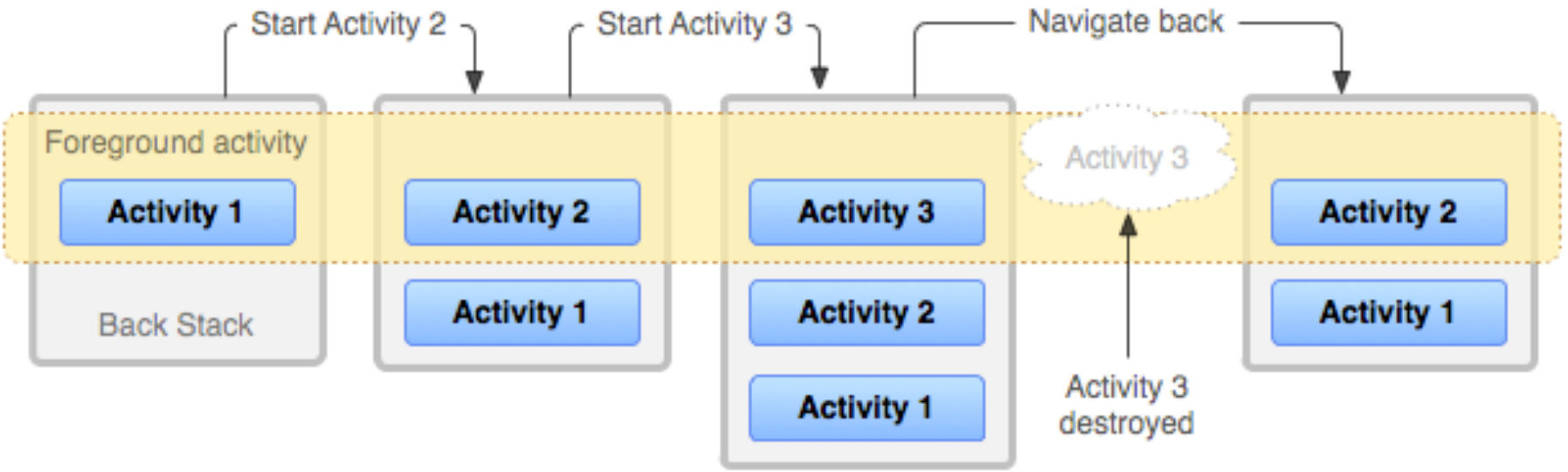
# SIL-1
# Cycle de vie des activités

# Application Android

- **Ensemble d'activités** liées entre elles
- Chaque fois qu'une activité est lancée, elle est placée dans une **pile** appelée back stack



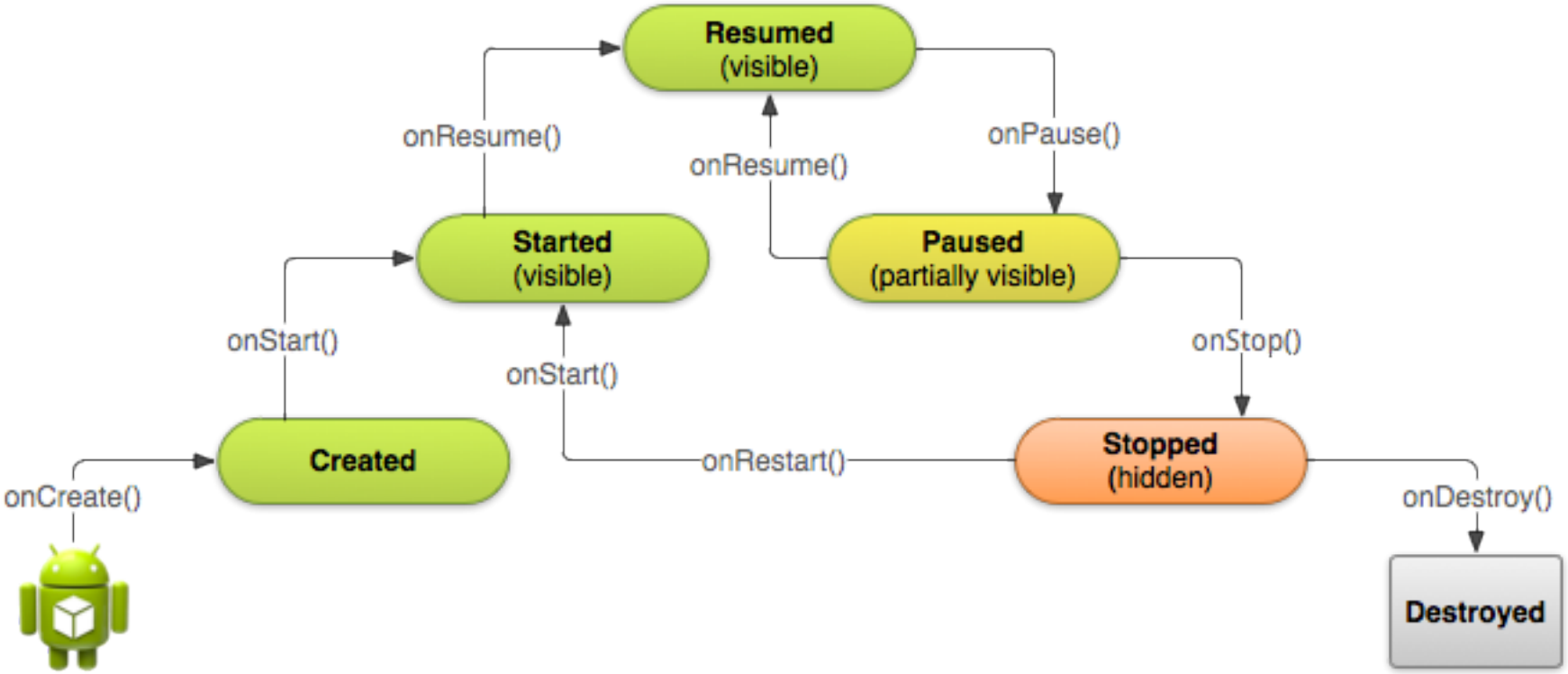
# Cycle de vie d'une activité (1/2)

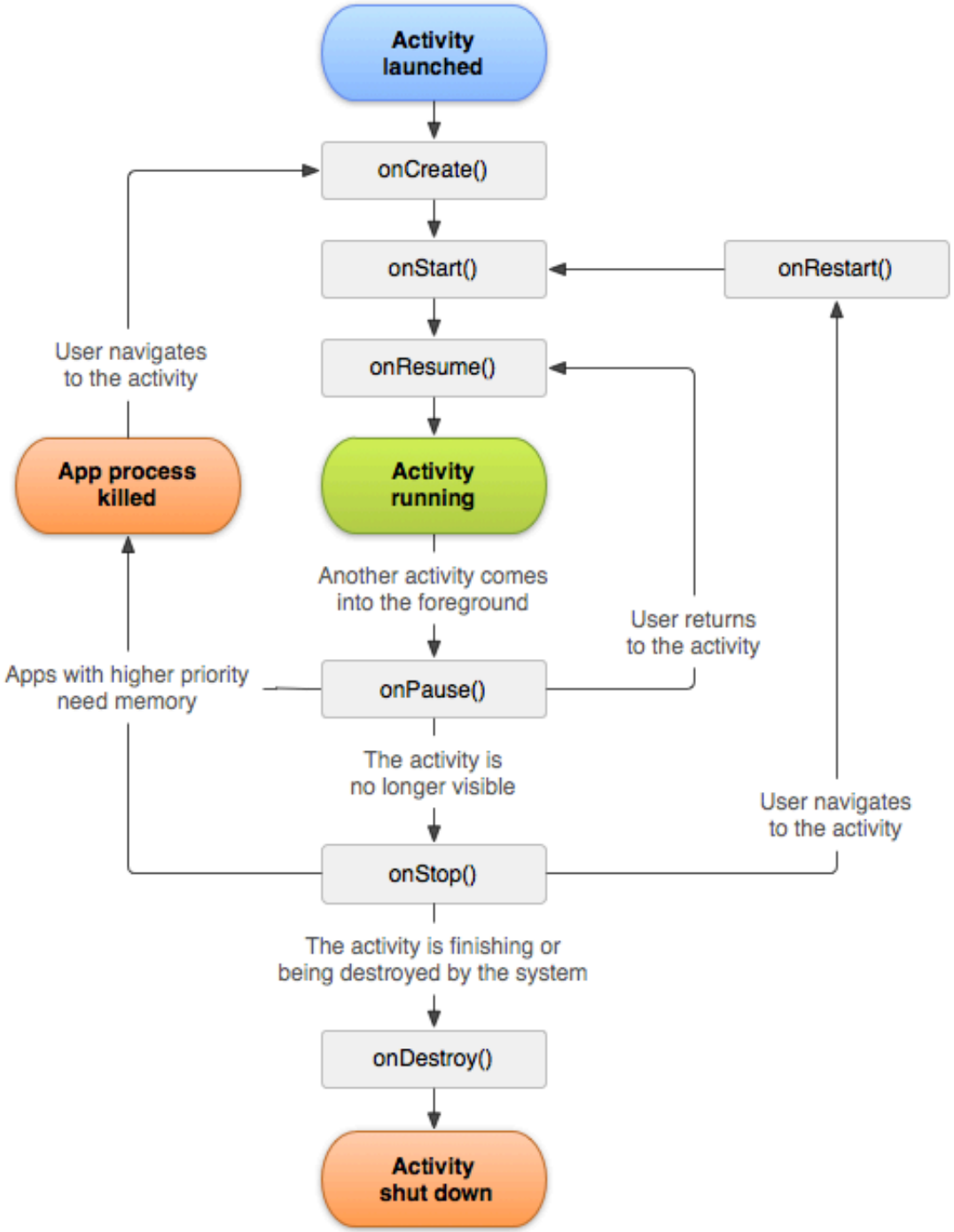
- le cycle de vie des activités, est clairement défini pour **éviter de surcharger le système**

- En tant que développeur, vous devez faire attention à **respecter les ressources** du device et à **les libérer** lorsque votre application est stoppée.

- http://developer.android.com/reference/android/app/Activity.html

# Cycle de vie d'une activité (2/2)






les **différents états** sont symbolisés pas les **rectangles colorés et arrondis**,

tandis que les **rectangles grisés** représentent les **méthodes de callback** que vous pouvez implémenter dans les phases de transition entre deux états.

Des méthodes **complémentaires** :

- onCreate()/onDestroy()
- onStart()/onStop()
- onResume()/onPause()

# Les méthodes de callback (1/3)

- **onCreate()** : appelée quand l'activité est crée la première fois. C'est ici que les déclarations statiques seront définies, les vues créées.

- **onRestart()** : appelée lorsque l'application a été stoppée et qu'elle est réactivée.

- **onStart()** : appelée juste avant que l'activité devienne visible à l'utilisateur.

- **onResume()** : appelée juste avant que l'activité interagisse avec l'utilisateur. L'activité est à cet instant tout en haut de la back stack

# Les méthodes de callback (2/3)

- **onPause()** : appelée quand le système est sur le point de rappeler une autre activité. Cette méthode est généralement **utilisée pour sauvegarder les données non persistées, stopper les animations ou toute autre action consommatrice de CPU**. Ces actions doivent être par contre exécutées très rapidement car la prochaine activité ne sera pas rappelée avant la fin de ces actions. Le système peut par contre couper le traitement s'il est trop long

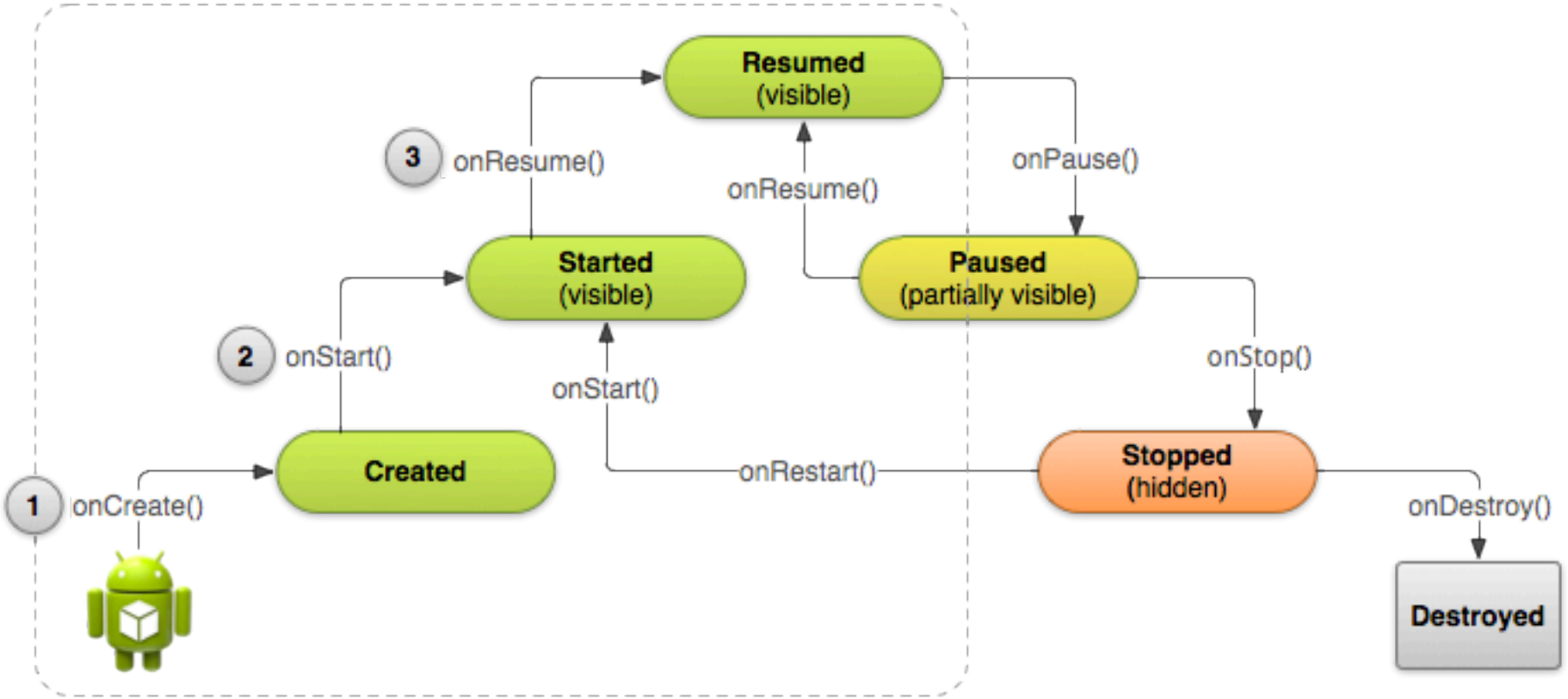
# Les méthodes de callback (3/3)

- **onStop()** : appelée quand l'activité n'est plus visible de l'utilisateur, quand l'activité est détruite ou quand elle est mise en arrière plan au dépend d'une autre activité. Le traitement de cette méthode peut être coupé s'il est trop long

- **onDestroy()** : appelée avant que l'activité soit détruite. Peut être appelée si l'activité est finie ou si le système a besoin d'espace mémoire. Tout comme les deux précédentes méthodes, le système peut couper le traitement si ce dernier prend trop de temps

# Création d'une activité

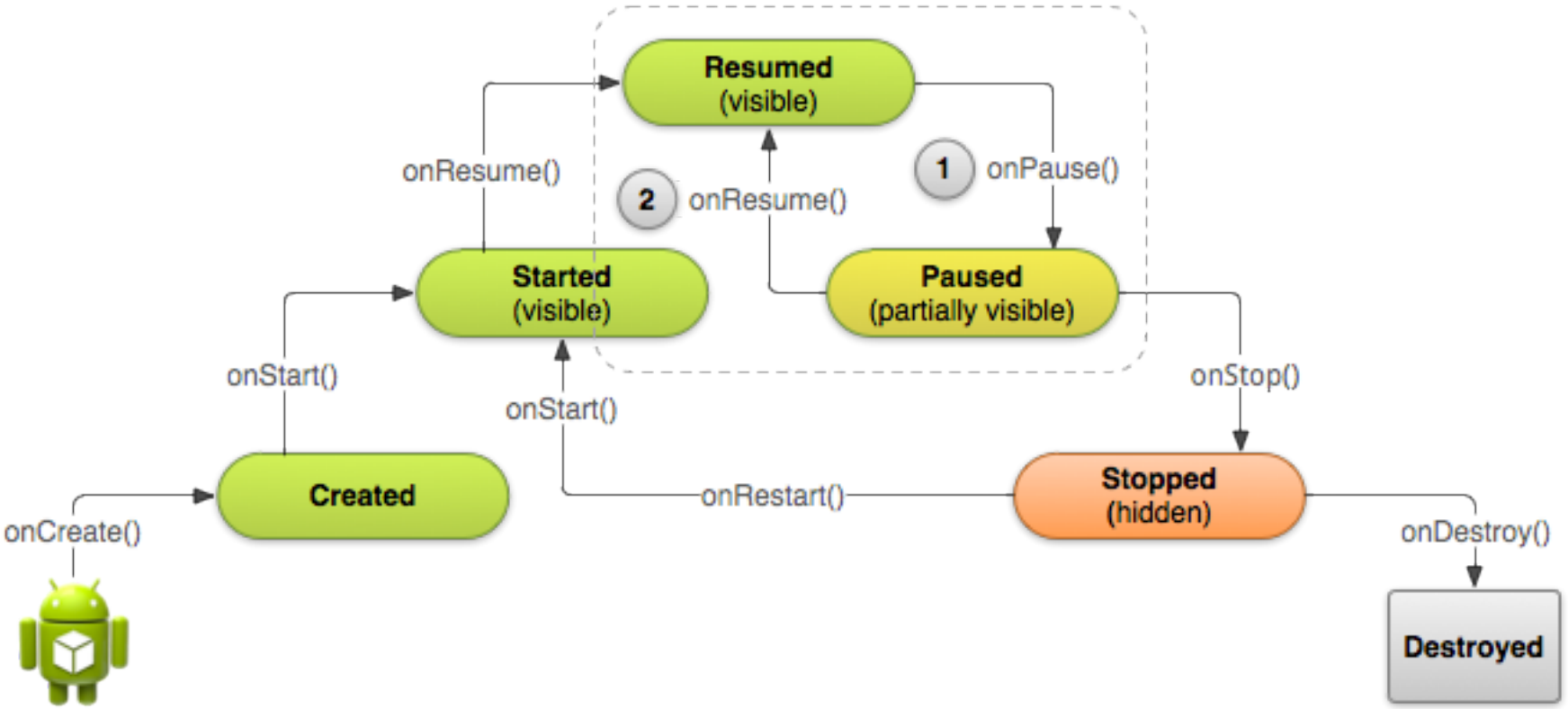
L'activité se crée en exécutant trois méthodes à la suite : onCreate(), onStart() et onResume().




http://developer.android.com/training/basics/activity-lifecycle/starting.html

# En suspend (1/2)

L’activité n'est plus que **partiellement visible**, comme quand elle est voilée par une boîte de dialogue.




http://developer.android.com/training/basics/activity-lifecycle/pausing.html

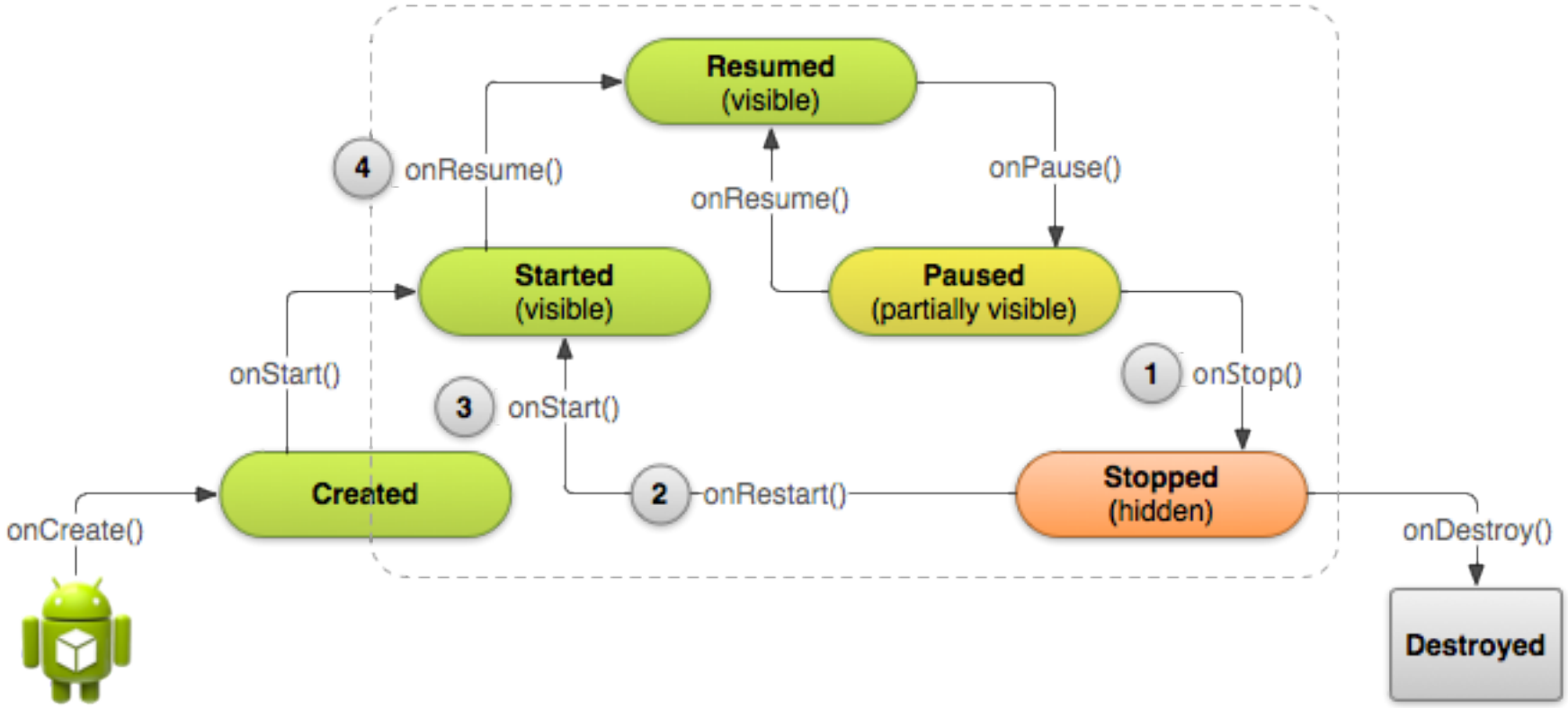
# En suspend (2/2)

- onPause() : arrêter des animations, libérer des ressources

- onResume() : initialiser les ressources

# En arrière-plan (1/2)

L'activité n'est **plus visible du tout**, mais elle n'est pas arrêtée non plus.




http://developer.android.com/training/basics/activity-lifecycle/stopping.html

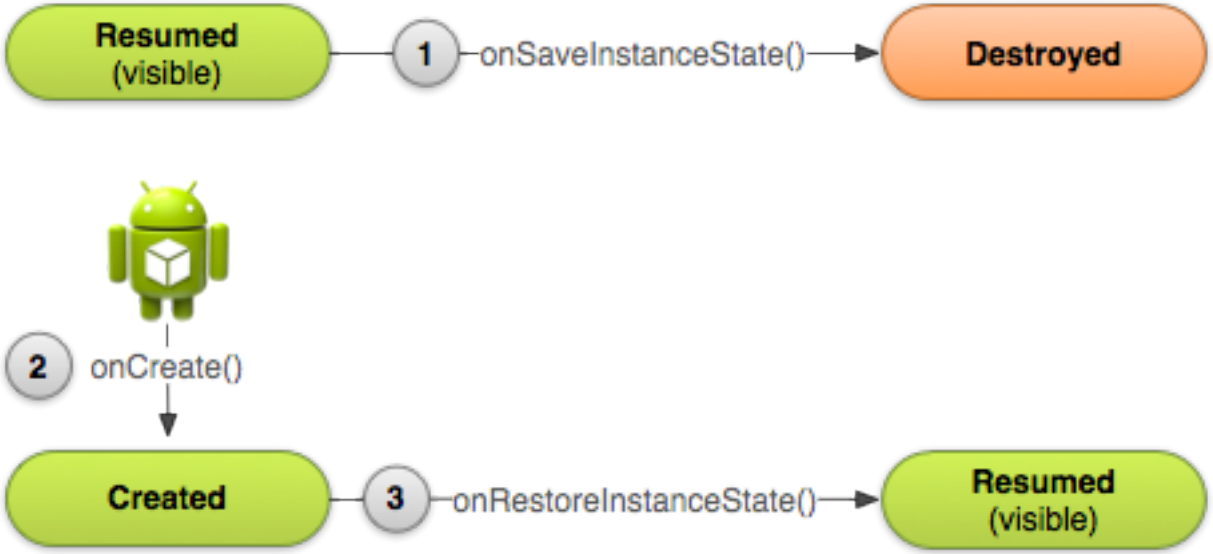
# En arrière-plan (2/2)

- onStop() : **sauvegarde** des éléments dans votre base de données

- onStart() ou onResume() : **restaurer** les éléments.

# Recréer une activité (1/5)

si Android a dû tuer le processus




http://developer.android.com/training/basics/activity-lifecycle/recreating.html

# Recréer une activité (2/5)

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {

    // Save the user's current game state
    savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
    savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);

    // Always call the superclass so it can save
    // the view hierarchy state
    super.onSaveInstanceState(savedInstanceState);
}
```

# Recréer une activité (3/5)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // Always call the superclass first
    super.onCreate(savedInstanceState);


    // Check whether we're recreating a previously destroyed instance
    if (savedInstanceState != null) {
        // Restore value of members from saved state
        mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
        mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);

    } else {
 // Probably initialize members with default values for a new instance
    }
    ...
}
```

# Recréer une activité (4/5)

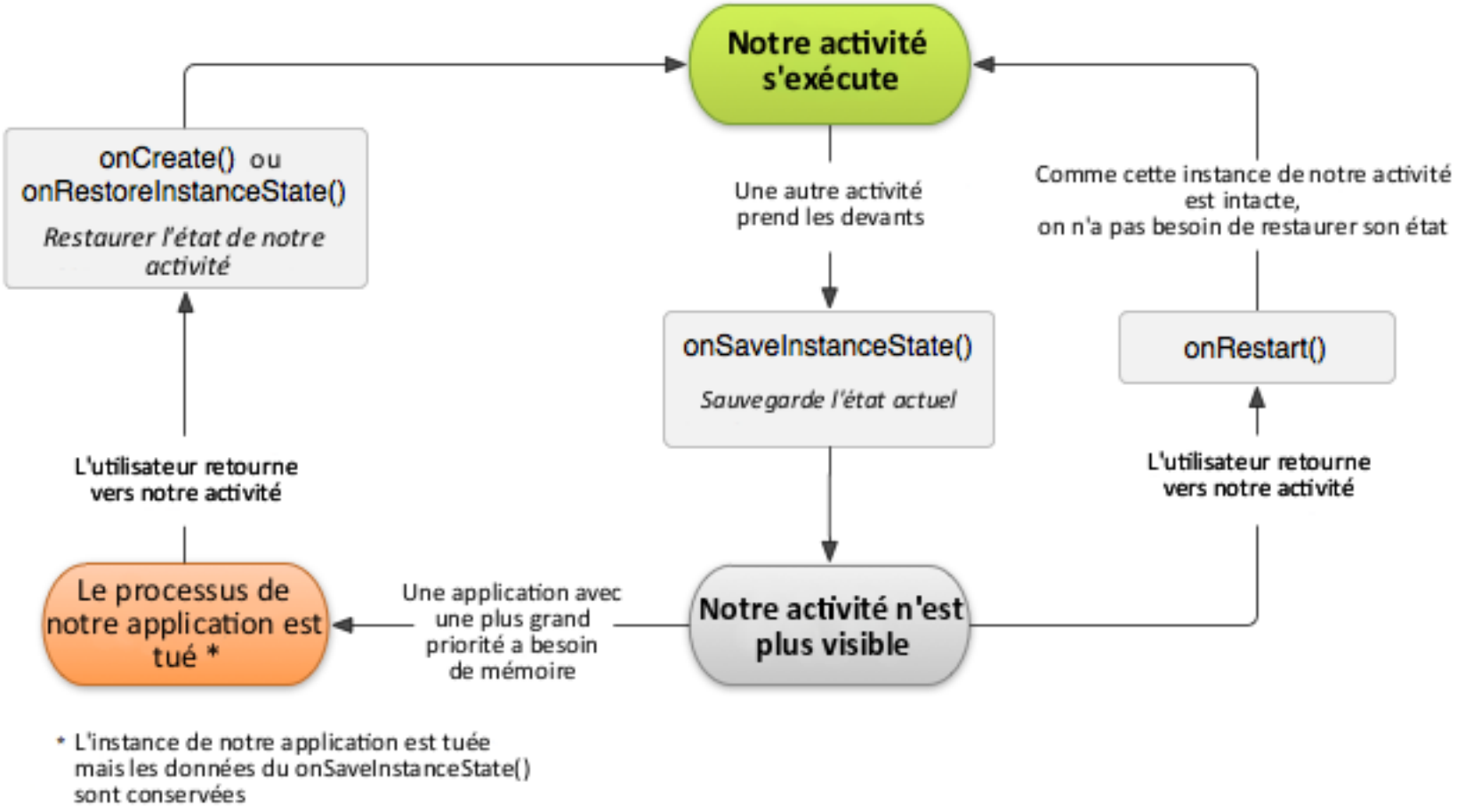
OU

```
public void onRestoreInstanceState(Bundle
savedInstanceState) {

    // Always call the superclass so it can restore
    // the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);

    // Restore state members from saved instance
    mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
    mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
}
```

# Recréer une activité (5/5)



# Source

- Openclassrooms
  - http://fr.openclassrooms.com/informatique/cours/creez-des-applications-pour-android/gerer-correctement-le-cycle-des-activites

- Managing the Activity Lifecycle
  - http://developer.android.com/training/basics/activity-lifecycle/index.html