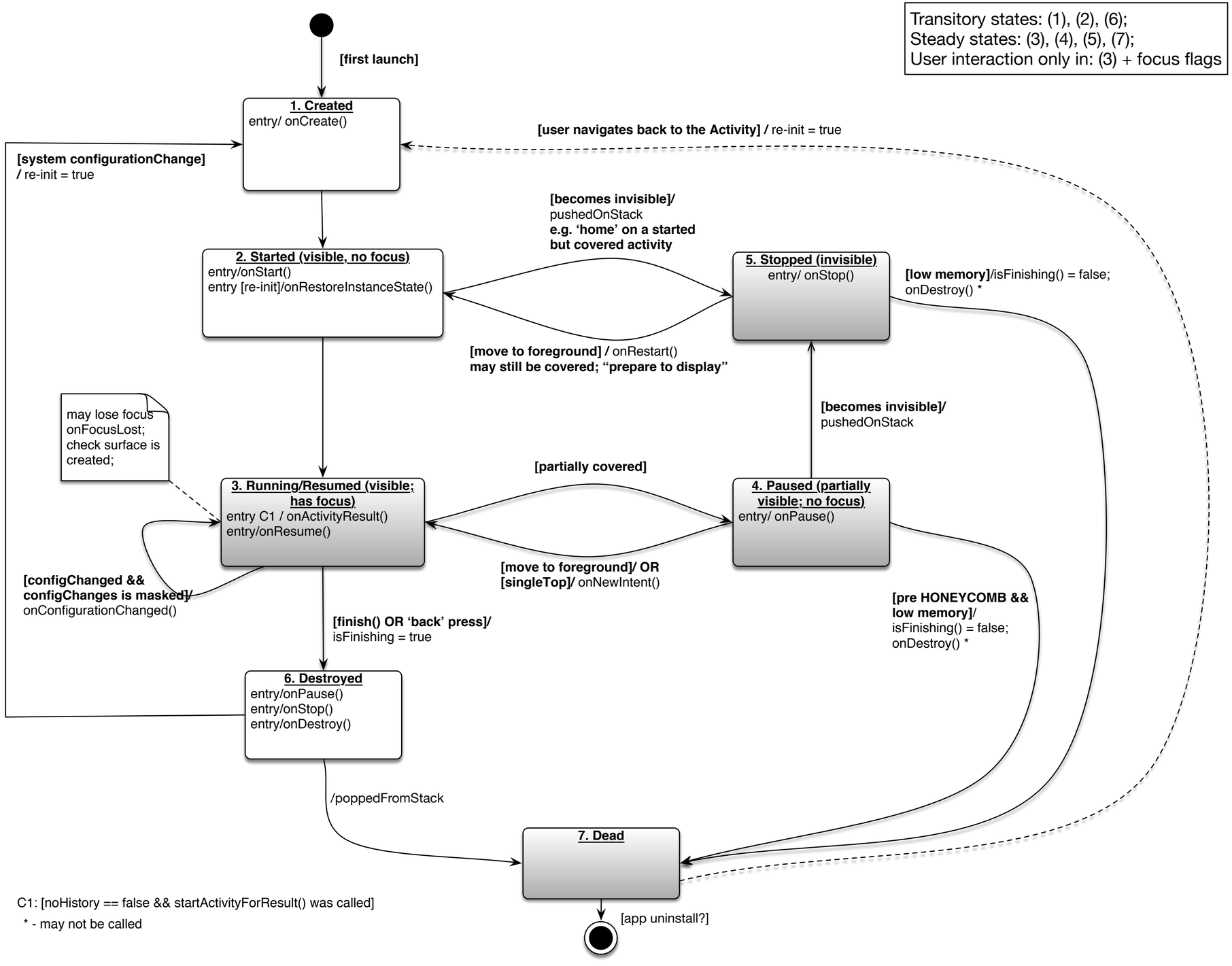

C1: [noHistory == false && startActivityForResult() was called]

* - may not be called

Transitory states: (1), (2), (6);
Steady states: (3), (4), (5), (7);
User interaction only in: (3) + focus flags

Guaranteed execution:
onCreate() -> onStart()
onRestart() -> onStart()
onResume() -> onPause()

All other callback methods may appear in different combinations, yet adhering partial order;

Focus flags:
onWindowFocusChanged(TRUE);
plus for games using SurfaceView: surfaceCreated && surfaceChanged

If left for a long time, the system clears the task of all activities except the root activity.

You probably should be loading native lib in `Application.onCreate` and not `Activity` [to account for killed app that might still think it has access to the native resources, but needs to reacquire them.]

Killable: Starting with Honeycomb (API 11), an application is not in the killable state until its `onStop()` has returned. This impacts when `onSaveInstanceState(Bundle)` may be called (it may be safely called after `onPause()` and allows and application to safely wait until `onStop()` to save persistent state.

It is important to save persistent data in `onPause()` instead of `onSaveInstanceState(Bundle)` because the latter is not part of the lifecycle callbacks, so will not be called in every situation.

Lifecycle callback onStateNotSaved() after API 23, what does it do?

Author: Konstantin Rubinov 02/2016

this diagram is extended from: http://aleung.github.io/blog/2010/12/16/Android-activity-lifecycle-in-UML-state-machine-diagram/