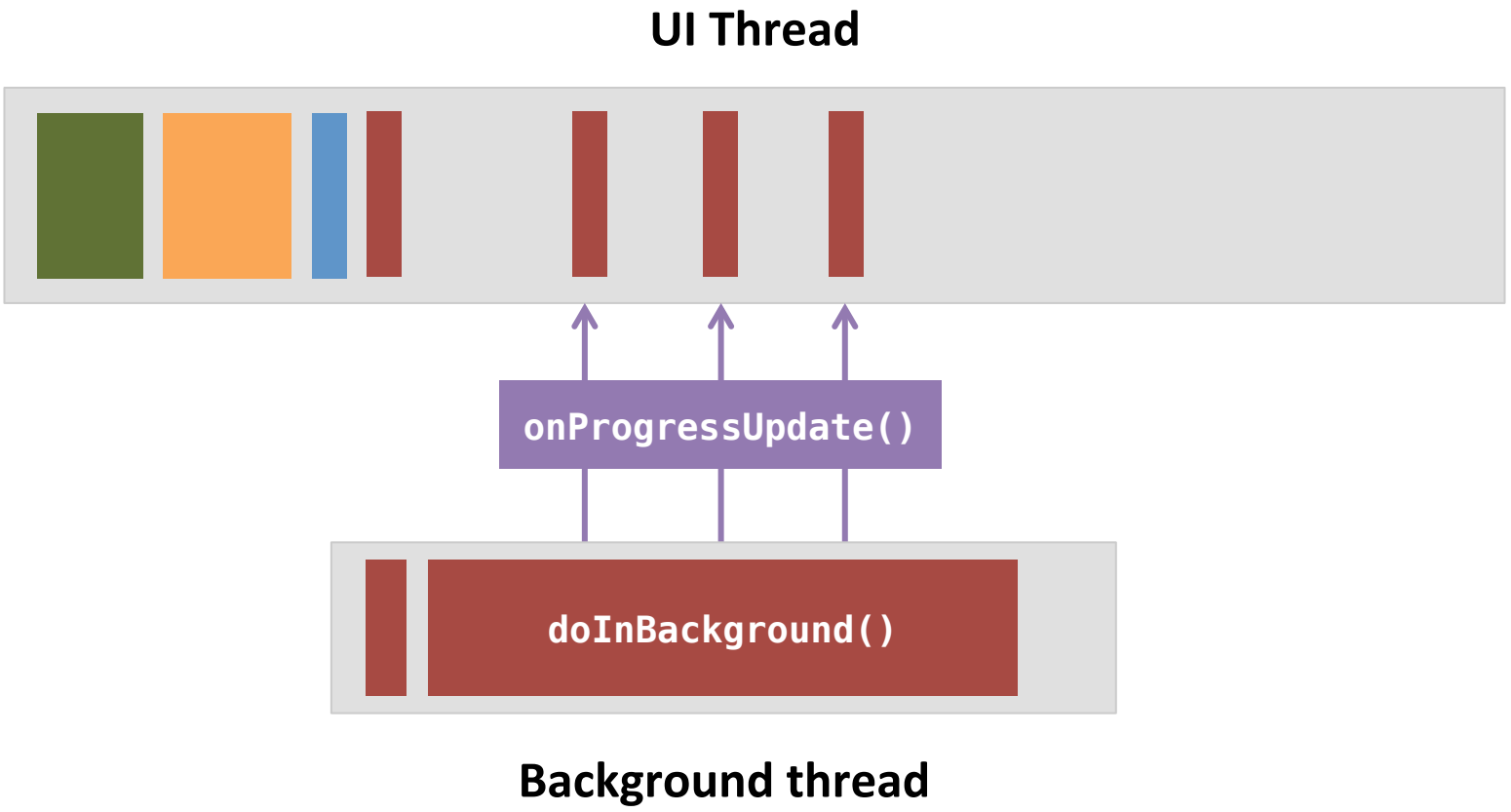
```
class MyAsyncTask extends AsyncTask<Param, Param, Param> {

    // Called on the UI Thread before executing the task
    @Override
    protected void onPreExecute() {}

    // Task to execute. Happen in a background Thread
    @Override
    protected Param doInBackground(Param... params) {
        publishProgress(progress)
        return res;
    }

    // Called on the UI Thread to publish progress on task execution
    @Override
    protected void onProgressUpdate(Param... values) {}

    // Called on the UI Thread when the task is done
    @Override
    protected void onPostExecute(Param res) {}
}
```

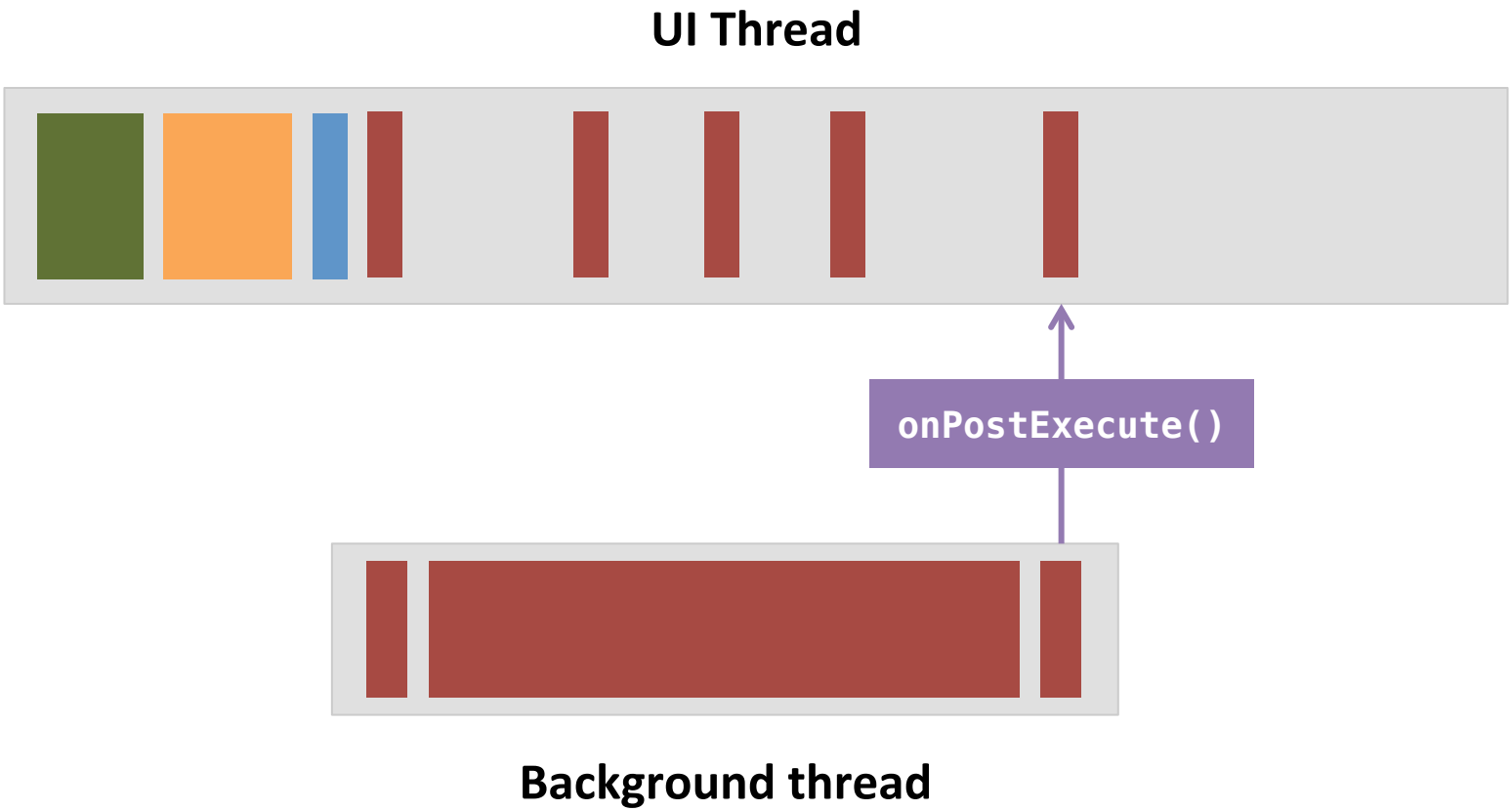
UI Thread




UI Thread
execute()
Background thread


UI Thread
onPreExecute()
Background thread


UI Thread
onProgressUpdate()
doInBackground()
Background thread


UI Thread
onPostExecute()
Background thread

```
class MyAsyncTask extends AsyncTask<Void, Void, Void> {

    @Override
    protected void onPreExecute() {
        button.setVisibility(View.GONE);
        progressBar.setVisibility(View.VISIBLE);
    }

    @Override
    protected Void doInBackground(Void... params) {
        Utils.workTodo();
        return null;
    }

    @Override
    protected void onPostExecute(Void aVoid) {
        button.setVisibility(View.VISIBLE);
        progressBar.setVisibility(View.GONE);
    }
}

MyAsyncTask asyncTask = new MyAsyncTask();
asyncTask.execute();;
```

Effectuer un appel réseau

- Permission INTERNET nécessaire

```
<uses-permission android:name="android.permission.INTERNET" />
```

Effectuer un appel réseau

- Permission INTERNET nécessaire
- Traitement long à effectuer dans un background Thread
  - Exécution d'un appel unique par Activity → AsyncTask
  - Exécution de plusieurs appels à la suite → HandlerThread
  - Exécution d'appels en simultanés → ThreadPoolExecutor

Effectuer un appel réseau

- Permission INTERNET nécessaire
- Traitement long à effectuer dans un background Thread
- Utilisation d'une librairie externe OKHttp

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.squareup.okhttp3:okhttp:3.1.2'
}
```

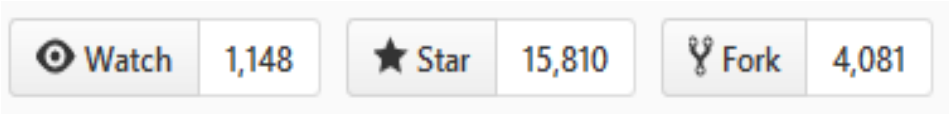
Définition de la librairie OKHttp dans build.gradle

Fichier build.gradle

- Permet de définir les librairies dont dépend le projet

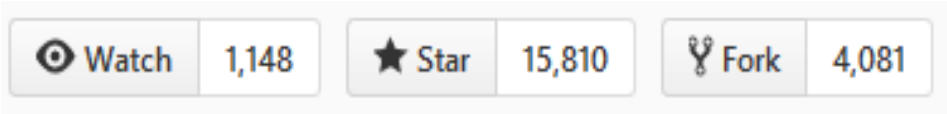
```
apply plugin: 'com.android.application'

android {
    ...
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.squareup.okhttp3:okhttp:3.4.2'
    compile 'io.reactivex:rxjava:1.1.6'
    compile 'com.android.support:recyclerview-v7:24.2.1'
    compile 'com.android.support:support-v4:24.2.1'
}
```

OKHttp



- Client HTTP basé sur l'interface HttpUrlConnection utilisé par Android
- Facilite et accélère les appels réseaux
  - Permet d'effectuer des appels GET / POST, l'envoi de ficher etc…
  - Mécanisme de réessaie quand une requête échoue
  - Possibilité d'annuler une requête
  - Gestion de cache pour les réponse des requêtes
  - Possibilité de définir un timeout
- Librairie robuste largement testée et utilisée

OKHttp



Composant de la librairie :

- OKHttpClient
- Requests
- Calls
- Responses

OKHttpClient

- Doit avoir une seule instance dans toute l'application
- Permet d'effectuer les appels réseaux

```
private final static OkHttpClient client = new OkHttpClient();
```

Déclaration du client OkHttp

Requests

- Défini la requête à envoyer
- Contient :
  - Une URL
  - La méthode d'appel (GET, POST)
  - Une liste de headers

```
Request request = new Request.Builder()
        .url("www.google.com")
        .addHeader("name", "value")
        .build();
```

Construction d'une requête GET vers Google avec un header

Requests

- Défini la requête à envoyer

```
RequestBody formBody = new FormBody.Builder()
        .add("name", "value")
        .build();

Request request = new Request.Builder()
        .url("www.google.com")
        .post(formBody)
        .build();
```

Construction d'une requête Post vers Google avec le paramètre name=value

Calls

- Représente l'envoi d'une requête
- Peut être exécuté en synchrone (Gestion soit même du background Thread) ou asynchrone
- Peut être annulé

```
try {
   Response response = okHttpClient.newCall(request).execute();
} catch (IOException e) {
   Log.e(TAG, e.getMessage(), e.getCause());
}
```

Exécution d'une requête en synchrone

Calls

```
client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        Log.e(TAG, e.getMessage(), e.getCause());
        // Background thread
    }

    @Override
    public void onResponse(Call call, Response response)
    throws IOException {
        // Background thread
        // Use an handler to update UI
    }
});
```

Exécution d'une requête en asynchrone

Responses

- Représente le retour d'un appel. Contient :
  - Code retour (200, 404 etc…)
  - Body

```
if (!response.isSuccessful()) {
      throw new IOException("Unexpected code " + response);
}

Log.d("Response received from server", response.body().string());
```

Parser des retours JSON

```
{
  "login": "okhttp",
  "id": 20570444,
  "images": [
    {
      "name": "image_1",
      "location": "location_1"
    },
    {
      "name": "image_2",
      "location": "location_2"
    }
  ]
}
```

Exemple de retour JSON

Création de la classe User et Image

```
public class User {
    String login;
    int id;
    ArrayList<Image> images;

    public User(String login, int id, ArrayList<Image> images) {
        this.login = login;
        this.id = id;
        this.images = images;
    }
}

public class Image {
    String name;
    String location;

    public Image(String name, String location) {
        this.name = name;
        this.location = location;
    }
}
```

Parser des retours JSON

```
{
  "login": "okhttp",
  "id": 20570444,
  "images": [
    {
      "name": "image_1",
      "location": "location_1"
    },
    {
      "name": "image_2",
      "location": "location_2"
    }
  ]
}
```

Object User

Parser des retours JSON

```
{
  "login": "okhttp",
  "id": 20570444,
  "images": [
    {
      "name": "image_1",
      "location": "location_1"
    },
    {
      "name": "image_2",
      "location": "location_2"
    }
  ]
}
```

Liste d'object Image

Parsing du JSON à la main

```
try {
    // Récupération de l'objet json User
    JSONObject userObj = new JSONObject(responseData);
    // Récupération du champ login
    String login = userObj.getString("login");
    // Récupération du champ id
    int id = userObj.getInt("id");
    // Récupération liste d'Image
    ArrayList<Image> images = getListOfImages(userObj);
    // Construction finale de l'utilisateur
    User user = new User(login, id, images);
} catch (JSONException e) {
    Log.e(TAG, e.getMessage());
}
```

Parsing du JSON

```
private ArrayList<Image> getListOfImages(JSONObject userObj)
throws JSONException {
    // Initialisation d'une liste d'Image
    ArrayList<Image> images = new ArrayList<>();
    // Récupération de l'array json liste d'Image
    JSONArray listImagesArr = userObj.getJSONArray("images");
    for (int i = 0; i < listImagesArr.length(); i++) {
        // Récupération de l'objet json Image
        JSONObject imageObj = listImagesArr.getJSONObject(i);
        // Récupération du champ name
        String name = imageObj.getString("name");
        // Récupération du champ location
        String location = imageObj.getString("location");
        // Ajout de l'Image à la liste
        images.add(new Image(name, location));
    }
    return images;
}
```