Bug 20141 - Request for native swipe syntax

Status: HIBERNATED
Version: 8.1.2 GM 1

Reported: 2017-07-22 20:40 EDT by Jacque Gay
Modified: 2021-03-09 15:59 EST
CC List: 7 users (show)

See Also:
Depends on:
Blocks:
Engine: Other

Show dependency tree / graph

Attachments
Add an attachment (proposed patch, testcase, etc.)

Jacque Gay 2017-07-22 20:40:01 EDT — Description

A request for swipe syntax and native behavior on mobile devices came up on the mailing list and it's a good idea. Here is my syntax suggestion:

```
lock screen for swipe effect
 switch pDir
   case "left"
     ...
   case "right"
     ...
   case "up"
     ...
   case "down"
     ...
 end switch
 unlock screen with swipe effect pDir
```

Right now, swipe gestures can only be approximated with visual effects. Native swiping behavior would allow LC to offer the same feature that other languages support.

Panos Merakos 2017-07-25 16:47:27 EDT — Comment 1

Hi Jacque,

Thank you for the report. This would be a useful addition.

We will update you when the status of this report changes.

Best regards,
Panos
--

Richard Gaskin 2017-07-28 19:59:07 EDT — Comment 2

Specification requested by Mark Waddingham:

A swipe gesture occurs when the user moves one or more fingers across the screen in a specific horizontal or vertical direction. iOS and Android provide APIs for recognizing and responding to this gesture, and most desktop OSes do too:

Android:
https://developer.android.com/design/patterns/swipe-views.html

iOS:
https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures/handling_swipe_gestures?language=objc

Windows:
https://msdn.microsoft.com/en-us/library/windows/apps/jj655416.aspx

Gnome:
https://developer.gnome.org/clutter/stable/ClutterSwipeAction.html

macOS:
https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/EventOverview/HandlingTouchEvents/HandlingTouchEvents.html

Swipes are most commonly used for two purposes:
- Transition from screen to screen (in LC that would be card to card)
- Perform an action on the object being touched (usually removing from a list)

Jacque's original request was for card-to-card transitions, and if it's easier to implement that first I believe that would also be the most valuable.

If object-specific swipes are of interest that will take more detail to specify, perhaps best left for a separate request.

So here we'll only focus on card-to-card transitions for now:

In many apps that have tabbed views we find that we can switch tabs at least two ways, either by clicking a tab icon, or swiping horizontally takes us to the next tab in the direction of the swipe.

We're somewhat close now with the push visual effect, portions of which may be useful for this implementation. The main difference is that once the push effect initiates it continues until the transition is complete, whereas a swipe follows the finger, and completes only if the finger is raised from the screen after a certain distance threshold. If released anywhere shorter than that threshold the view slides back to its original state.

So the flow looks more or less like:

1. Swipe detected (direction determined, dest card buffered)
2. Both old and new card visible on screen, tracking finger movement (as the finger slides one direction or another both images move along with it).
3. IF the finger is raised from the screen before the commit threshold distance:
   3a: THEN we complete the transition by having the new card push the old card out of view, arriving at the new card.
   3b: ELSE the buffered pair of rendered card images slides back to the original state, in which we remain on the old card.

One open question is what specifically that threshold should be. It may be defined in the respective OS specs, and if so that's one less thing to decide. :) If we do need to decide it we'll want to review a reasonable number of apps that use swipe transitions and aim for whatever they use that make them all feel reasonably consistent. I would be happy to help gather those metrics if needed.

A bigger question is how we specify the target card. I like Jacque's suggested syntax for a scriptable option (very similar to what we have with visual effects already, keeping things consistent), though would not be opposed to a property-driven alternative if it were easier to implement.

If other details are needed let's discuss.

Jacque Gay 2017-07-28 20:26:47 EDT **Comment 3**

A generic metric for determining a completed swipe might be the card width/height divided by 2.

Speed would probably need to be accounted for. A fast swipe that isn't half the

default distace is probably still intended to be a completed gesture.

Mark Waddingham 2017-07-28 21:01:50 EDT **Comment 4**

This is great - it makes it very clear :)

So, the only technicality I can see *right* now is the question of openness of cards. In particular, are there any issues with the following occurring:

```
Swipe starts.
Screen Locked.
Current card is snapshotted.
Current card is closed.
Next card is opened.
Next card is snapshotted.
Swipe continues.
If swipe ends successfully:
  All done
Else If swipe is cancelled:
  Next card is closed.
  Current card is reopened.
```

I'm not sure there's another way to handle this without breaking the existing invariant that there are never more than two cards open on a stack at once (which has rather deep implications for fields and their data, in particular).

Richard Gaskin 2017-07-28 21:12:25 EDT **Comment 5**

The sequence seems okay to me with one modification: in my own mind it seems to make more sense if the new card is never opened per se unless a swipe is completed. If a swipe is begun but cancelled, no change occurs.

Does that make sense?

Mark Waddingham 2017-07-28 21:43:31 EDT **Comment 6**

The reason you need to open the next card is so that you can take a snapshot of it in the way it will be when you arrive at it.

When you do:

```
lock screen for visual effect
go card X
unlock screen for visual effect
```

The step of open/closes and snapshots as outlined above happens. This means that the image which is transitioned to is identical to how the screen will look when the visual effect finishes.

If you don't open the (target) card to take the snapshot, it won't be in the state it should be in.

Jacque Gay 2017-07-28 21:46:33 EDT **Comment 7**

So all open/close messages would be sent, right? And if the user cancels, we might need to undo any changes to both cards?

Mark Waddingham 2017-07-28 21:54:28 EDT **Comment 8**

Okay so the current suggestion came from the idea of extending visual effects (for all intents and purposes). In that case, there is no cancellation, so the operation is:

```
  snapshot current card
  lock screen
  go next card (i.e. close this, open next)
```

```
  snapshot next card (actually now current card!)
  unlock screen after performing transition from current to next

So, the natural extension here is to change it as follows:

  snapshot current card
  lock screen
  go next card
  snapshot next card
  track gesture to determine visual effect state
  if gesture is not cancelled then
    unlock screen
  else
    go previous card (i.e. the original current card)
    unlock screen
  end if

So, as long as your open/close stuff is suitably symmetric, then it shouldn't be a
problem. Of course, that is a big 'if' - particularly if an app does a lot of setup
/ teardown in those messages.
```

Mark Waddingham 2017-07-28 22:00:44 EDT [Comment 9]

```
Of course, we're talking about engine functionality here - but this could be
prototyped and experimented with in LiveCode Script.

Indeed, it could be done as a script library which temporarily installs a
frontscript to capture input during the swipe.

So perhaps the critical engine piece here is cross-platform swipe. Mobile swipe can
be done in script very effectively; desktop swipe is more difficult as it is two-
finger gesture. Indeed, this also has (in full generality) a prerequisite two other
things:

   1) Touch support on desktop
   2) Gesture support on all platforms

Putting the actual animation into the engine does have an advantage though - on
mobile we can leverage the special way we handle the core visual effects (we create
a system layer to show / move the parts of the transition - which makes them nice
and smooth).
```