

Windowing Management

Interaction Design Specification

Version: 0.4
Revised September 10, 2010 5:37 PM
Created by Martin Gimpl

Version History

0.4	10-09-10	– Added information on the conceptual model of the window stack [4.1 Introduction to the window stack] – Added information on gestures that start as a drag, but end as a flick [2.11 Drag + flick gestures] – Added information on the sleep/awake system for WPC navigation [4.2.1 Application sleep and wake] – Added more information on reveal [4.3 Revealing the Desktop] – Added an example of quitting a dual-screen application [3.4 Quitting applications] – Fixed rules on modals when rotating dual-screen applications [5.3 Modals when rotating dual-screen applications]
0.3	10-09-03	– Changed rotation rules to keep maximized state when rotating from LD to PD [5.2 Rotating dual-screen applications] – Added more rotation examples (360 degree turn with two single screen applications) [5.1 Introduction to rotation rules] – Added rules on modals when rotating dual-screen applications [5.3 Modals when rotating dual-screen applications] – Added information on closing first app in stack when device is closed [3.4 Quitting applications]
0.2	10-08-27	– New rules for LD dual-screen application maximization and minimization [2.12 WPC rules in landscape orientation] – New rules for rotating parent-child dual-screen applications [5.2 Rotating dual-screen applications] – Added information on the window stack and triggering a desktop reveal [4.2 Navigating the window stack], 4.3.4 Revealing the desktop after reaching the end of the window stack] – Revised rules for opening and closing the device [4.4 Window stack behavior during device opening and closing] – Removed misleading minimization example [2.1 Introduction to maximization and minimization rules] – Fixed wire of a maximized parent-child dual-screen application [3.5 Launching and quitting applications in landscape orientation]
0.1	10-07-06	– Initial release

Table of Contents

Version History.....	2	4.2.1 Application sleep and wake.....	43
Table of Contents.....	3	4.3 Revealing the Desktop.....	44
Introduction to WPC transitions.....	4	4.3.1 Revealing the Desktop when the device is closed.....	45
Motion inventory.....	4	4.3.2 Revealing the Desktop when the device is opened.....	46
Open issues.....	5	4.3.3 Dual-screen applications and application launching during reveal.....	47
Legend.....	6	4.3.4 Revealing the desktop after reaching the end of the window stack.....	48
1 Gestures.....	7	4.4 Window stack behavior during device opening and closing.....	49
2 Maximization and minimization rules.....	9	4.5 Dual-screen applications during device opening and closing.....	51
2.1 Introduction to maximization and minimization rules.....	10	5 Rotation.....	53
2.2 Maximizing parent/child-views: root view.....	13	5.1 Introduction to rotation rules.....	54
2.3 Maximizing parent/child views: node views.....	14	5.2 Rotating dual-screen applications.....	55
2.4 Maximizing parent/child views: leaf view.....	15	5.3 Modals when rotating dual-screen applications.....	56
2.5 Minimizing parent/child views.....	16	6 Keyboard.....	57
2.6 Maximizing exposé views.....	18	6.1 PD keyboard behavior during WPC events.....	58
2.7 Minimizing exposé views.....	19	6.2 PD keyboard during minimization.....	59
2.8 Modal behavior during minimization.....	20	6.3 LD keyboard behavior during WPC events.....	60
2.9 WPC navigation rules.....	21	6.4 Keyboard rotations.....	61
2.10 Long-dragging parent-child views.....	23		
2.11 Drag + flick gestures.....	24		
2.12 WPC rules in landscape orientation.....	25		
3 Application launching and quitting.....	27		
3.1 Introduction to application launching and quitting.....	28		
3.2 Launching applications.....	29		
3.3 Auto-minimization during application launch.....	31		
3.4 Quitting applications.....	34		
3.5 Launching and quitting applications in landscape orientation.....	38		
4 Window stack & opening and closing the device.....	40		
4.1 Introduction to the window stack.....	41		
4.2 Navigating the window stack.....	42		

Introduction to WPC transitions

The transitions seen during WPC operations, like moving an application to the other screen, maximizing a dual-screen capable application to two screens, or minimizing one to a single-screen are generally performed with off-screen gestures.

Due to the nature of gestural interaction, it is paramount that the transitions respond immediately to the finger and maintain a high frame rate during the transition; failure to do so results in an immediate disconnect between the user and the system, making WPC interaction feel unresponsive and unreliable.

Motion inventory

MOT.1.1 Rotate device.....54

MOT.1.2 Gravity drop.....55

MOT.1.3 Maximize.....11

MOT.1.4 Minimize.....12

MOT.1.5 Reveal.....44

MOT.3.3 Launch App.....28

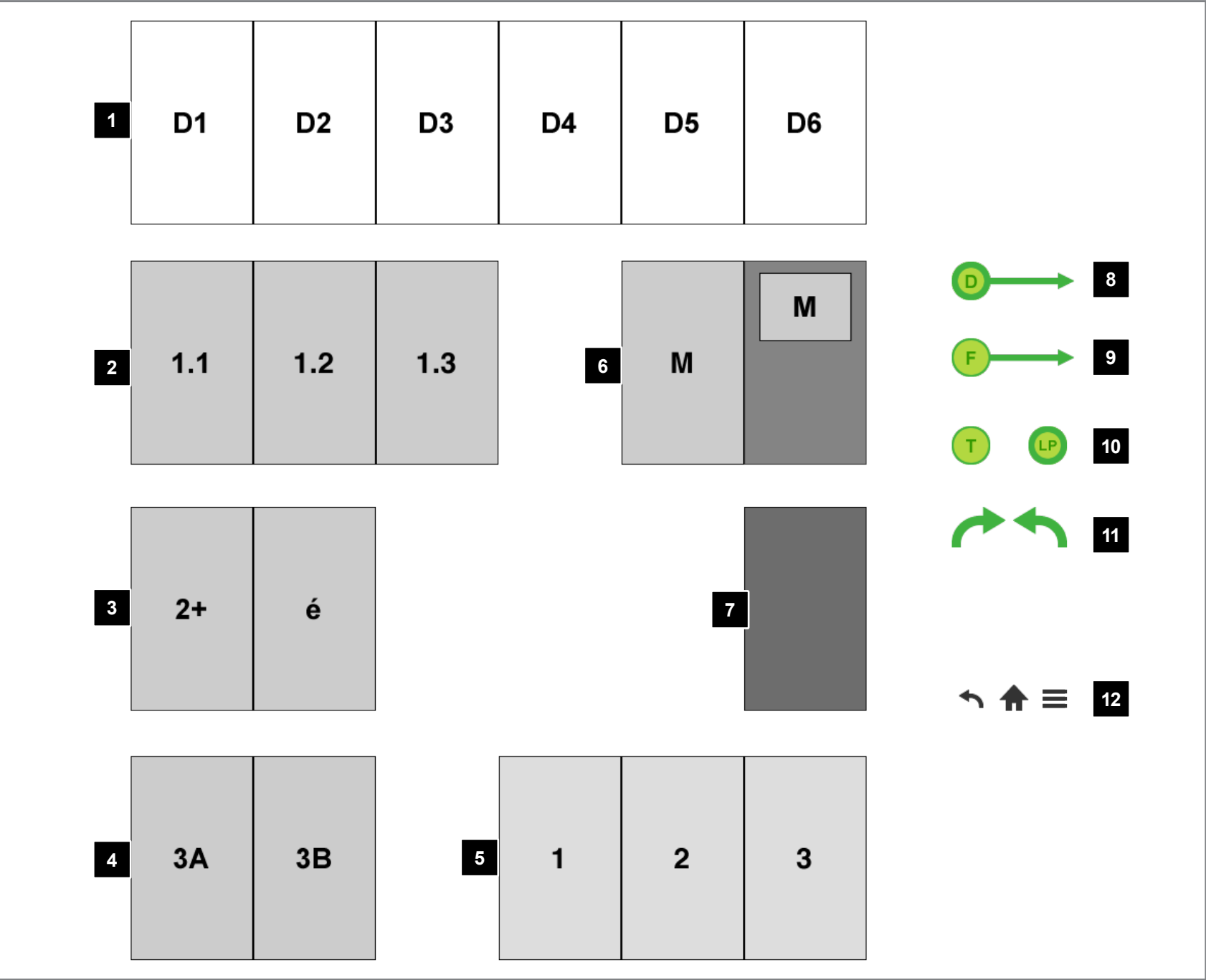
MOT.3.4 Quit App.....28

MOT.4.7 Full-Screen Modal Appear 28

MOT.4.8 Full-Screen Modal Disappear.....28

Open issues

- WPC performance issues when navigating between applications.
- When launching new apps from within maximized apps, should we retain the triggering view after minimization?



Legend

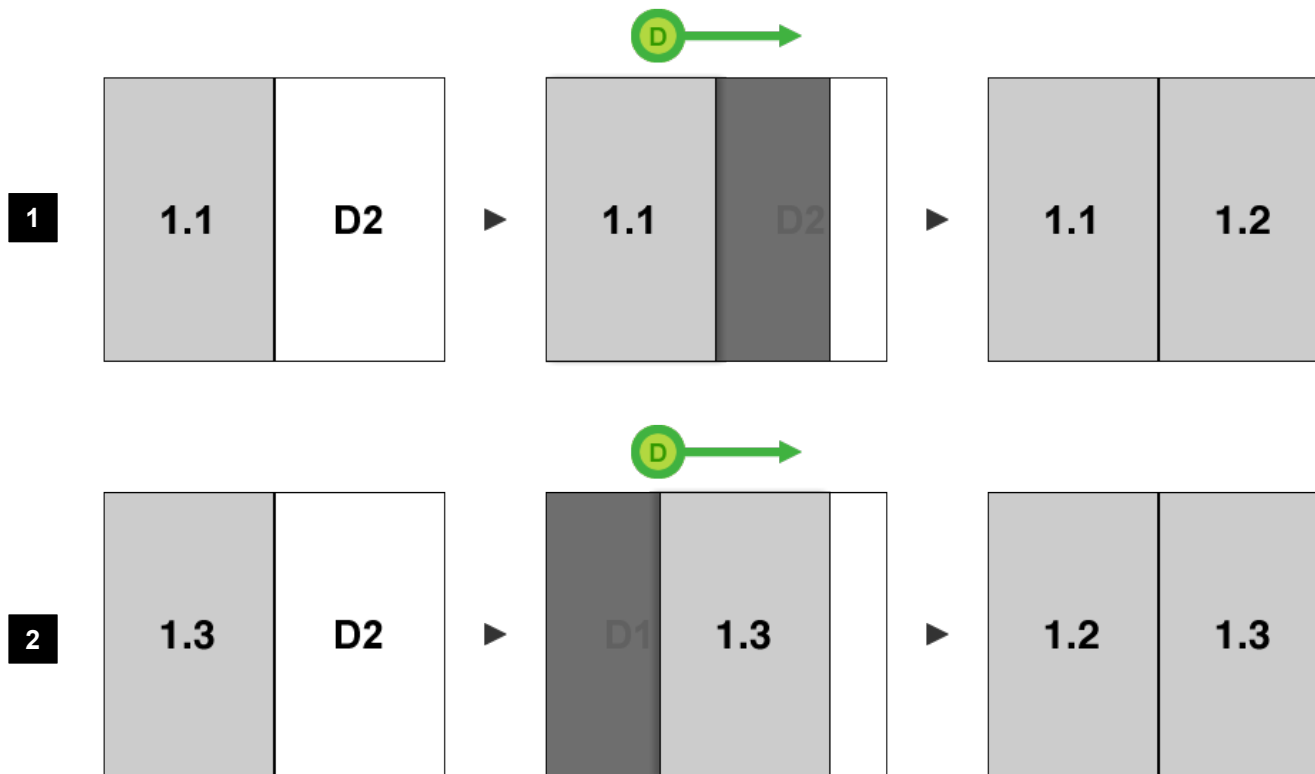
- 1. Desktops
 - Always layered underneath the applications.
 - The desktops cannot be reordered.
- 2. Application with a parent/child structure
 - 1.1 is the root of the application (view with no parent), 1.3 the leaf (view with no children).
 - 1.1 is the parent of 1.2, which is the parent of 1.3.
- 3. Application with an exposé structure
 - 2+ is the main view of the application, é is the exposé view.
- 4. Application with a parallel structure
- 5. Single-screen applications
- 6. Modal views
 - Modals can take over either the entire, or part of a view.
 - Modals dim out the view underneath them, as well as any other view within the application structure that is visible in PD orientations.
- 7. Empty view (well)
 - Used during WPC transitions when the actual view cannot be rendered
- 8. Off-screen drag gesture
- 9. Off-screen flick gesture
- 10. On-screen tap and long-press gestures
- 11. Rotation of the device clockwise and counterclockwise
- 12. Hardware buttons

1 Gestures

The following off-screen gestures are supported by the WPC:

Tap	Currently not utilized
Long-press	Currently not utilized
Drag	1. Moves applications from one screen to another 2. Maximizes and minimizes dual-screen capable applications
Long-drag	Considered a sequence of continuous drag operations.
Flick	1. Moves applications from one screen to another 2. Minimizes dual-screen applications
Pin & drag	Holds the pinned application in place while moving the other applications in the application stack. The pinned application always remains in view.
Pinch	Displays the Application Manager
Spread	Reveals the Desktop

2 Maximization and minimization rules



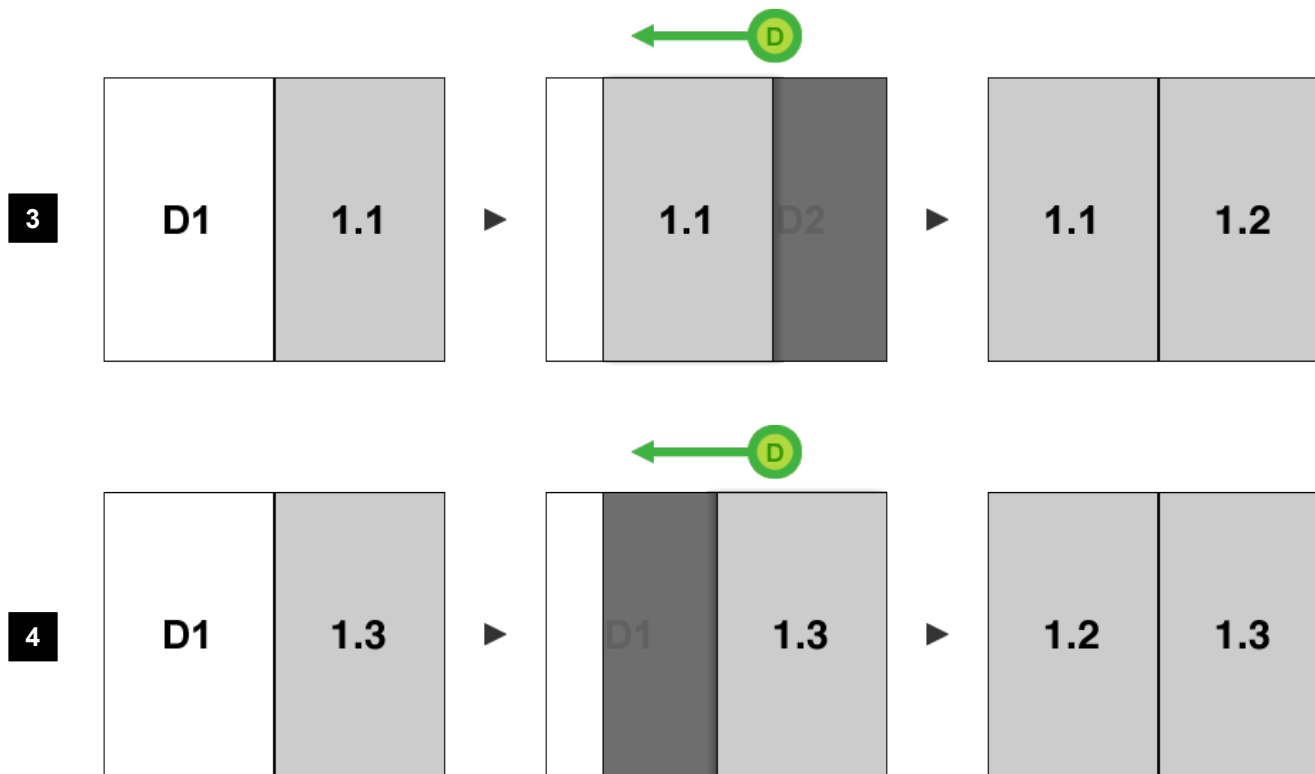
2.1 Introduction to maximization and minimization rules

From the point of view of the WPC system, applications are either maximized to span across two screens, or minimized to one screen; the application (and its internal structure) determines the details on that action, ie which of views move during maximization, and which remains visible during minimization.

During a maximization action, due to technical restrictions it may not be possible to show the uncovered view. In these cases, an empty view is shown, instead.

Examples of maximization scenarios:

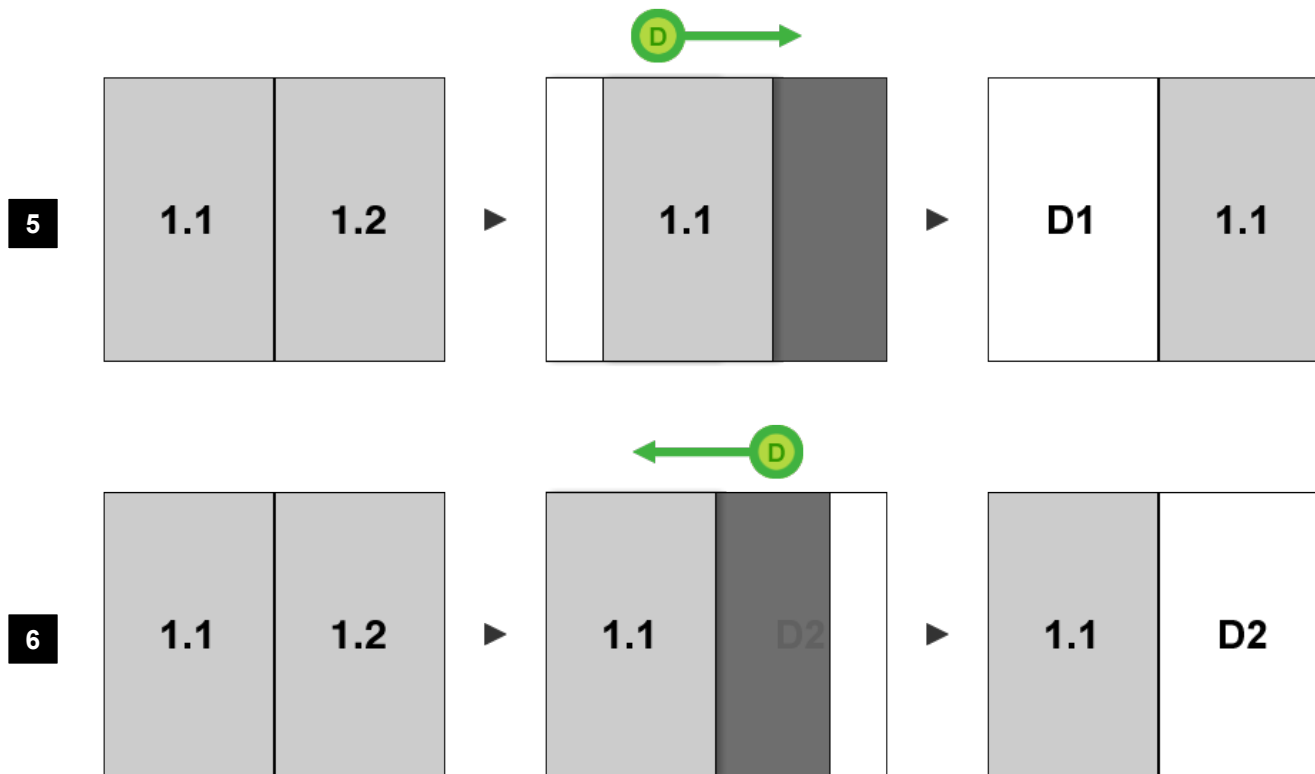
1. A root view on the left screen is maximized, sliding its child view on the right screen.
2. A leaf view on the left screen is moved to the right screen, and its parent is uncovered on the left screen



3. A root view on the right screen is moved to the left screen, uncovering its child view on the right screen.
4. A leaf view on the right screen is maximized, sliding its parent view onto the left screen

The motion for maximizing a window is:

MOT.1.3 Maximize

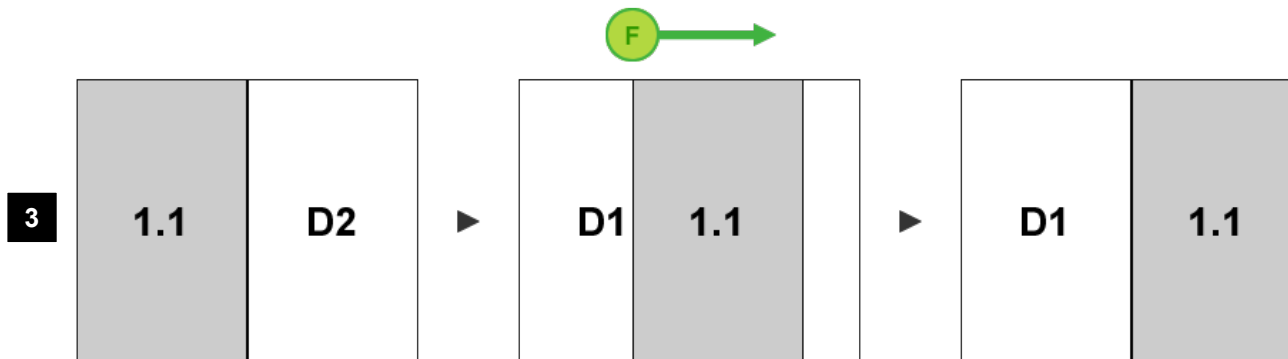
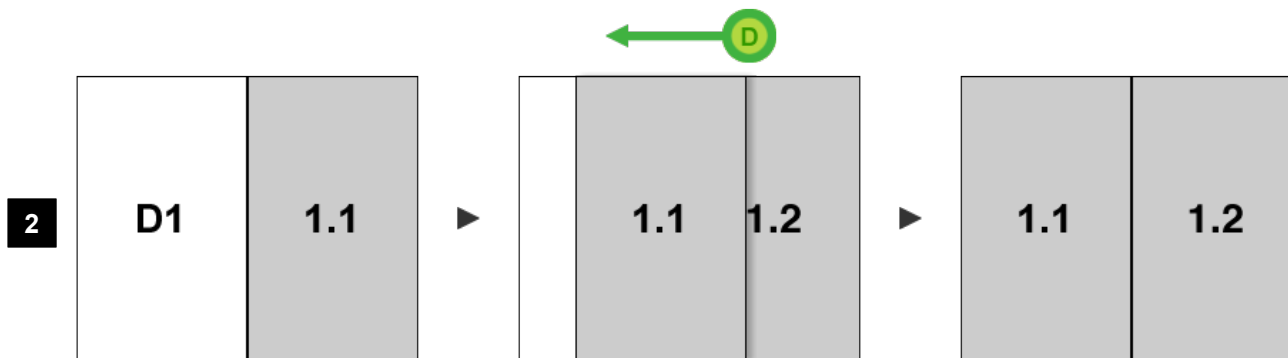
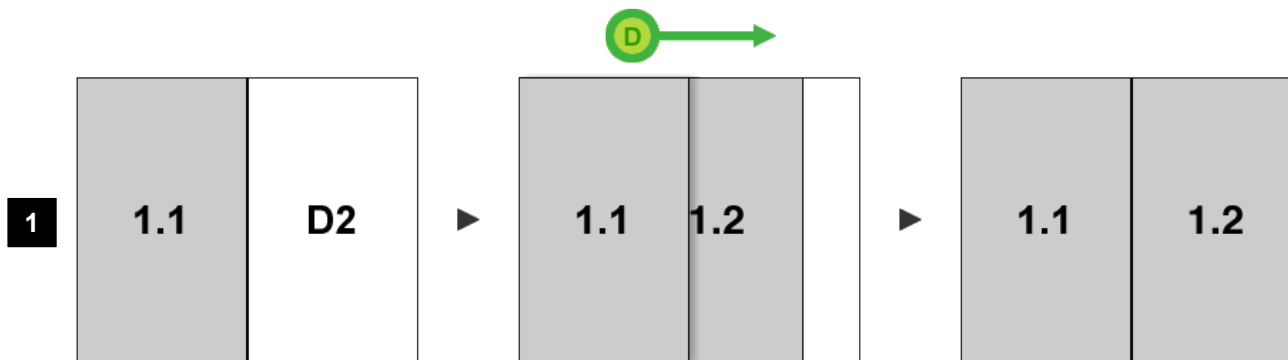


Examples of minimization scenarios:

5. A parent view on the left screen is moved on top of its child view on the right screen.
6. A child view on the right screen is moved underneath a parent view on the left screen.

The motion for minimizing a window is:

MOT.1.4 Minimize



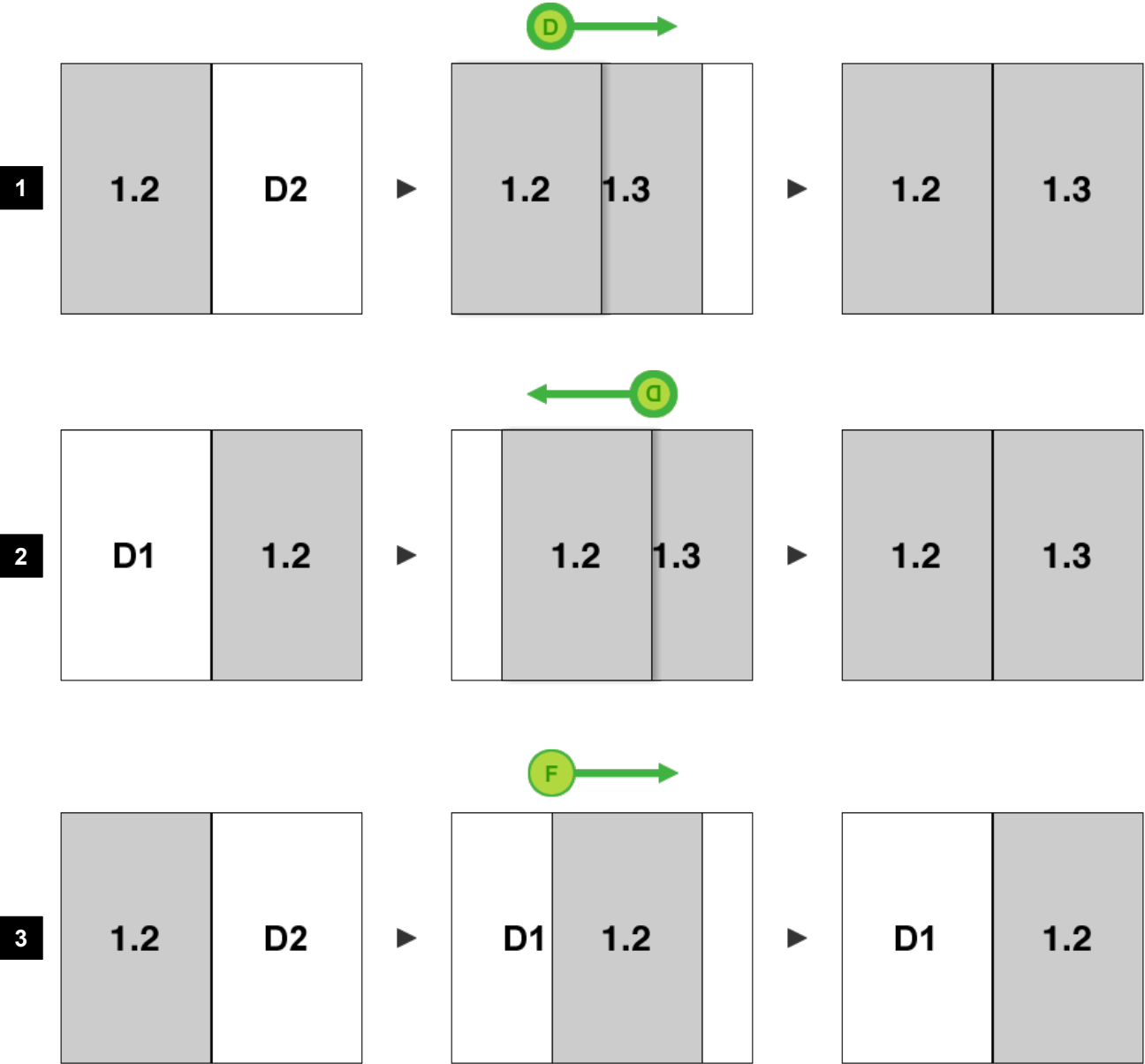
2.2 Maximizing parent/child-views: root view

As a rule of thumb, whenever children exist, they are uncovered on the right screen.

1. Maximizing a root view from the left screen always uncovers the child view on the right screen.
2. Maximizing a root view from the right screen always moves it to the left screen, and uncovers its child on the right screen.
3. A flick will move the entire root view to the other screen.

2.3 Maximizing parent/child views:
node views

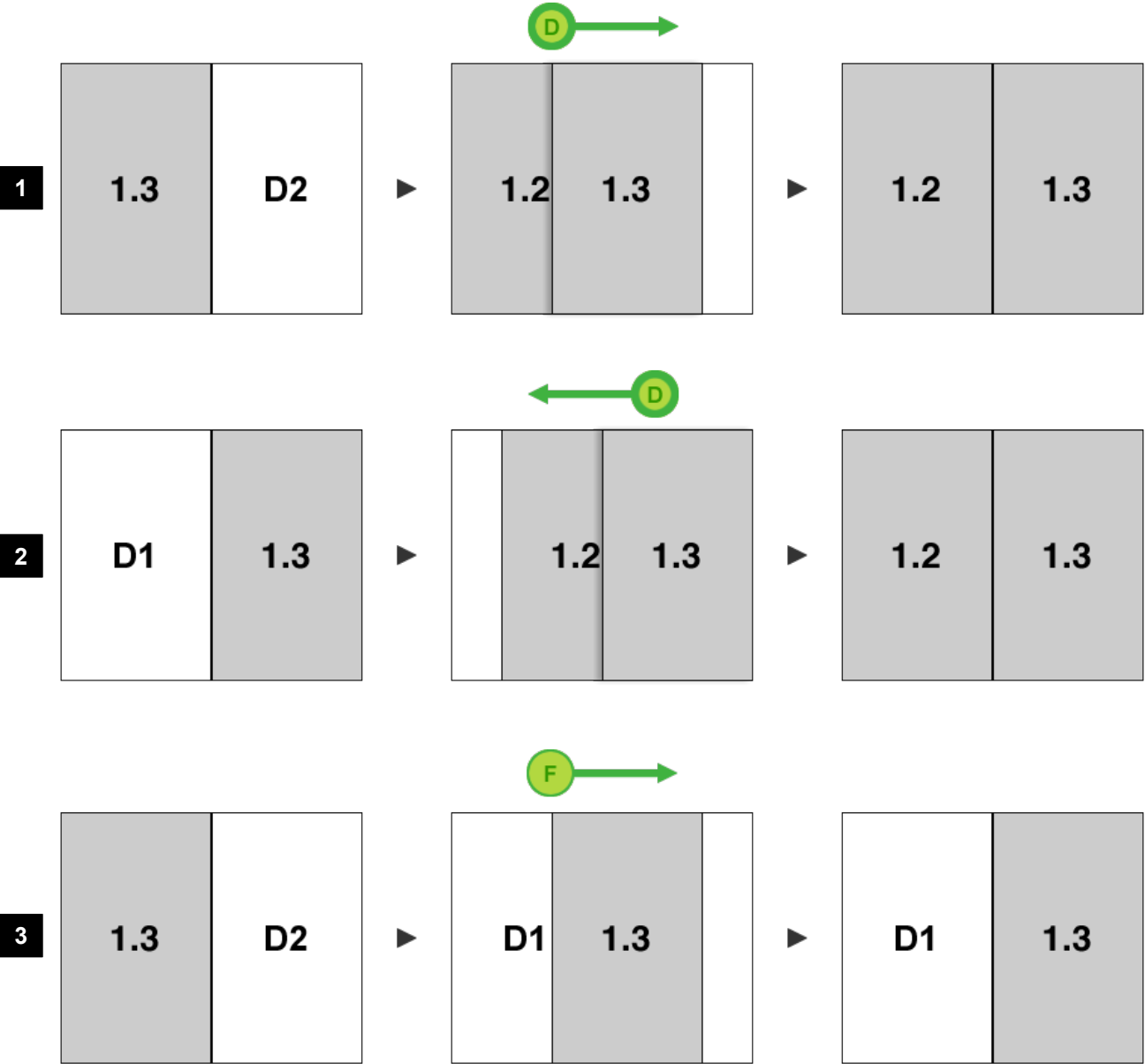
- 1. Maximizing a node view from the left screen will always uncover its child view on the right screen.
- 2. Maximizing a node view from the right screen will always move the node to the left screen and uncover its child on the right screen.
- 3. A flick will move the node view to the other screen.



2.4 Maximizing parent/child views:
leaf view

As a rule of thumb, if no children exist, the leaf view is positioned on the right screen.

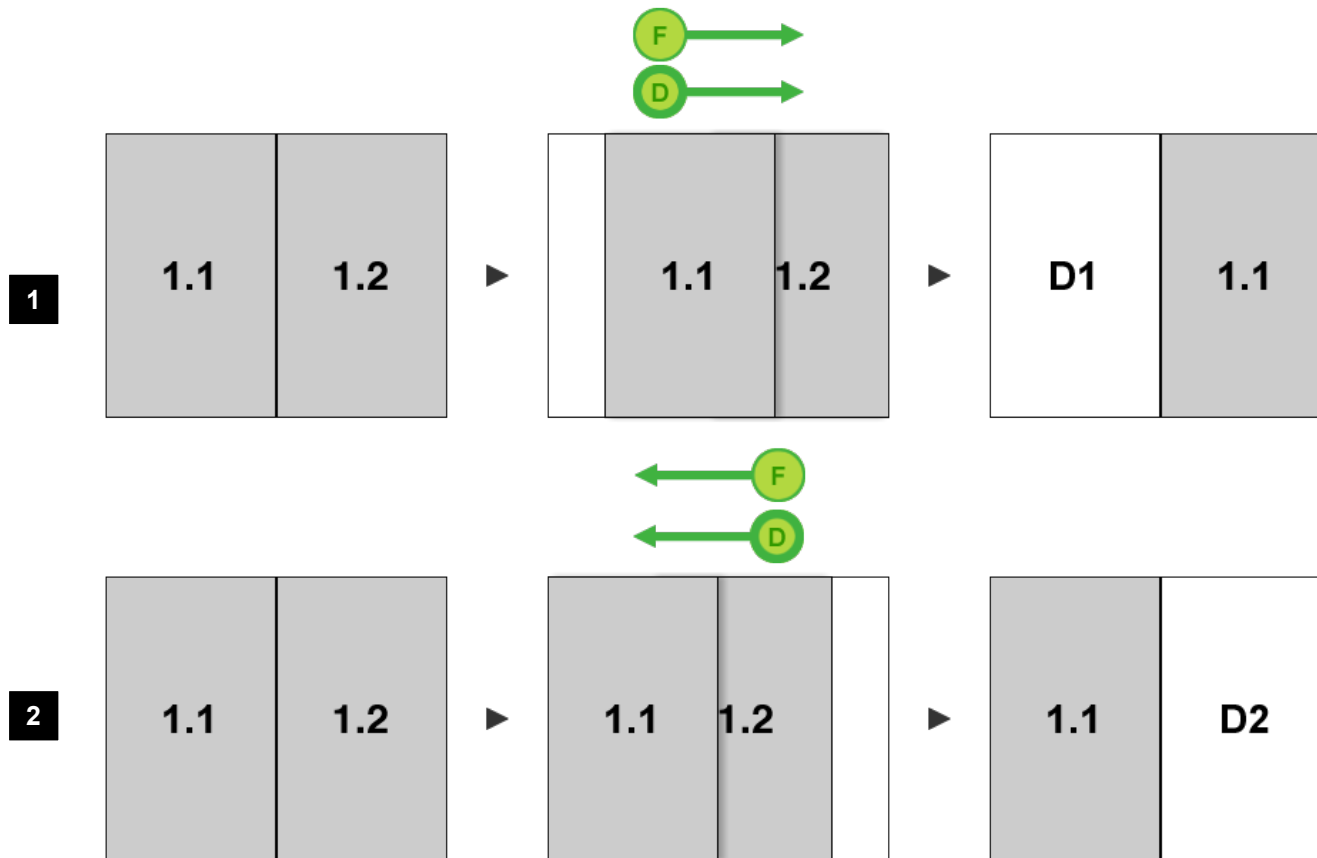
- 1. Maximizing a leaf view on the left screen will move the leaf to the right screen and uncover its parent on the left screen.
- 2. Maximizing a leaf on the right screen will uncover its parent on the right screen.
- 3. A flick will move the leaf to the other screen.

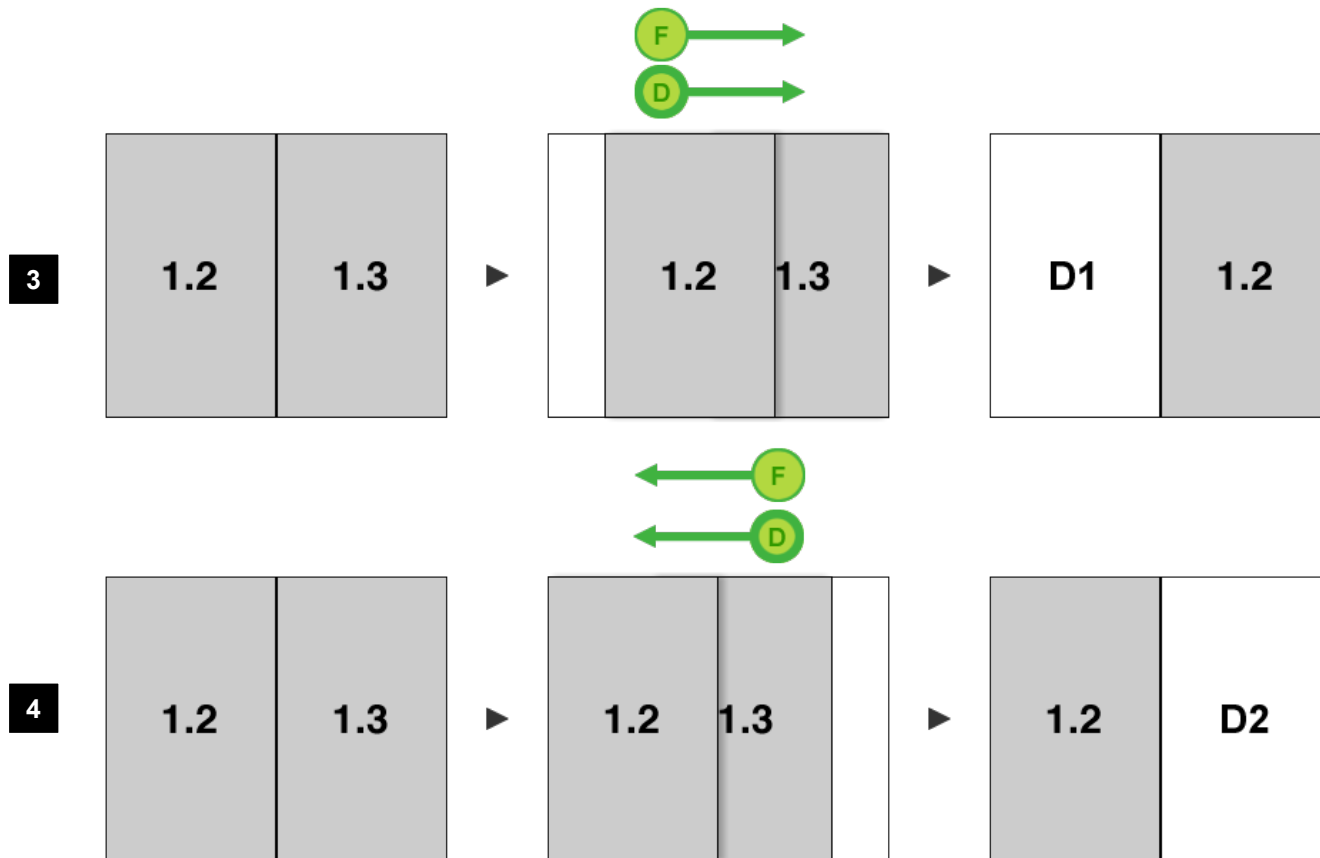


2.5 Minimizing parent/child views

As a rule of thumb, the parent view is retained whenever possible.

1. Minimizing a root+node view from the left screen moves the parent view to the right screen, covering the child view
2. Minimizing a root+node view from the right screen moves the child view to the left screen, sliding below the parent view



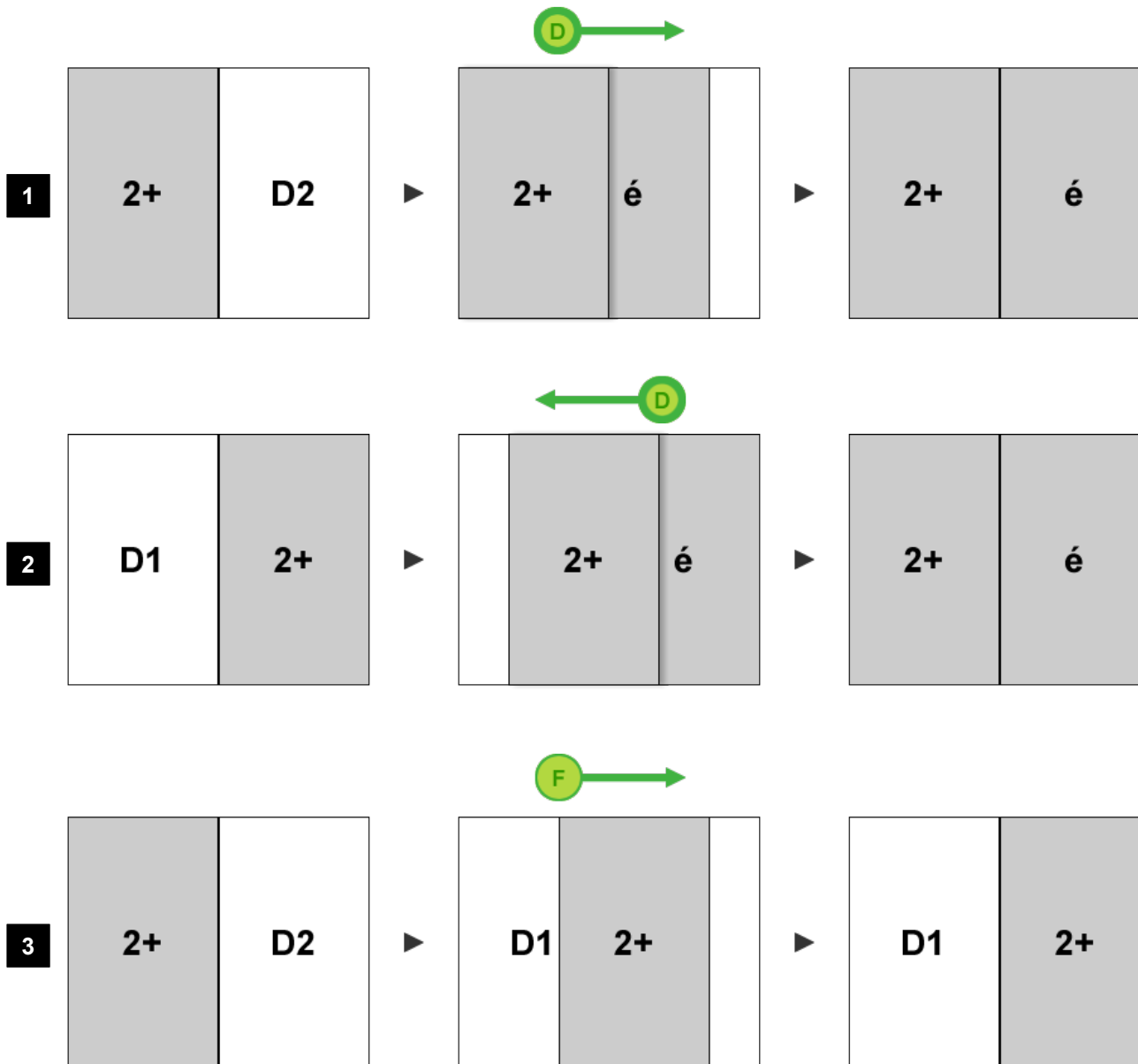


3. Minimizing a node+leaf view from the left screen moves the parent view to the right screen, covering the child view.
4. Minimizing a node+leaf view from the right screen moves the child view to the left screen, sliding below the parent view.

2.6 Maximizing exposé views

As a rule of thumb, exposé views are always on the right side of their parent views.

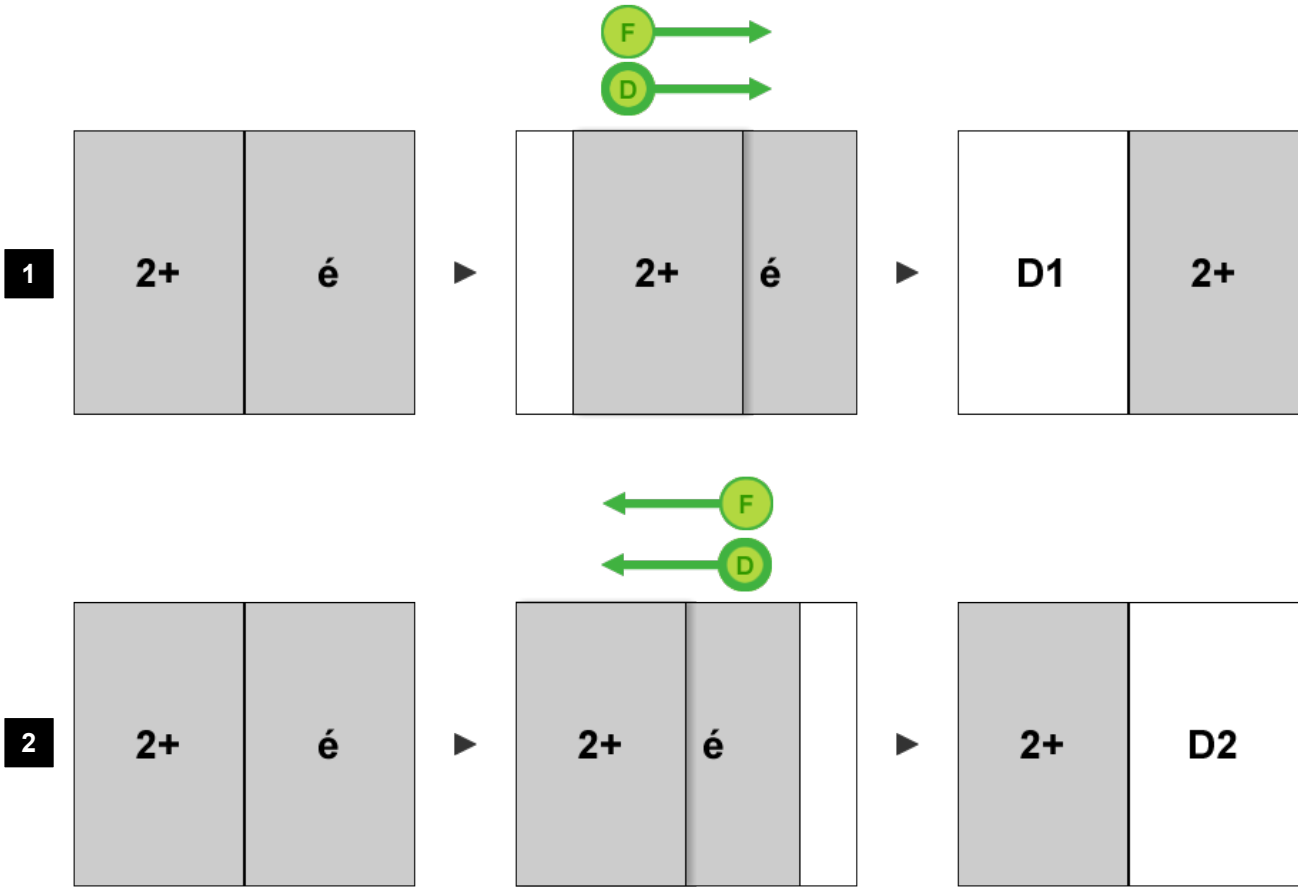
1. When maximizing an exposé view from the left, the parent stays on the left
2. When maximizing an exposé view from the right, the parent moves to the left
3. A flick will move the parent to the other screen.



2.7 Minimizing exposé views

As a rule of thumb, exposé views may not cover their parent view

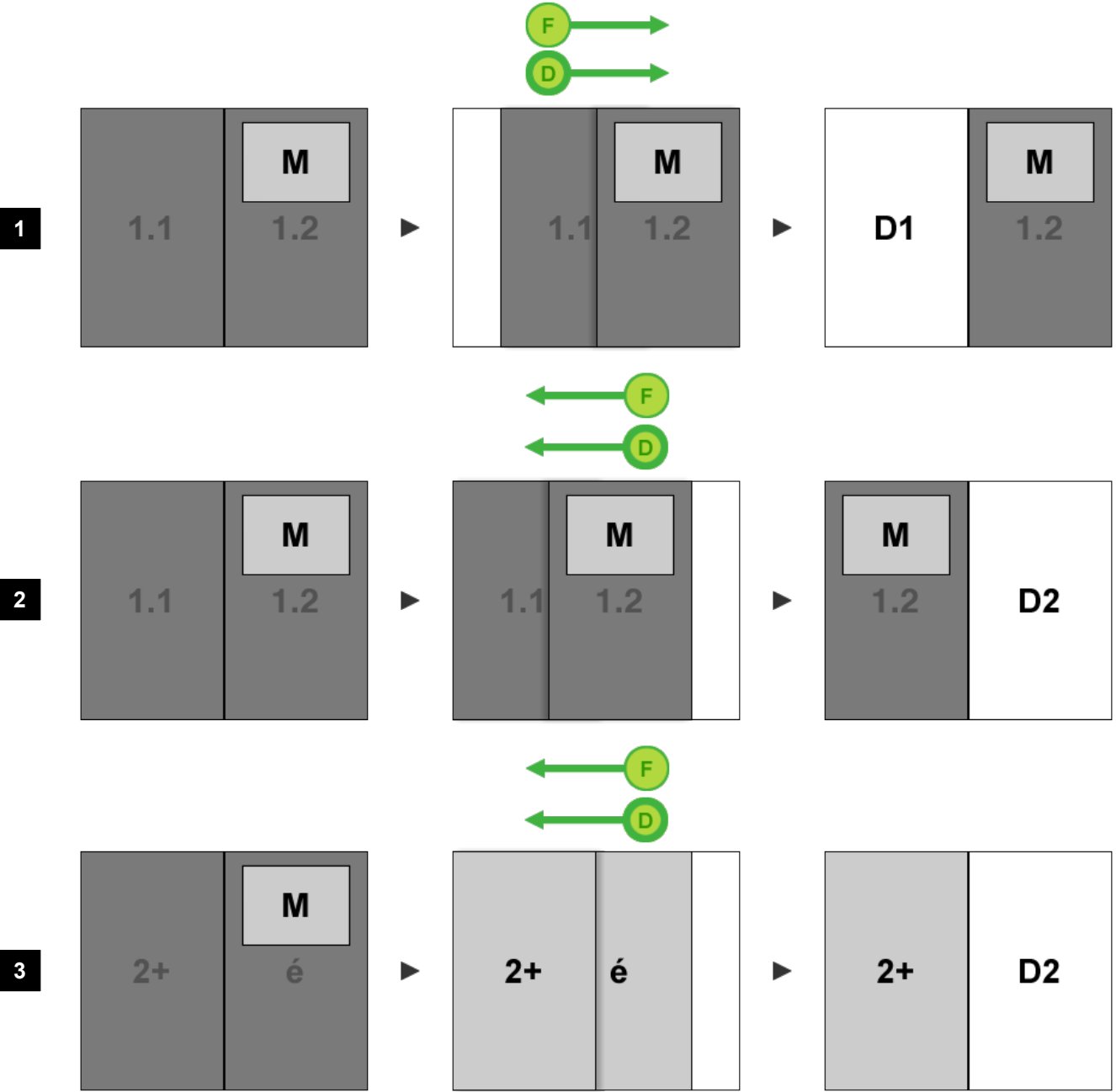
- 1. Minimizing an exposé view to the right will move the parent view to the right.
- 2. Minimizing an exposé view to the left will keep the parent view on the left.

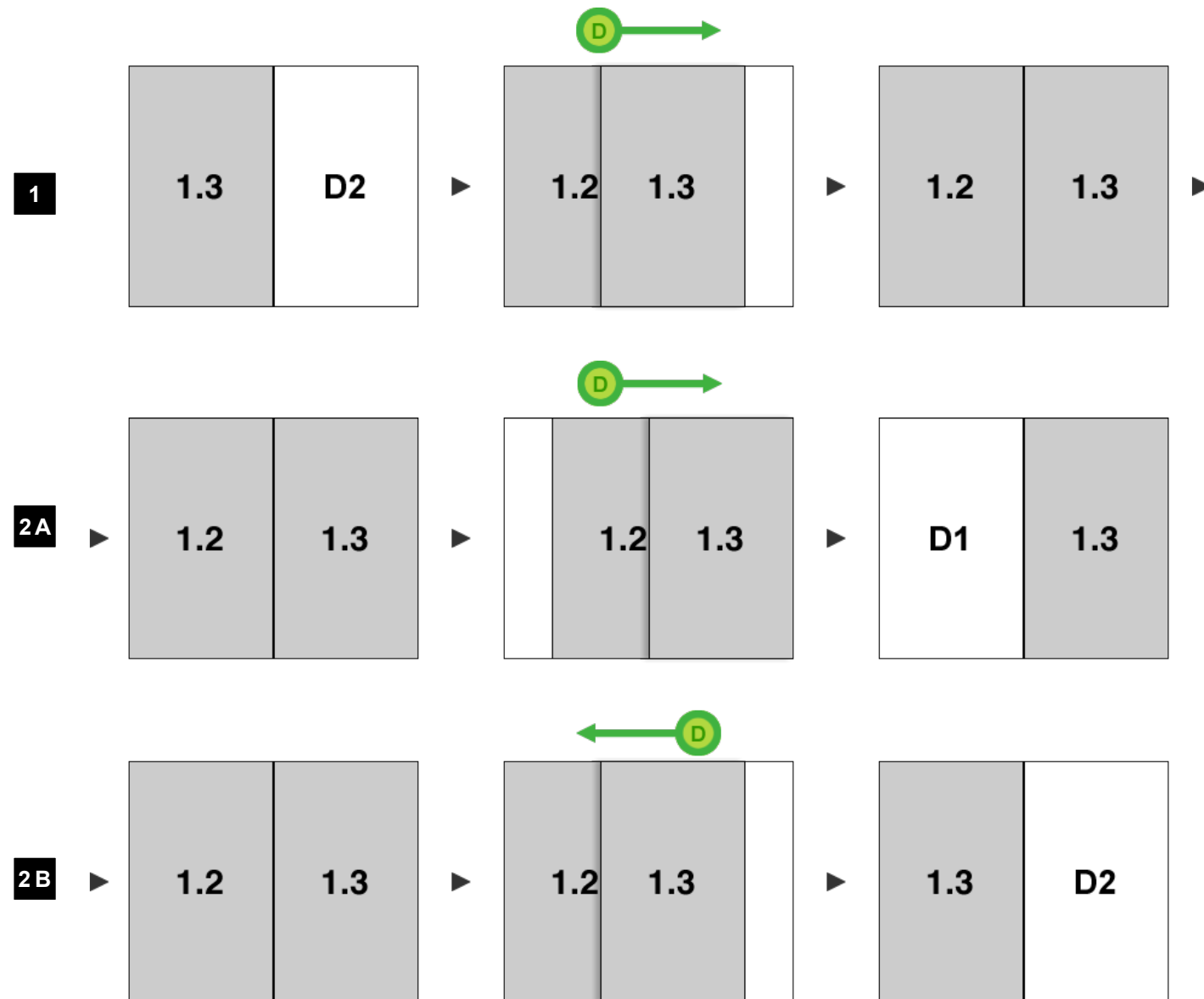


2.8 Modal behavior during minimization

Dual-screen enabled applications can contain modal views on either screen. As a rule of thumb, the view containing the modal will always be visible after a minimization action whenever possible. This rule supersedes any other minimization rules.

1. Minimizing a parent+modal child view from the left will keep the modal view on the right screen and hide the parent view
2. Minimizing a modal parent+child view from the right will move the modal view to the left and hide the parent view
3. If the modal is a view that cannot be minimized to, like an expose view, the modal is cancelled upon minimization.





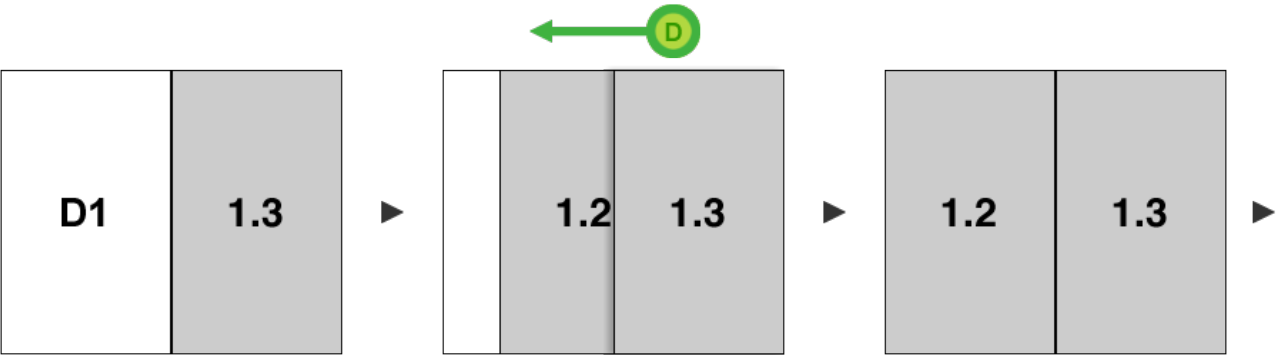
2.9 WPC navigation rules

When applications are being navigated via the off-screen gestures without any interaction with the applications themselves in between the gestures, different maximization and minimization rules apply.

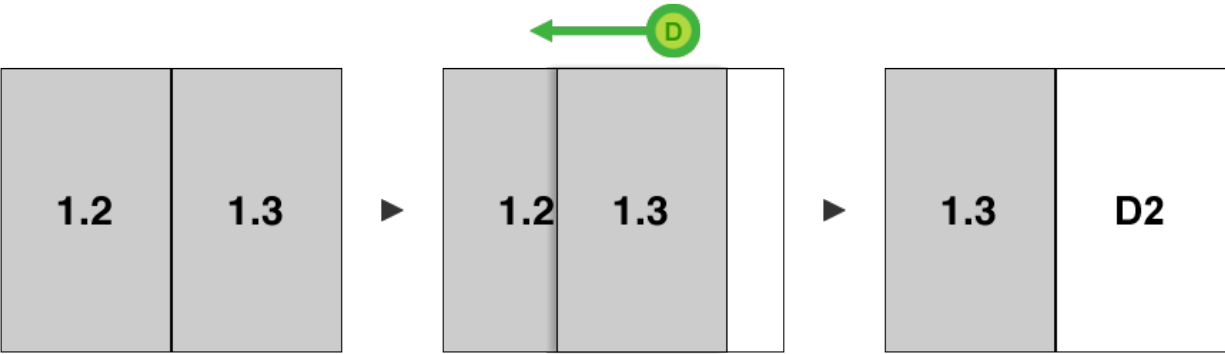
As a rule of thumb, whenever pure WPC navigation is performed, applications retain their original view when maximized and minimized consecutively.

1. Regular maximization rules apply.
2. When a consecutive minimization action is performed without interacting with the application, the original view is retained when minimizing the application.

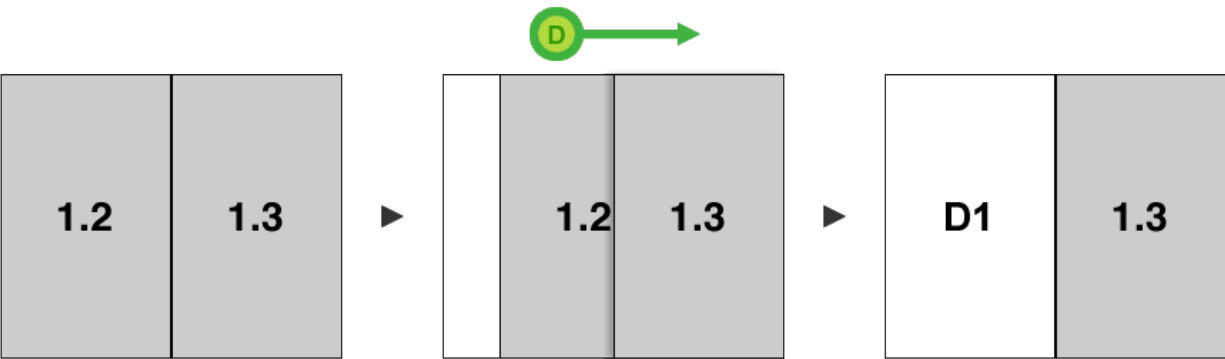
3



4A



4B



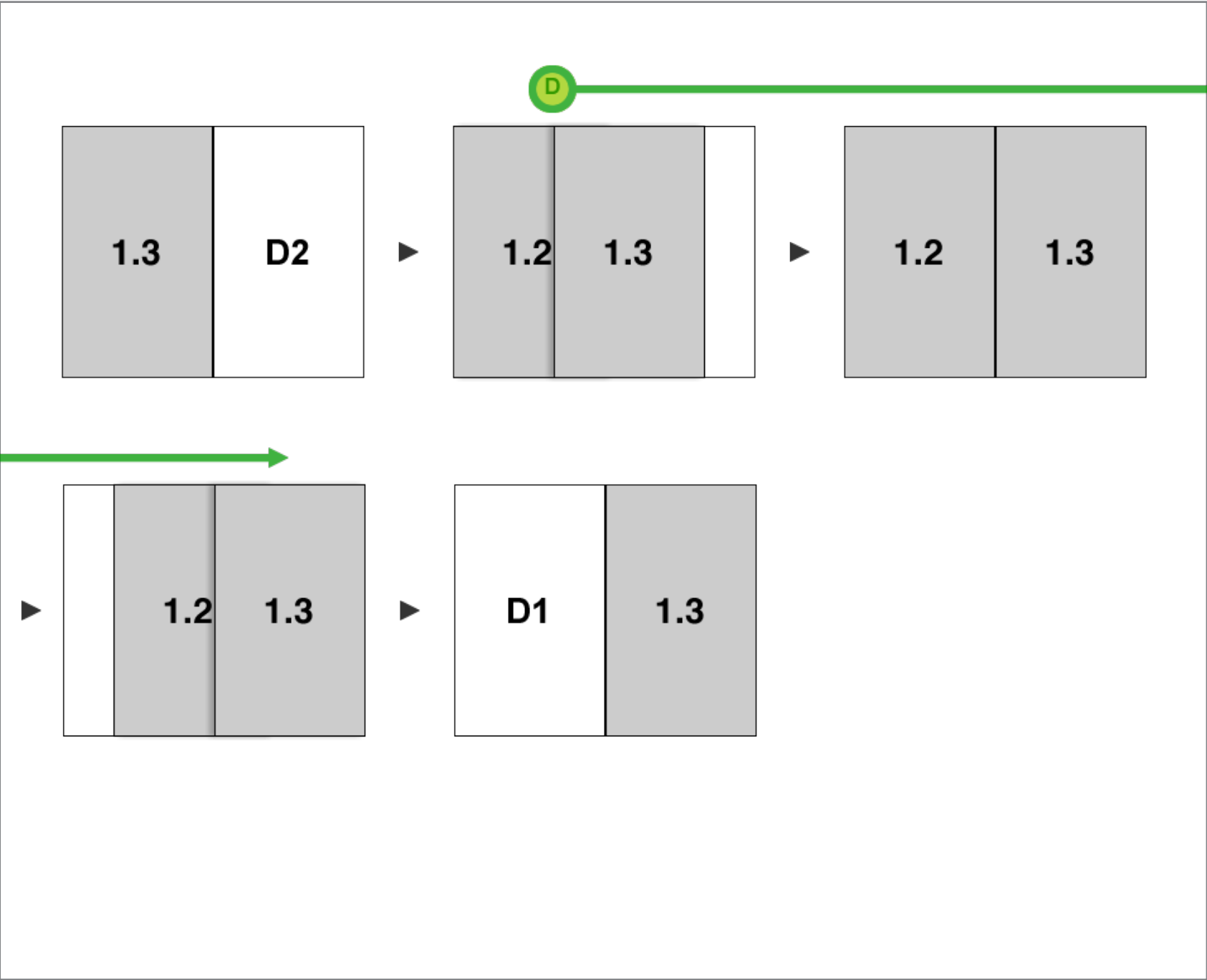
- 3. Regular maximization rules apply.
- 4. When a consecutive minimization action is performed without interacting with the application, the original view is retained when minimizing the application.

2.10 Long-dragging parent-child views

It is possible to perform a long-drag operation in order to first maximize, then minimize an application with one continuous gesture.

Essentially, the long-drag gesture equals several individual drag operations performed sequentially; thus, WPC navigation rules apply.

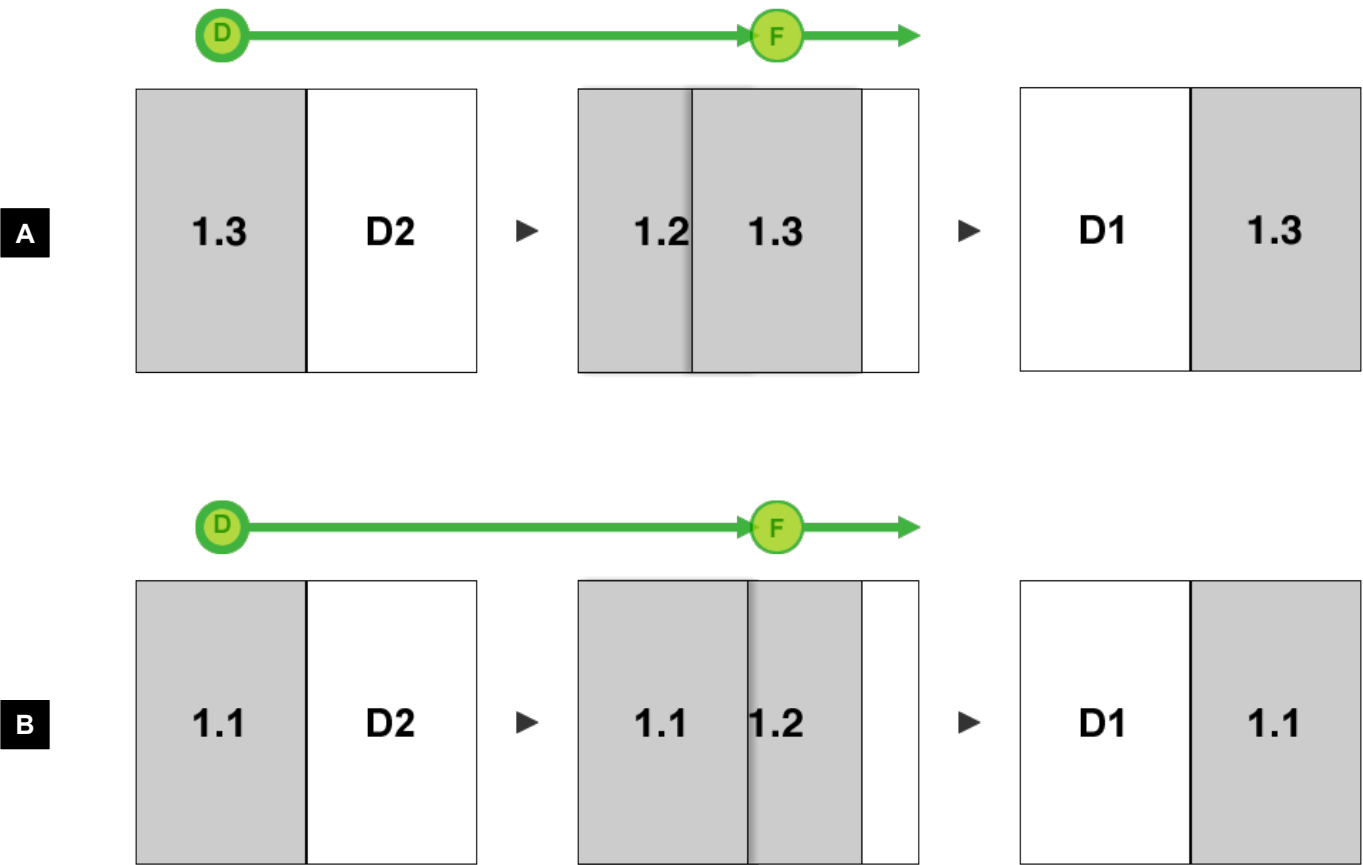
Therefore, using the long-drag, it is possible to move several single and/or dual-screen applications, depending on the length of gesture.

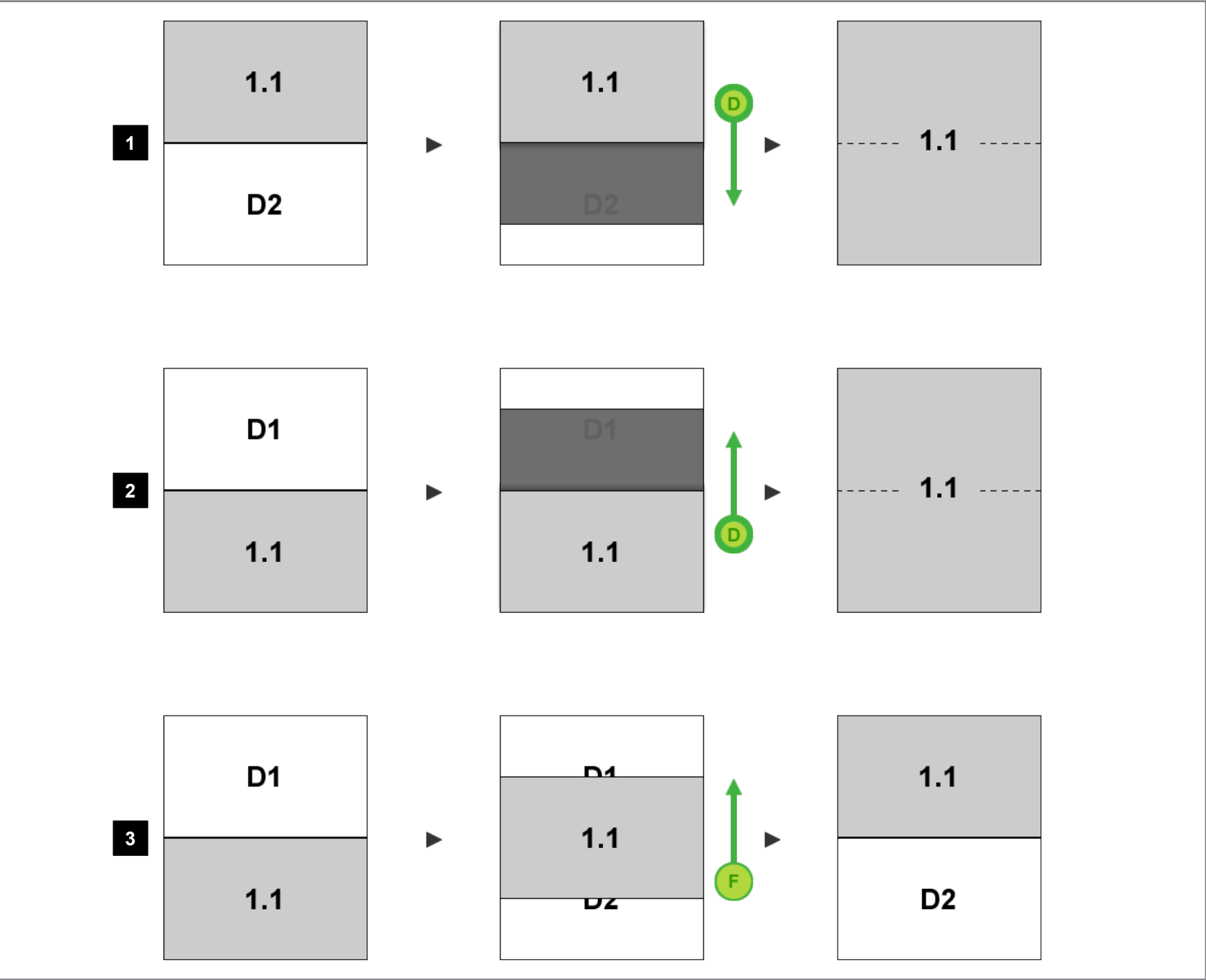


2.11 Drag + flick gestures

At any time during a WPC drag gesture, it is possible to initiate a flick gesture. As long as the drag flick gesture is not yet detected, the gesture is considered a drag and minimized dual-screen applications behave accordingly; as soon as a flick is detected, the ongoing maximization transition is retracted and the view is moved as per standard flick operations.

In the examples on the left, the operation starts as a drag, but half-way through a flick is initiated; the maximization operation is retracted at this point while the view moves to the other screen as per standard flick rules.



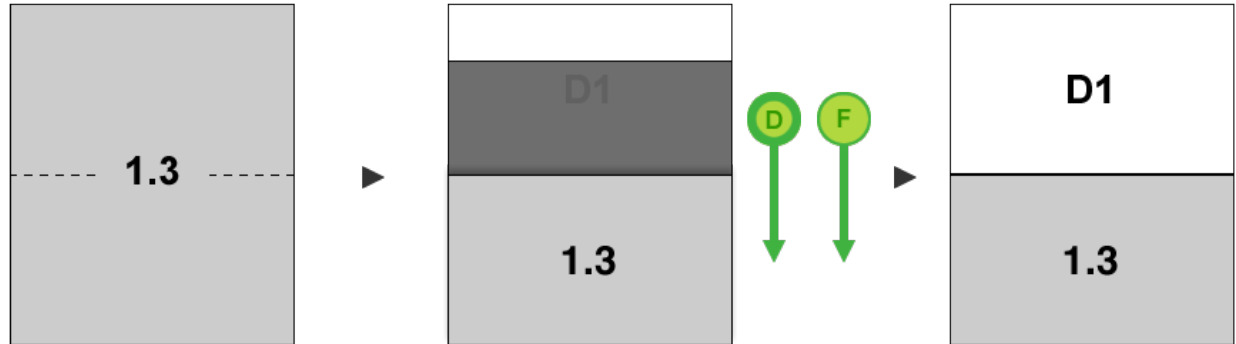


2.12 WPC rules in landscape orientation

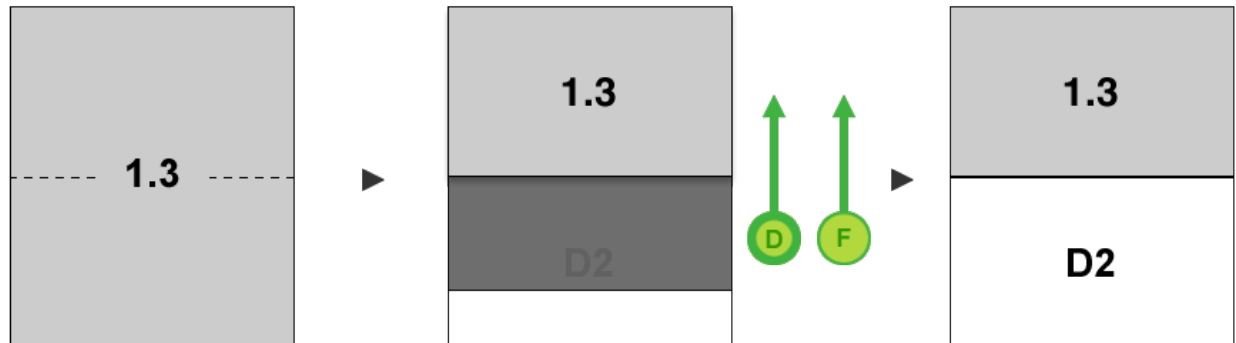
In landscape orientations (LS, LD), the WPC rules apply similarly like in portrait orientations; the difference is that dual-screen applications in LD may present one view, not two, which in turns affects the maximization and minimization rules.

1. Maximization of a parent-child dual-screen application in LD orientation
2. Maximization of a parent-child dual-screen application in LD orientation
3. Flicking behavior in LD orientation for dual-screen applications

4

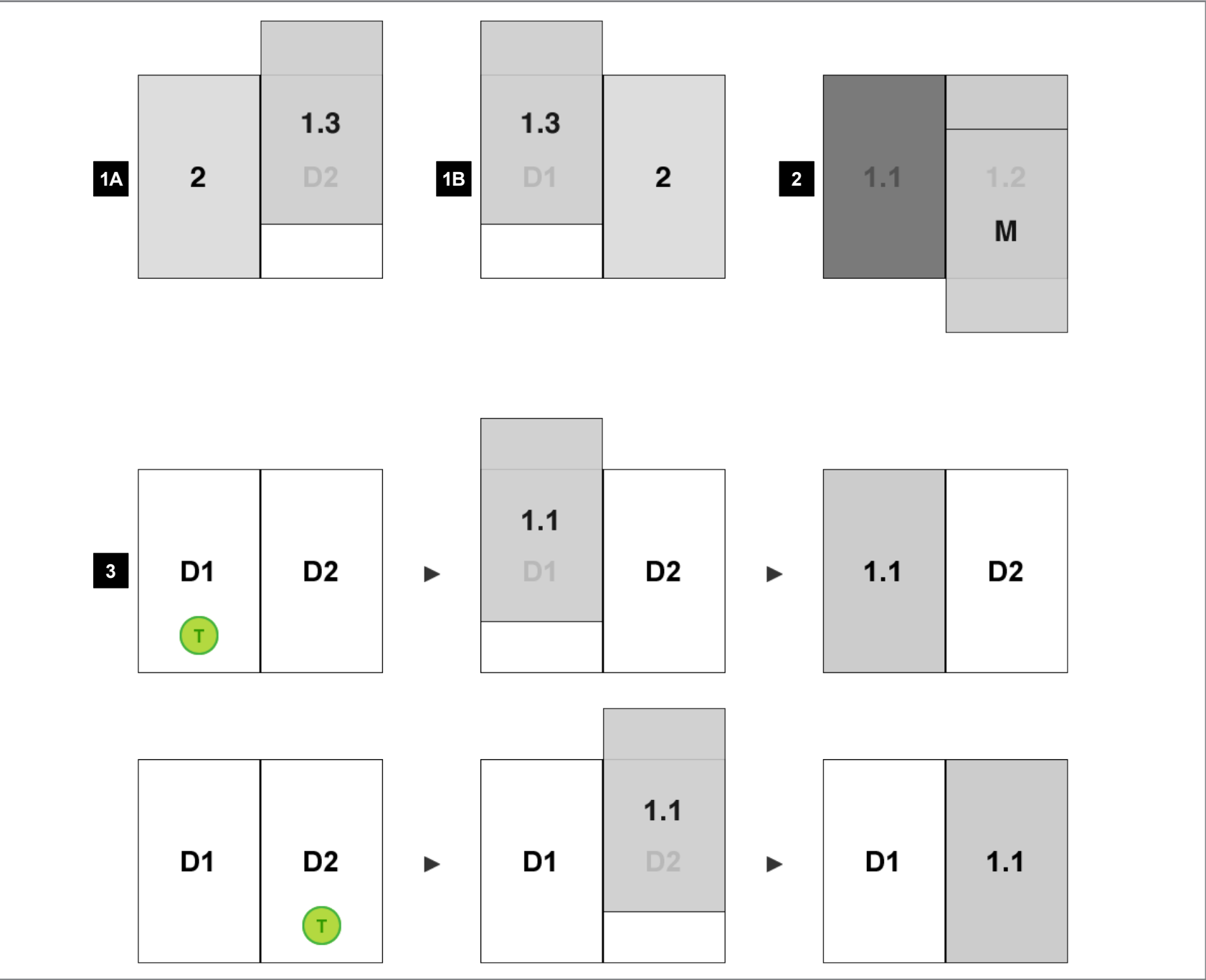


5



- 4. Minimization of a parent-child dual-screen application in LD orientation; note that minimization in LD retains the current view after minimization.
- 5. Minimization of a parent-child dual-screen application in LD orientation; note that minimization in LD retains the current view after minimization.

3 Application launching and quitting



3.1 Introduction to application launching and quitting

When new applications or application instances are launched, they are placed either on the left or the right screen.

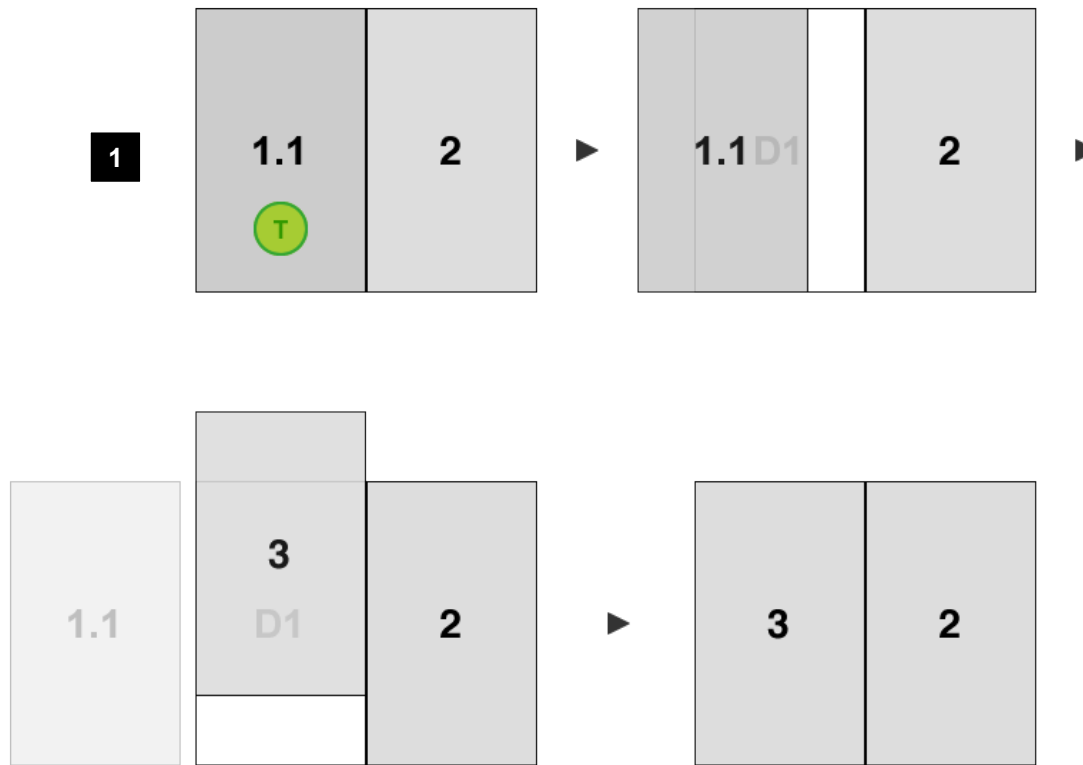
1. Applications can be launched to either screen.
 - Applications remember their last-seen view within their internal structure, ie a parent-child application quit from the leaf view will relaunch with the leaf view.
2. Full-screen modals are launched with a different transition, and dim out the other screen.
3. Dual-screen capable applications are always launched in single-screen mode, on the view they were launched from.
 - If a dual-screen application is quit and then relaunched, the view shown upon relaunch is the parent view of the two views seen upon quitting. Exposé views reemerge with their parent view.

The motion for launching a new application is:
MOT.3.3 Launch App

The motion for quitting an application is:
MOT.3.4 Quit App

The motion for launching full-screen modals is:
MOT.4.7 Full-Screen Modal Appear

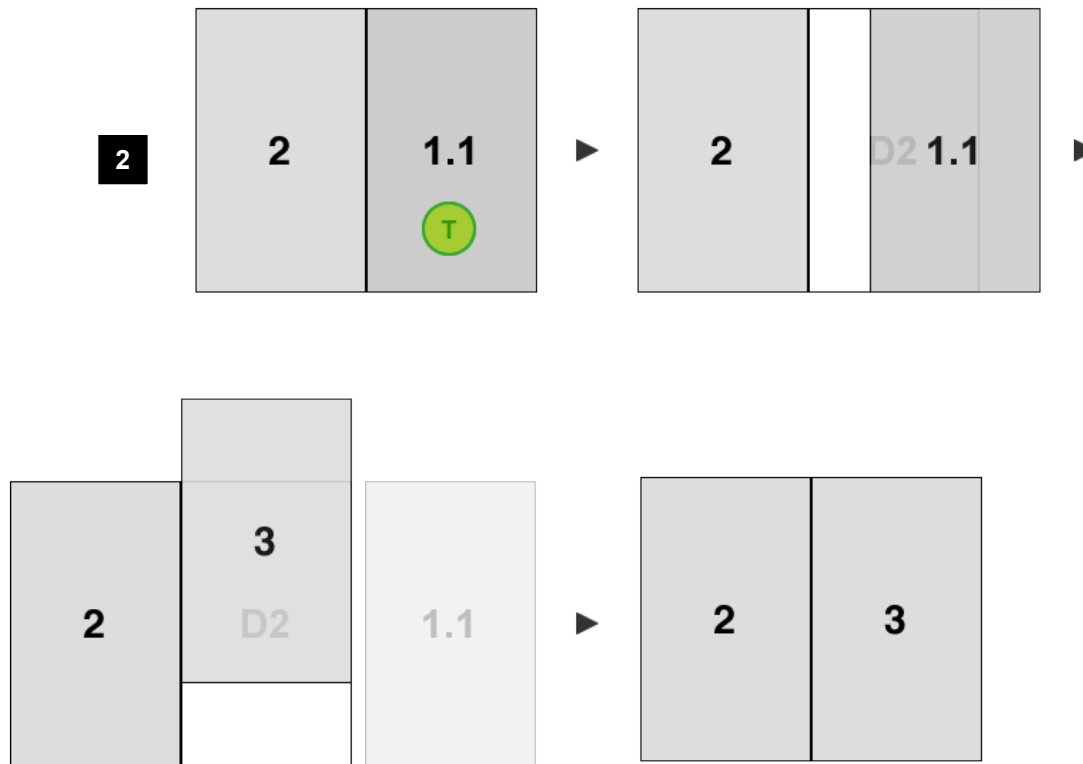
The motion for dismissing full-screen modals is:
MOT.4.8 Full-Screen Modal Disappear



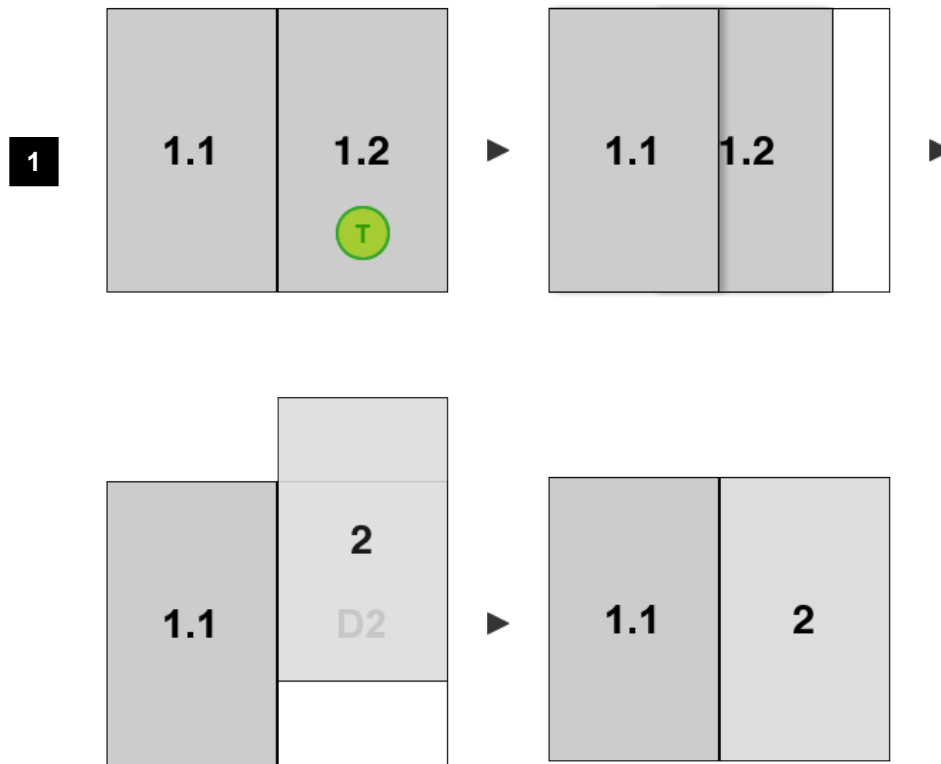
3.2 Launching applications

As a rule of thumb, new applications are always placed on the screen from which they were launched from. If necessary, the new application moves the view that occupied its spot towards the direction of the screen (applications on left move leftward, applications on the right rightward).

1. An new application is launched from the left screen; the application that initiated the launch is moved out of view towards the left, and the new application is placed on the left screen.



2. An new application is launched from the right screen; the application that initiated the launch is moved out of view towards the right, and the new application is placed on the right screen.



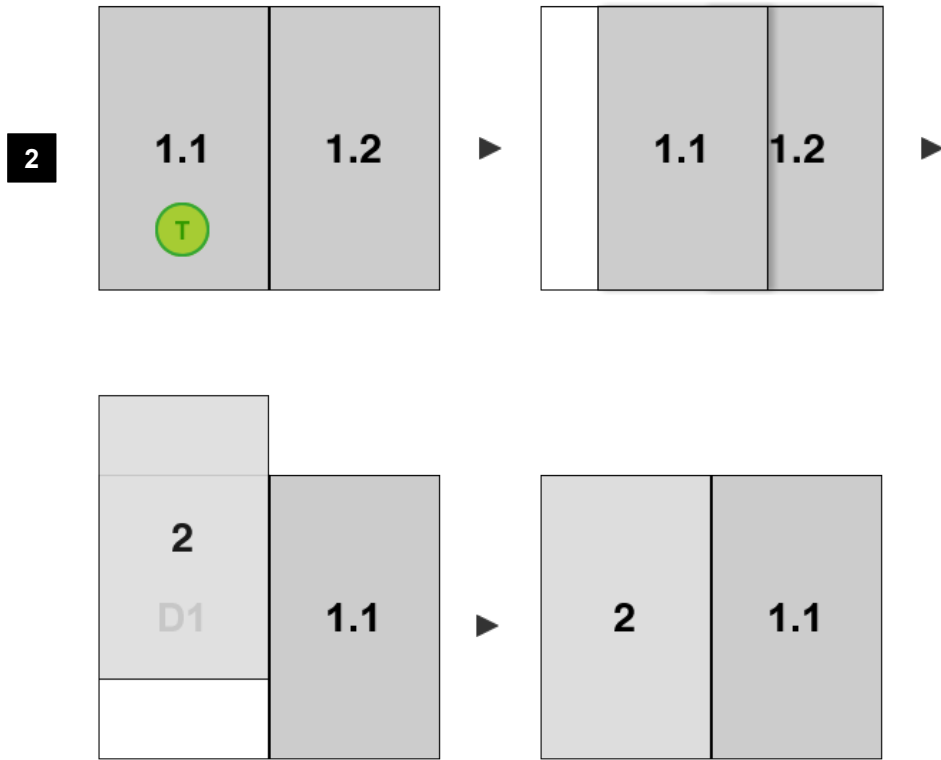
3.3 Auto-minimization during application launch

RULE SUGGESTION: PRESERVE VIEW THAT INITIATED APP LAUNCH

When new application windows are launched, dual-screen applications auto-minimize in order to make room for the newly launched window.

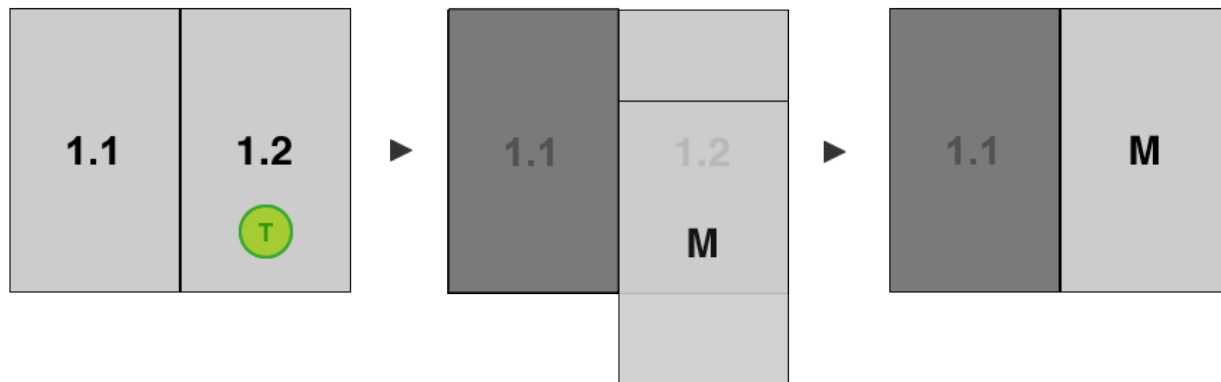
Because of the rules of application launching, the dual-screen application is minimized in such a manner that it occupies the opposite screen of where the new application is placed into.

1. A new application is launched onto the right screen; the dual-screen application minimizes to the left according to standard minimization rules, and the new application is brought into the left screen.

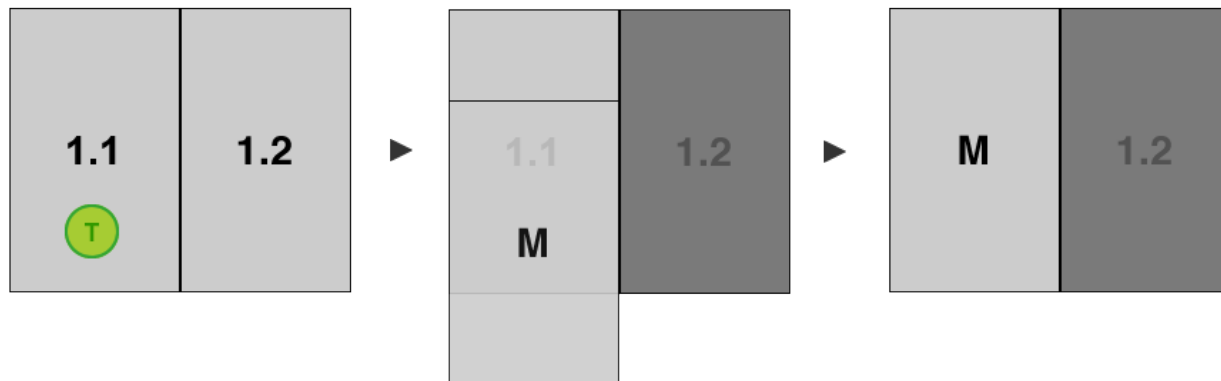


2. A new application is launched onto the left screen; the dual-screen application minimizes to the right according to standard minimization rules, and the new application is brought into the right screen

3A



3B



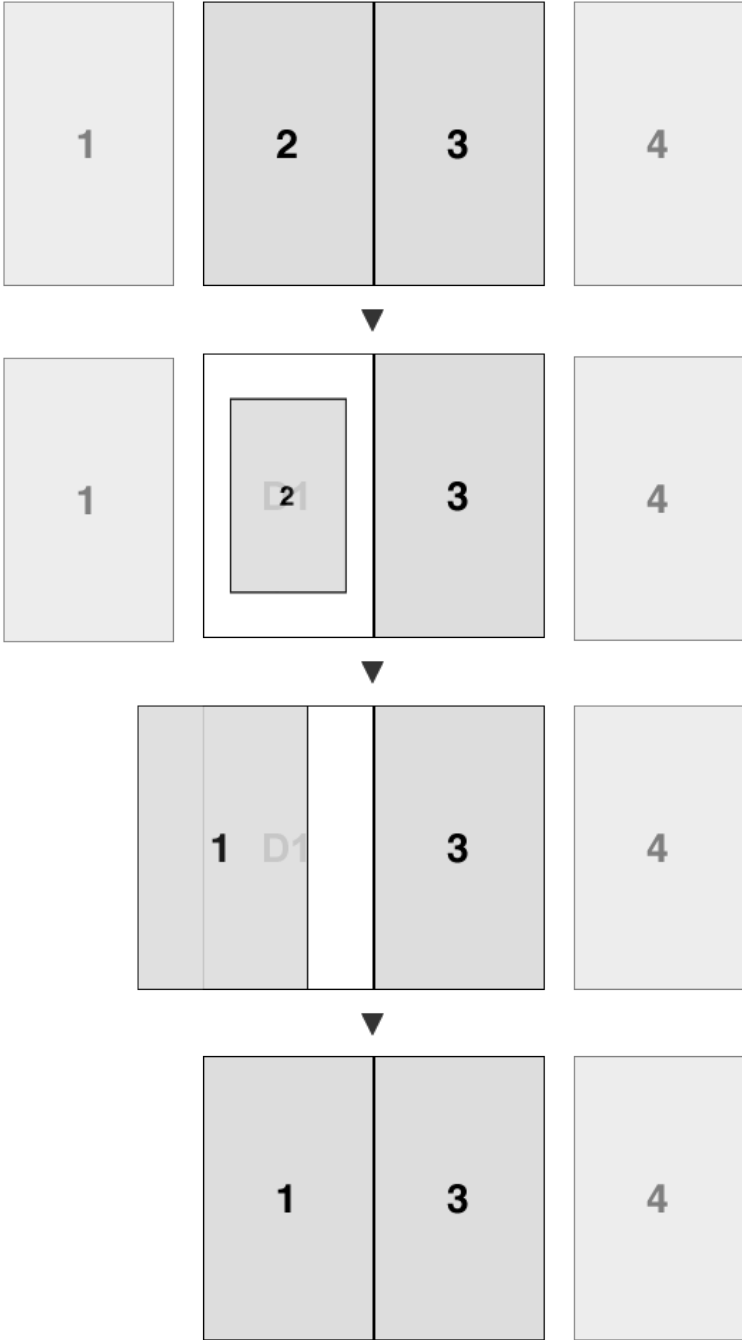
3. A full-screen modal is summoned from within the application; the modal appears on the side it was launched from, on top of the view, dimming out the other screen.

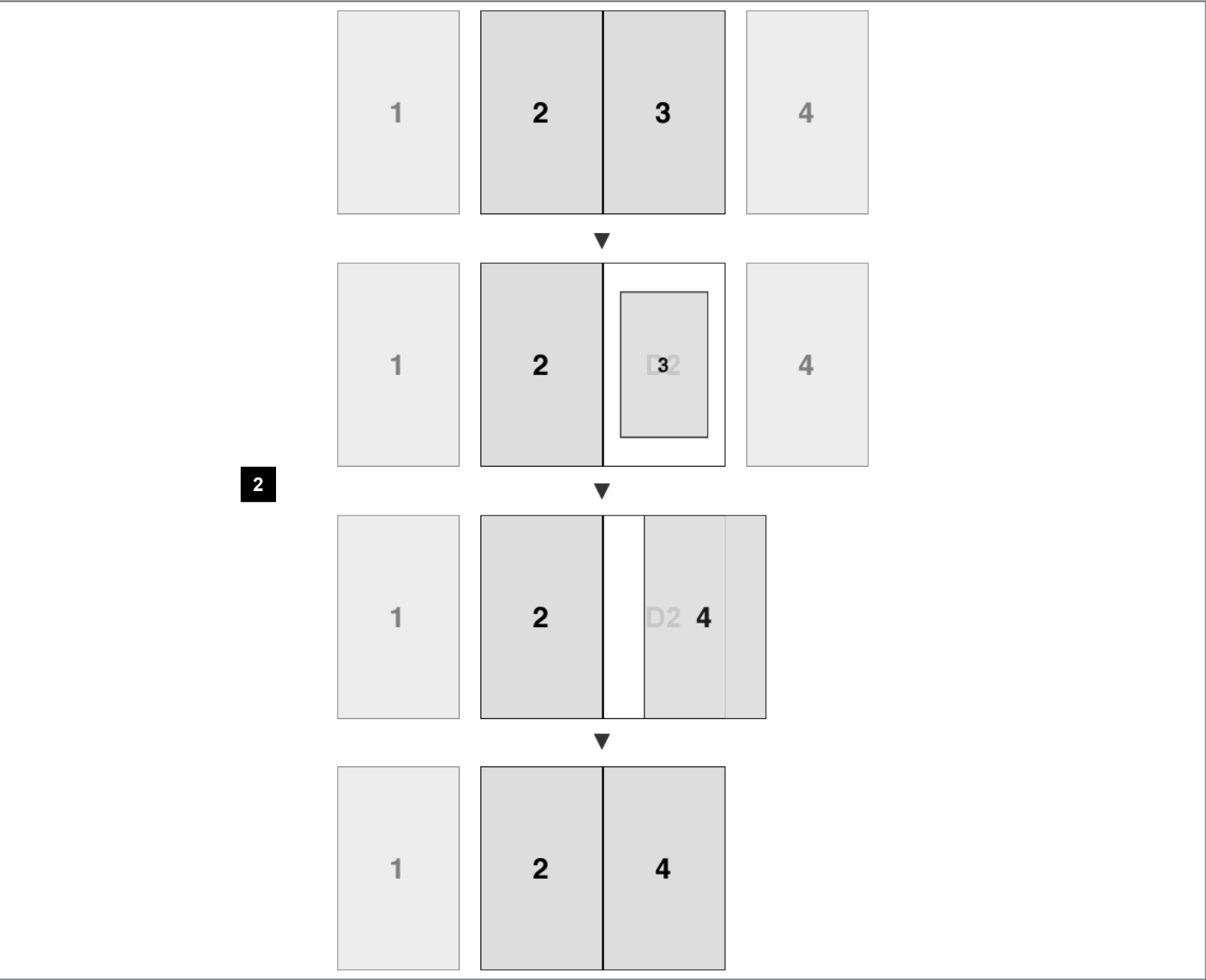
3.4 Quitting applications

As a rule of thumb, when applications are quit the gap they leave behind in the stack is filled with the next application on the same screen.

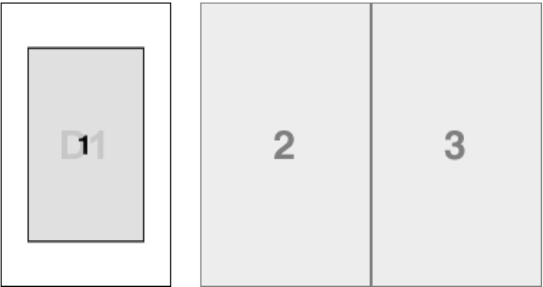
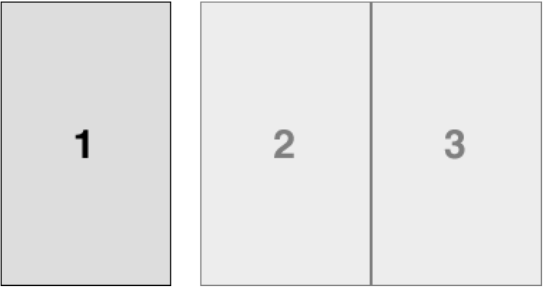
1. The application on the left screen is closed
 - The closed application disappears
 - The gap created by the closed application is filled from the left; If there is an application located to the left of the closed application, it slides into view
 - If there is no application to the left of the closed application, the underlying desktop is shown.

1

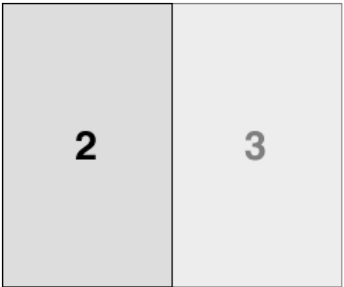
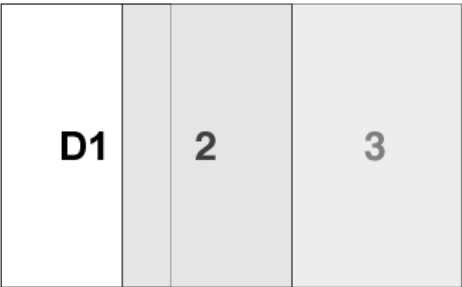




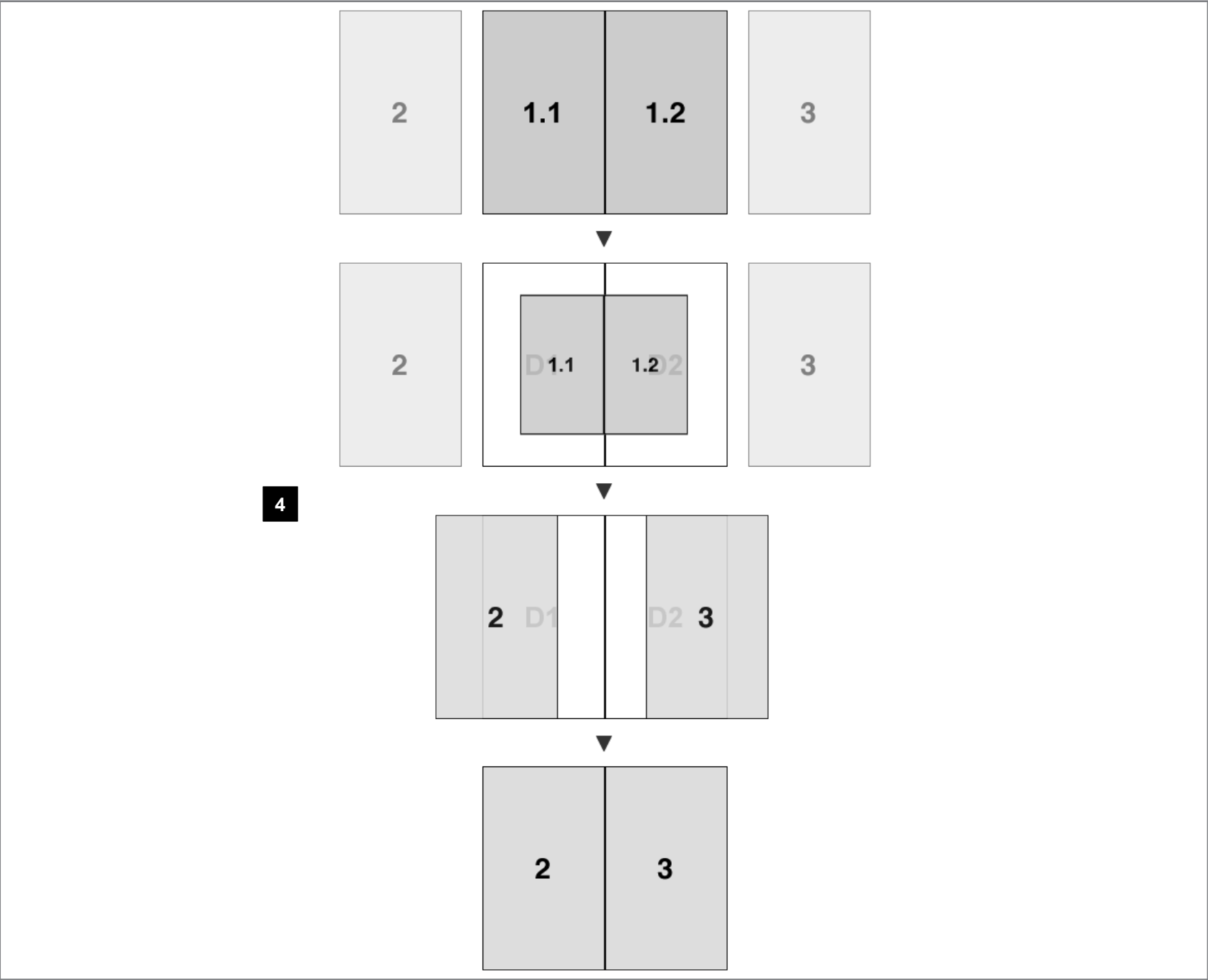
2. The application on the right screen is closed
- The closed application disappears
 - The gap created by the closed application is filled from the right; If there is an application located to the right of the closed application, it slides into view
 - If there is no application to the right of the closed application, the underlying desktop is shown.



3



3. If the first application in the stack is quit when the device is closed, the remaining gap is filled from the right.

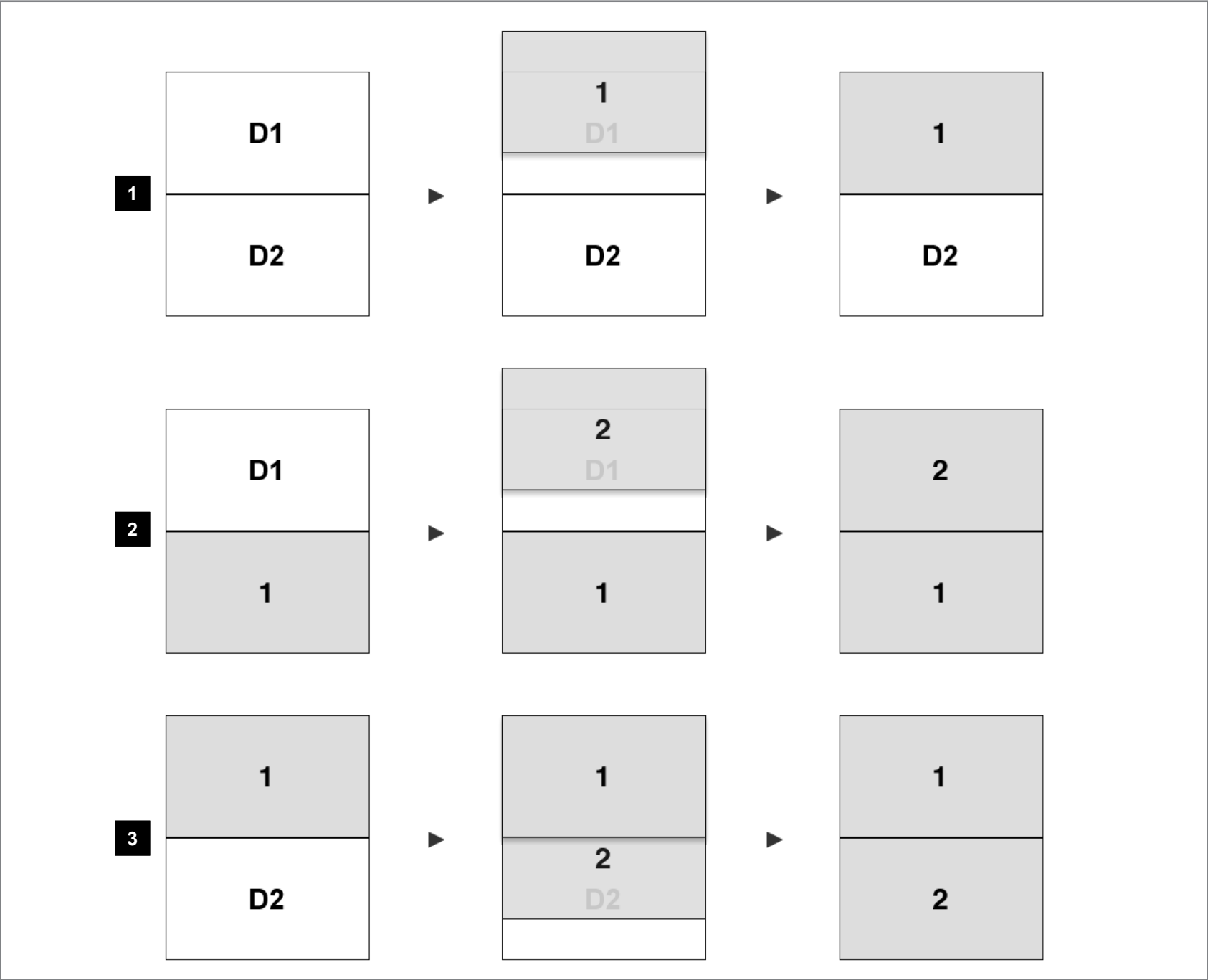


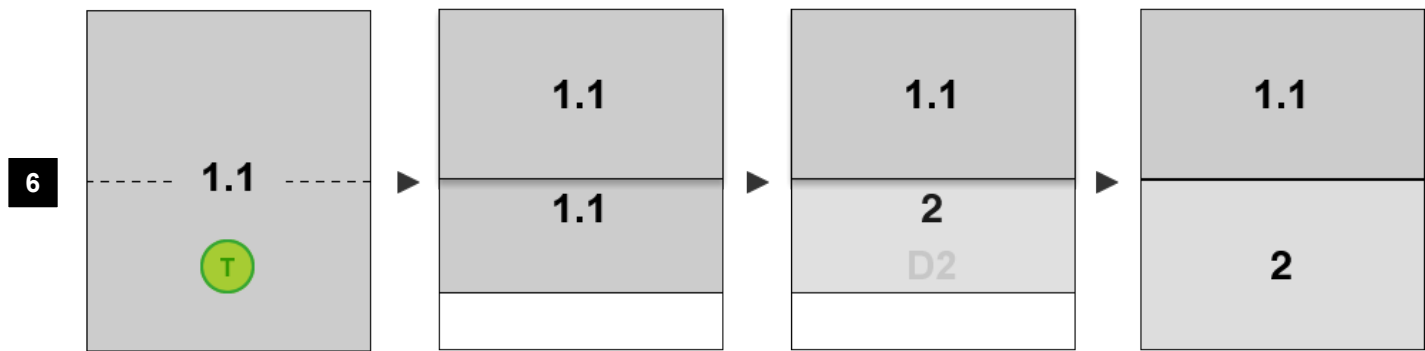
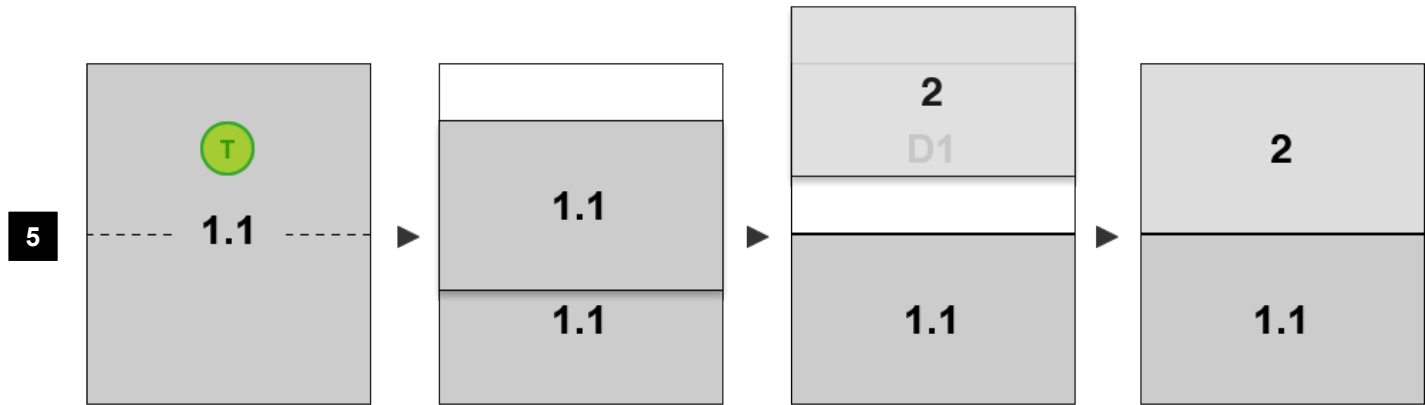
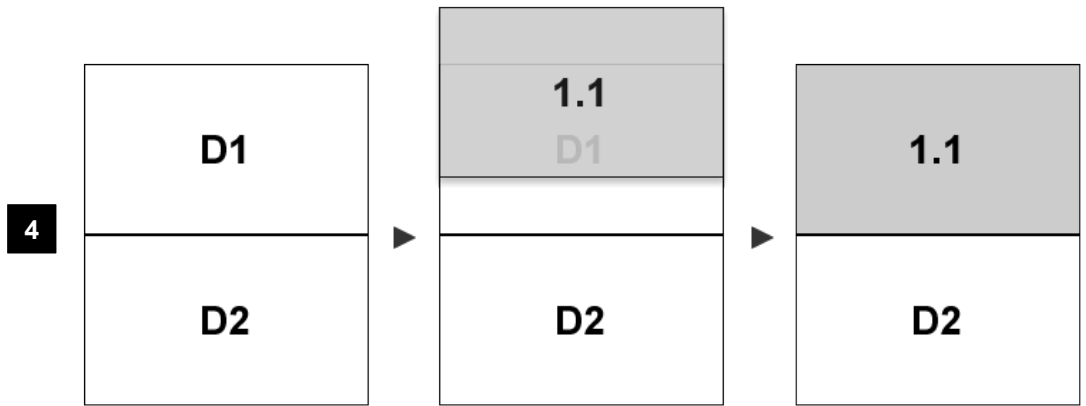
4. Quitting a maximized dual-screen application; the remaining gaps are filled (when possible) from both sides respectively.

3.5 Launching and quitting applications in landscape orientation

Launching and quitting applications in landscape orientation is very similar to portrait orientation.

- 1. The top screen is considered the primary screen, and applications launch there by default if the complete desktop is revealed.
- 2. An application is launched on the top screen.
- 3. An application is launched on the bottom screen.





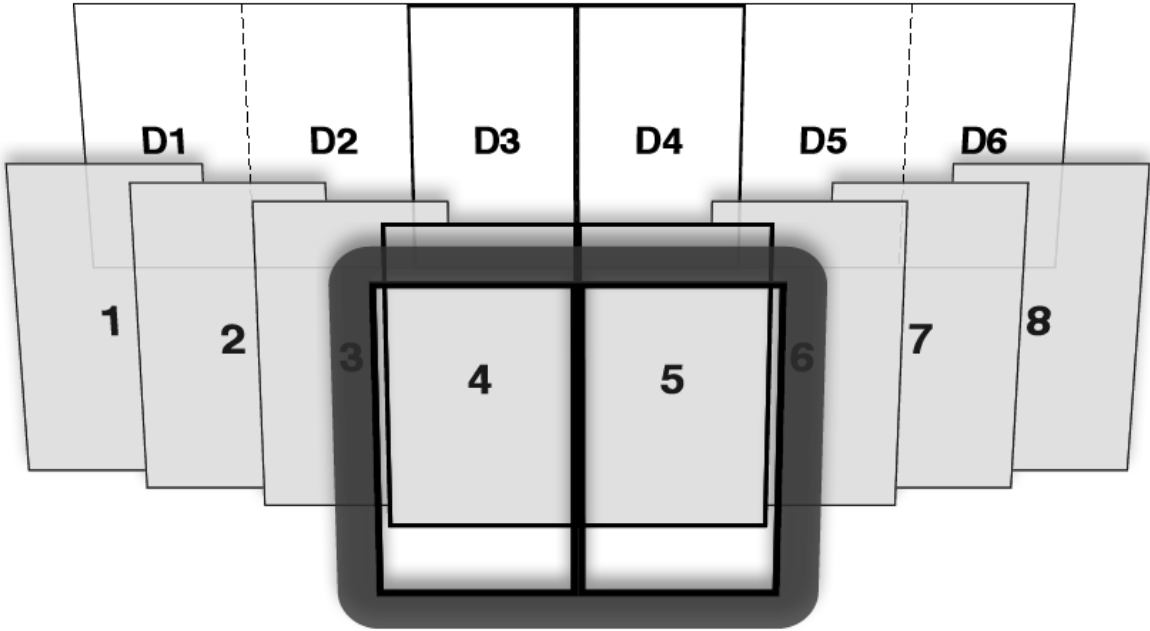
4. A dual-screen application is launched in collapsed form. Gravity drop is not applied automatically.
5. A new application is launched from the top screen within a maximized dual-screen application; the dual-screen application is minimized, and the new application launched on the screen that it was initiated from.
6. A new application is launched from the bottom screen within a maximized dual-screen application.

4 Window stack & opening and closing the device

4.1 Introduction to the window stack

Conceptually, application windows are represented in two stacks, one on each screen of the device. The Desktop lies underneath the window stack, as a large canvas divided into six sections of which two can be displayed on the screens. This is visualized in the example on the left.

When the device is closed, the conceptual model persists. Only one stack can be seen (the one on the primary screen, left), but the other still exists on the right; it is simply not shown because the second screen is not available.

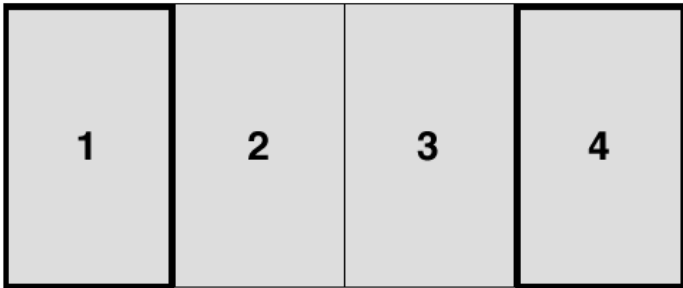


4.2 Navigating the window stack

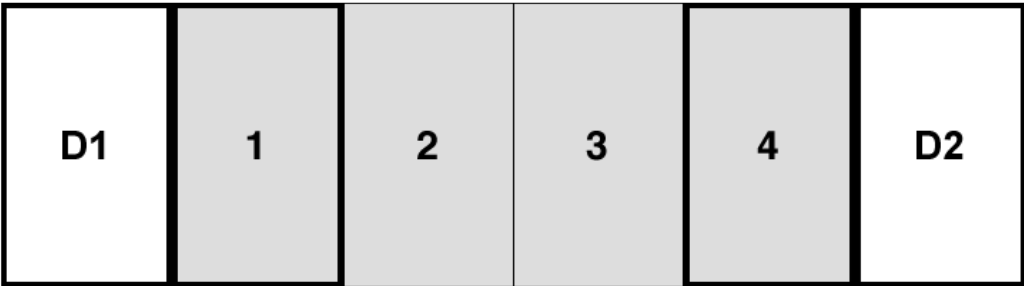
Using WPC, the window stack can be navigated by dragging applications from the left screen to the right and vice versa.

1. Device closed. Using WPC, the two ends of the window stack can be reached.
2. Device open. Using WPC, the two ends of the window stack can be reached, with a desktop shown on the second screen respectively.
3. A dual-screen application within the window stack. Dual-screen applications can only be maximized when in view.

1



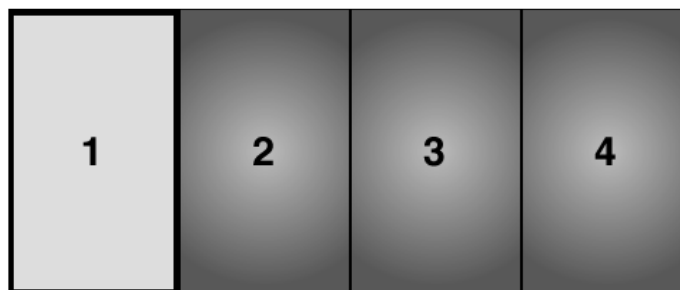
2



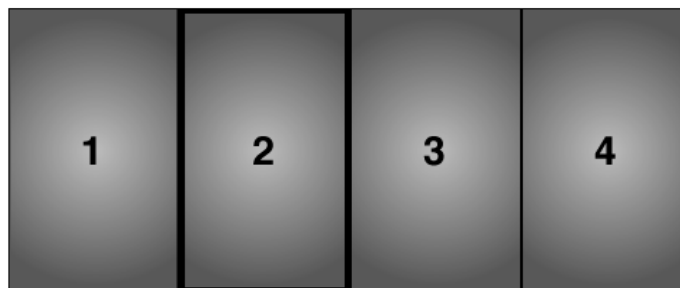
3



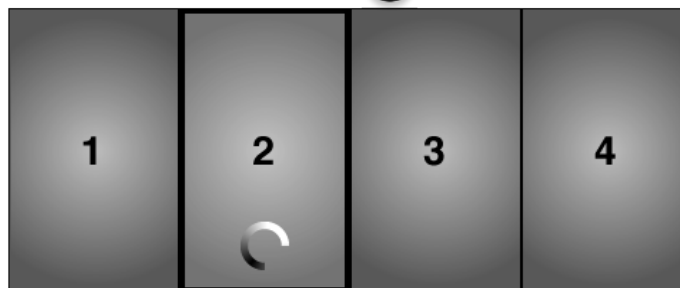
1



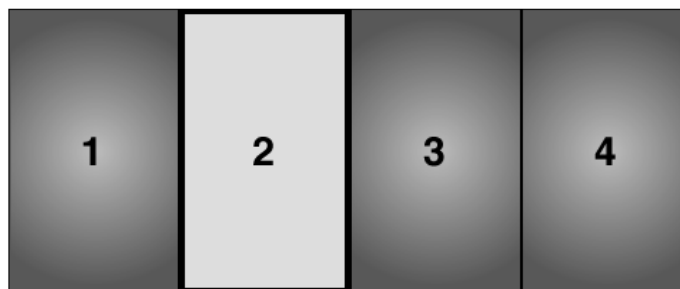
2



3



4



4.2.1 Application sleep and wake

Due to performance considerations, applications that are not in view are put to a sleep state. This means that when WPC is used to navigate along the window stack, applications that come into view must be awoken before they can be used again.

1. Applications that are out of view are put in a sleep state.
2. When the WPC is used to navigate to another application in the stack, the application that comes into view is still asleep; its state is represented with a dimmed-out screenshot and possibly an icon/label. The application previously in view is now put asleep.
3. Once an application that is asleep comes into view, a timer sets off to trigger the awakening call for the application. This means that if WPC navigation operations continue without a delay, the application in view is not woken up. In this example, however, the timer runs out and the application is awoken: an indicator is shown to reflect this.
4. Once the application is awake, it is presented in view.

4.3 Revealing the Desktop

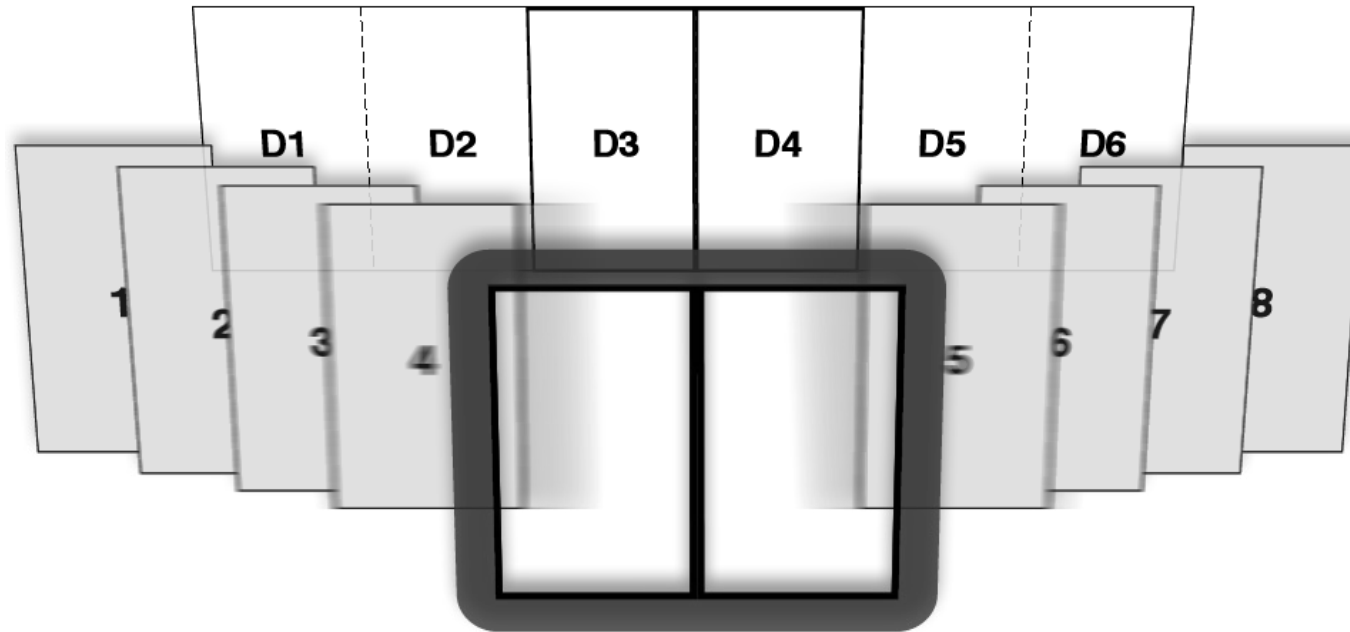
The Desktop can be reached by a *reveal* action. Conceptually, the reveal pushes the application stack out of the way (by moving the left stack out of view to the left, and the right stack out of view to the right).

The reveal is a temporary state; as a rule of thumb, whenever the user is not interacting with the Desktops directly, the reveal is cancelled. This includes WPC operations.

The reveal is initiated by pressing the hardware home-key.

The motion for the desktop reveal is:

MOT.1.5 Reveal

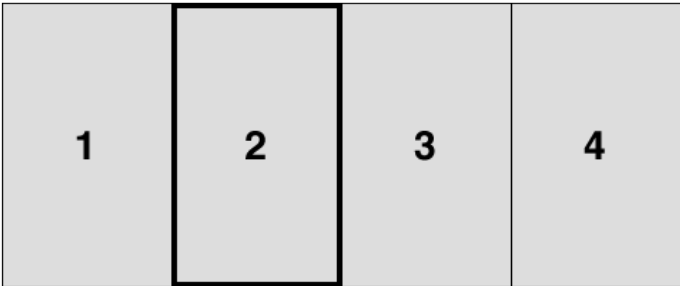


4.3.1 Revealing the Desktop when the device is closed

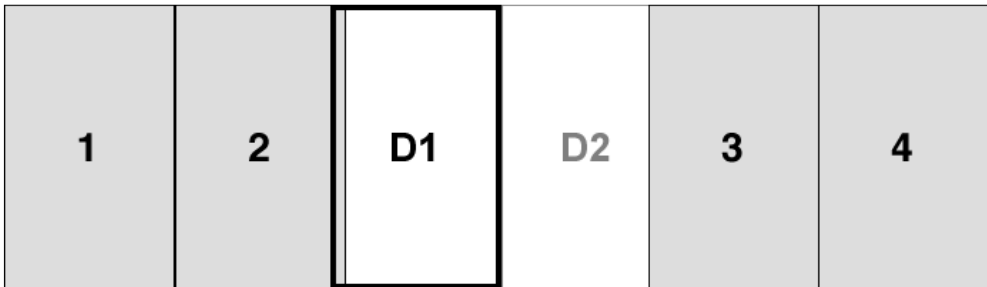
The Desktop reveal can be initiated when the device is closed.

1. Applications 1 and 2 are on the left stack, with application 2 in view; applications 3 and 4 are on the right stack (out of view)
2. Reveal is initiated; the left stack moves out of view to the left. The right stack is pushed towards the right (this would only be visible if the device was suddenly opened while the reveal was initiated).
3. The Desktops can now be interacted with; it is possible to change the visible desktop by dragging gestures.

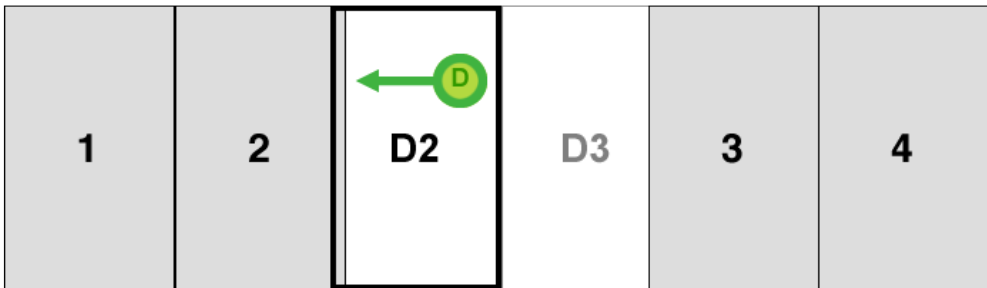
1



2



3

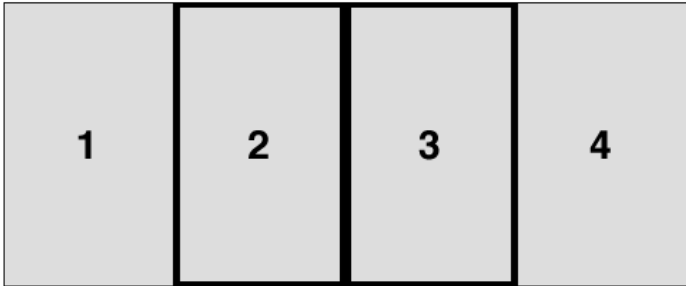


4.3.2 Revealing the Desktop when the device is opened

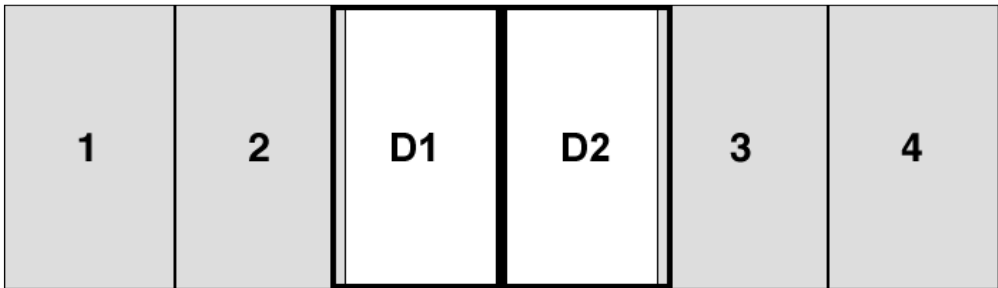
The Desktops can be revealed when the device is open.

1. Applications 1 and 2 are on the left stack, with application 2 in view; applications 3 and 4 are on the right stack, with application 3 in view.
2. Reveal is initiated; the left stack is pushed towards the left, the right stack towards the right.
3. The Desktops can now be interacted with. In this example, the Desktops are navigated by an on-screen drag gesture.

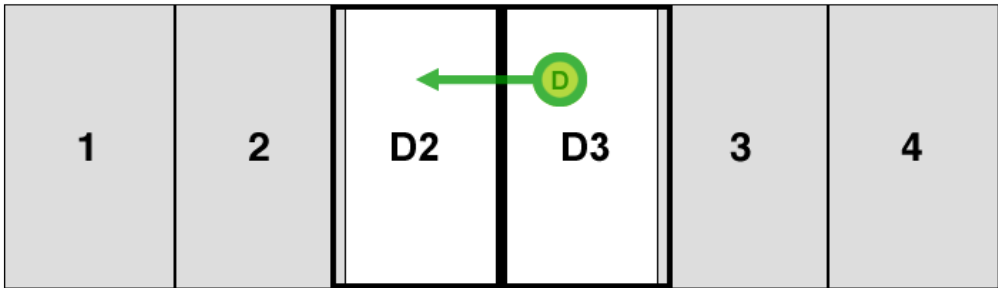
1



2



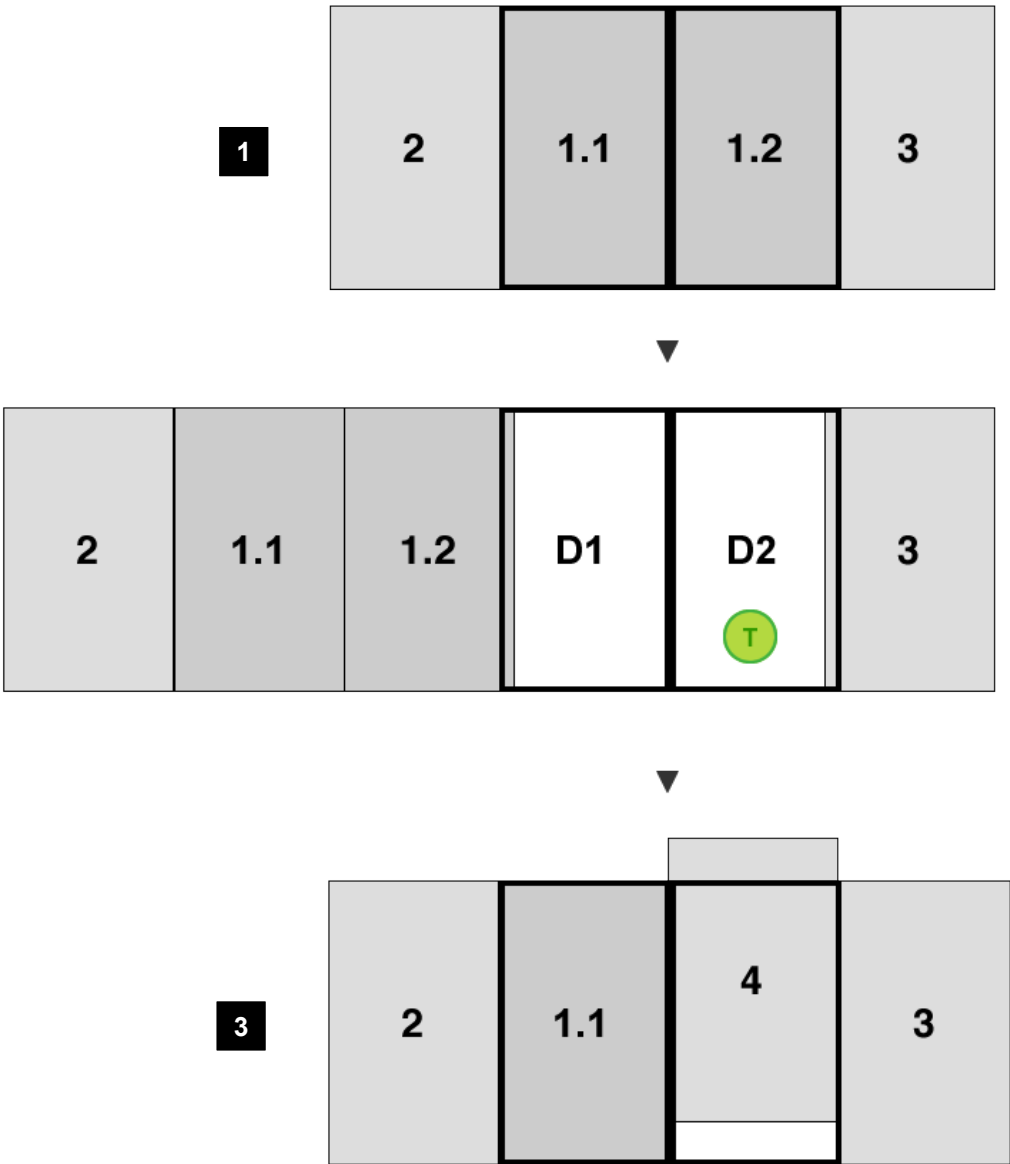
3



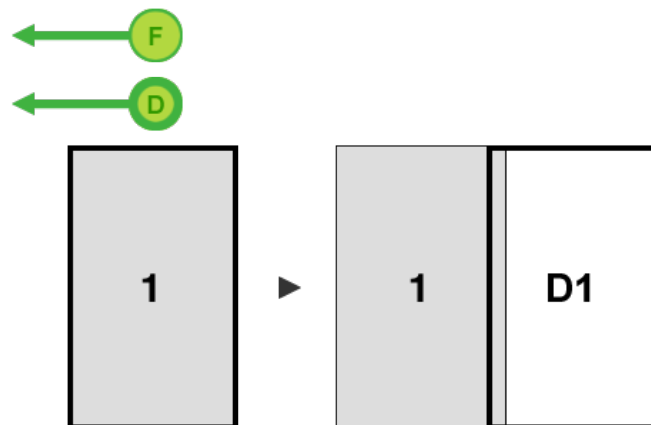
4.3.3 Dual-screen applications and application launching during reveal

Maximized dual-screen applications are can also be pushed aside by the reveal action. Their state is not affected unless a new application is launched.

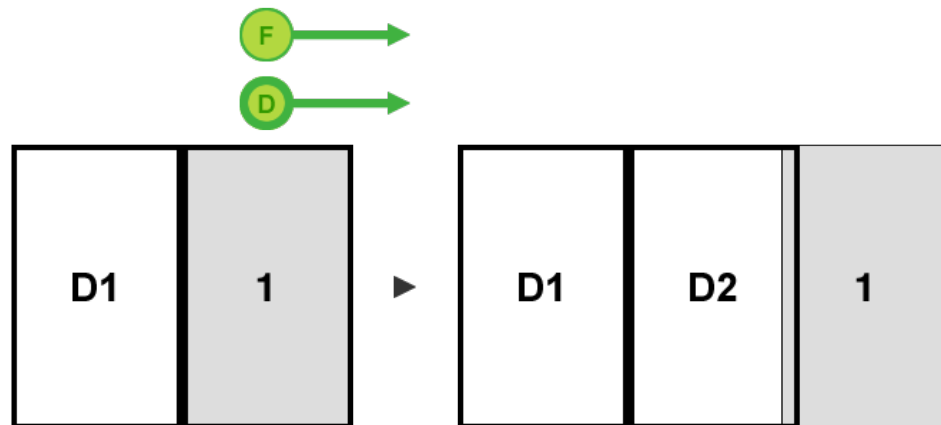
1. A maximized dual-screen application is in view.
2. Reveal is initiated; dual-screen applications are moved towards the primary screen (left).
 - A new application is launched from Desktop 2 (right screen)
3. The newly launched application (number 4) comes into view.
 - The reveal is cancelled.
 - The new application is displayed on the screen it was launched from.
 - The dual-screen application is minimized according to standard minimization rules.



1



2



4.3.4 Revealing the desktop after reaching the end of the window stack

It is possible to reveal the desktop(s) when the end of the window stack is reached on either side of the screen. This is accomplished by a drag or flick operation directed beyond the end of the window stack.

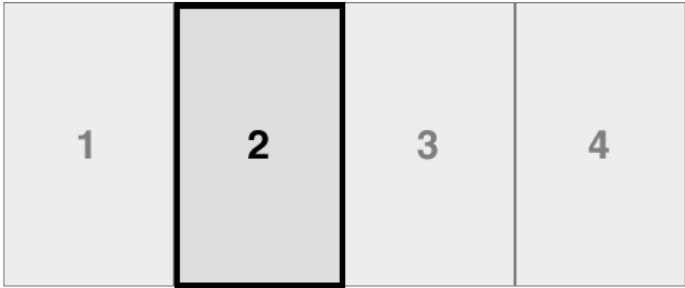
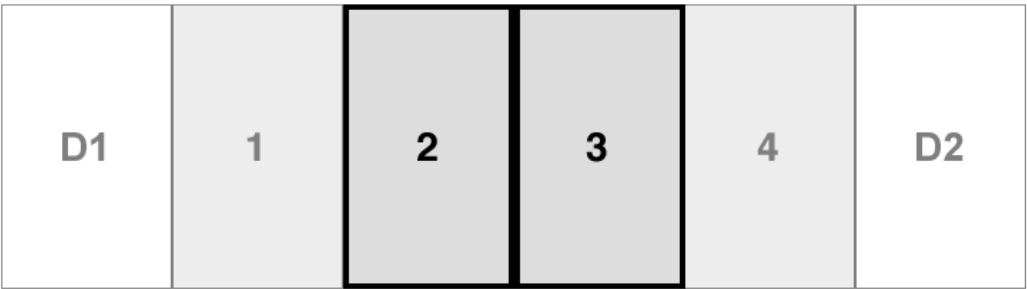
1. Device closed; the desktop is revealed when the first or last application in the stack is dragged or flicked to the right or left, respectively.
2. Device open; both desktops can be revealed by dragging or flicking the first or last application in the stack to the right or left, respectively.

4.4 Window stack behavior during device opening and closing

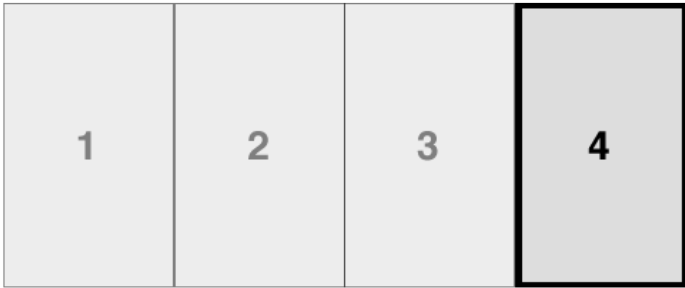
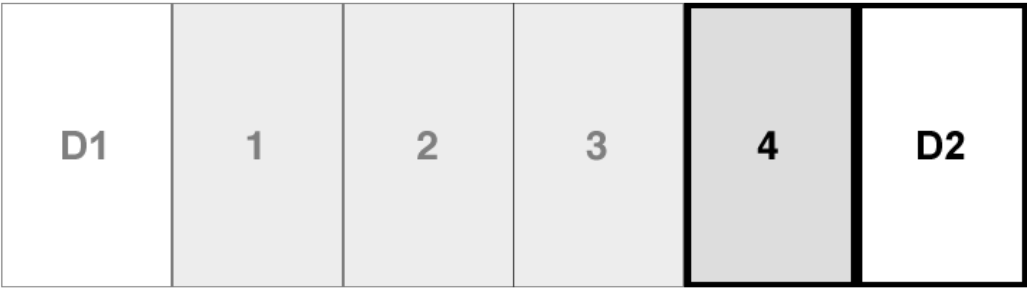
When the device is opened and closed, the window stacking order is preserved whenever possible. The left screen of an open device is considered to be the sole screen of a closed device.

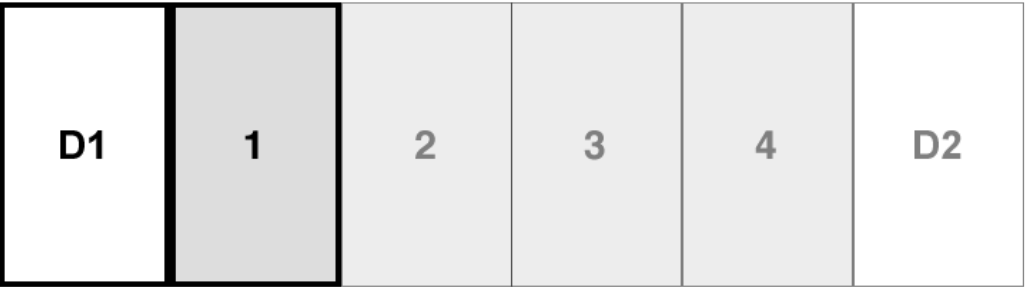
- 1. The current left screen is preserved in its location whenever the device is opened and closed.
- 2. The current left screen is preserved in its location whenever the device is opened and closed.

1

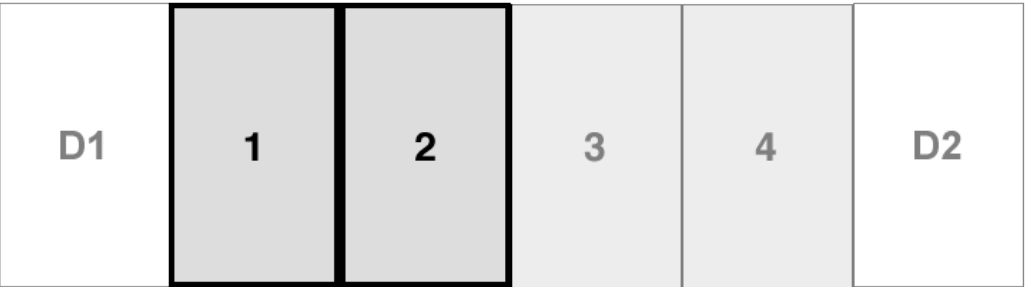


2





3



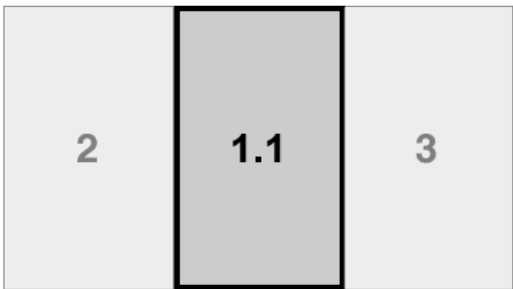
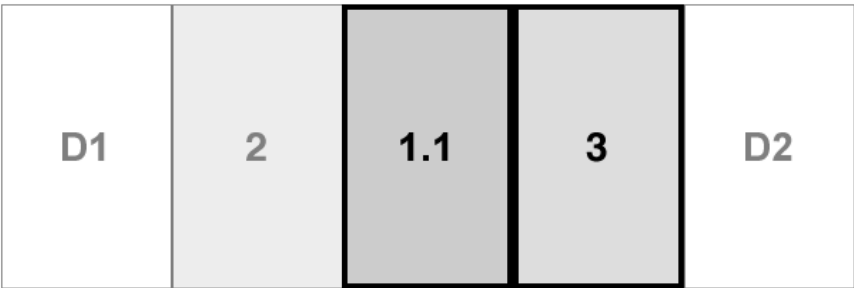
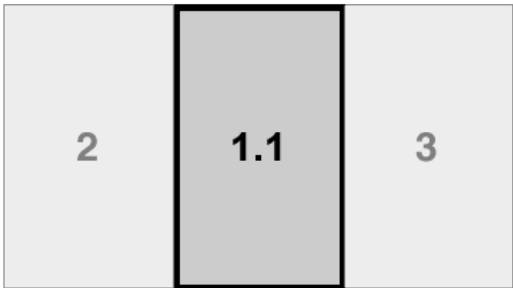
3. When viewing the first card in the stack on the right and a desktop on the left, closing the device will preserve the application on the screen, instead. Reopening the device will follow regular opening rules.

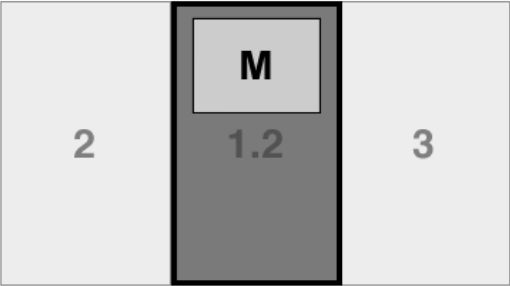
4.5 Dual-screen applications during device opening and closing

Maximized dual-screen applications are auto-minimized when the device is closed using the standard minimization rules. However, upon opening the device, an application is not automatically maximized.

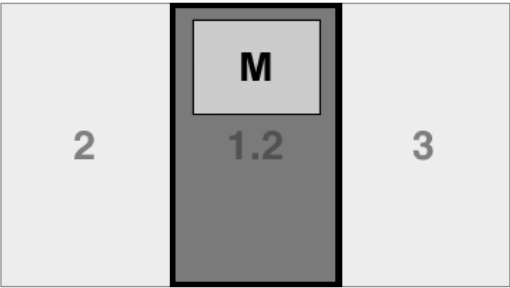
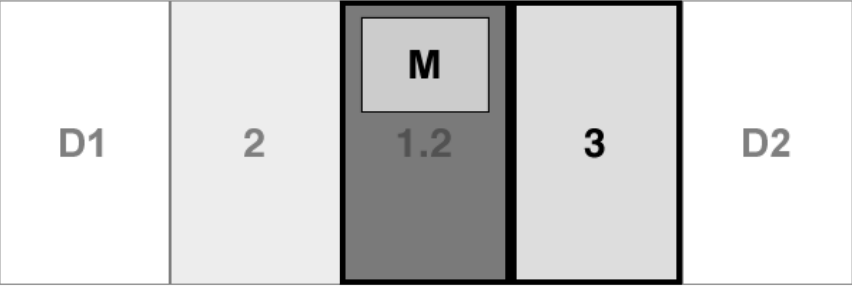
- 1. A maximized application is minimized when the device is closed; once minimized, the application is not maximized when the device is opened again.

1



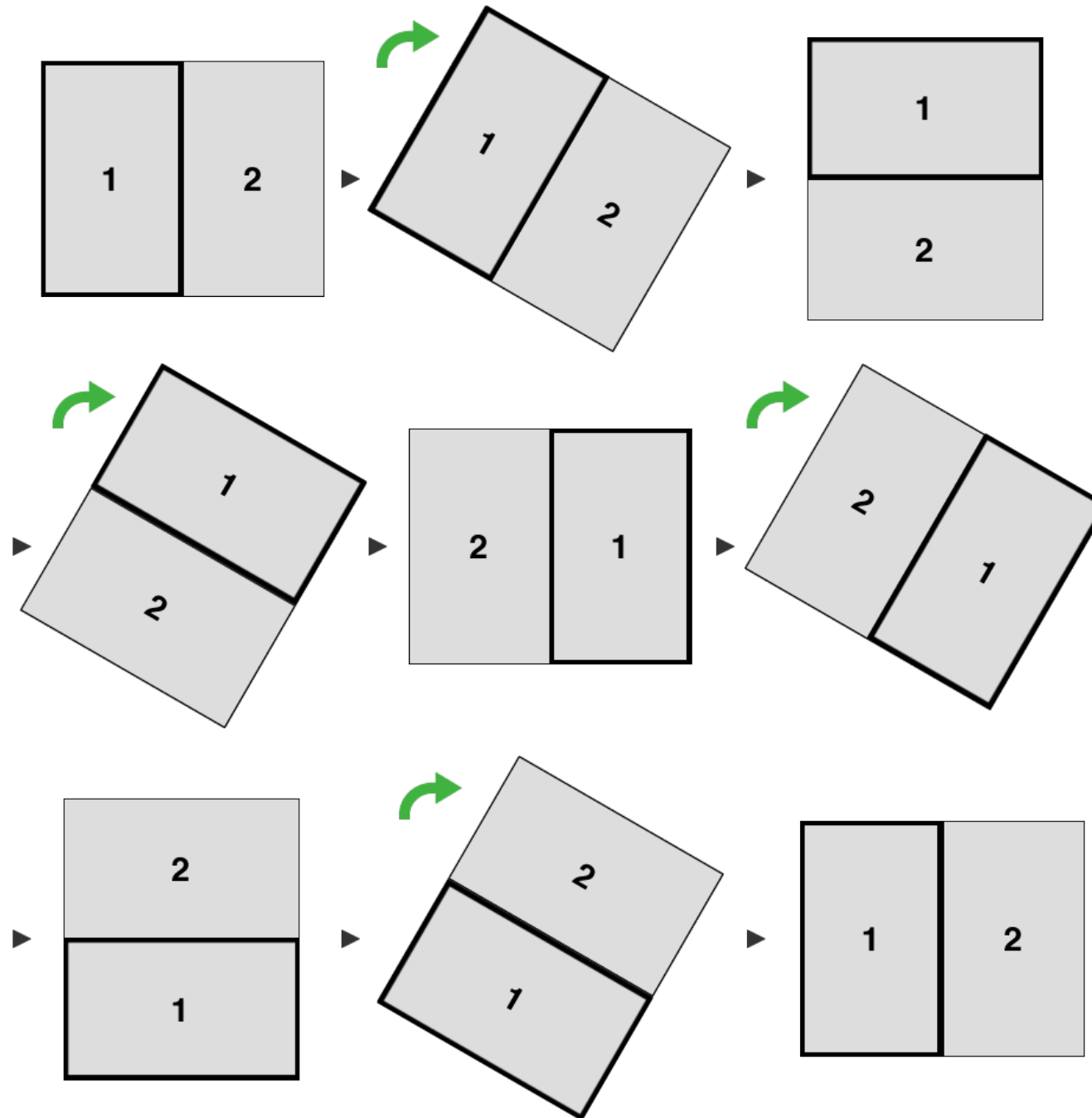


2



2. Modal views behave according to standard modal minimization rules; upon reopening the device, the view is not maximized automatically.

5 Rotation



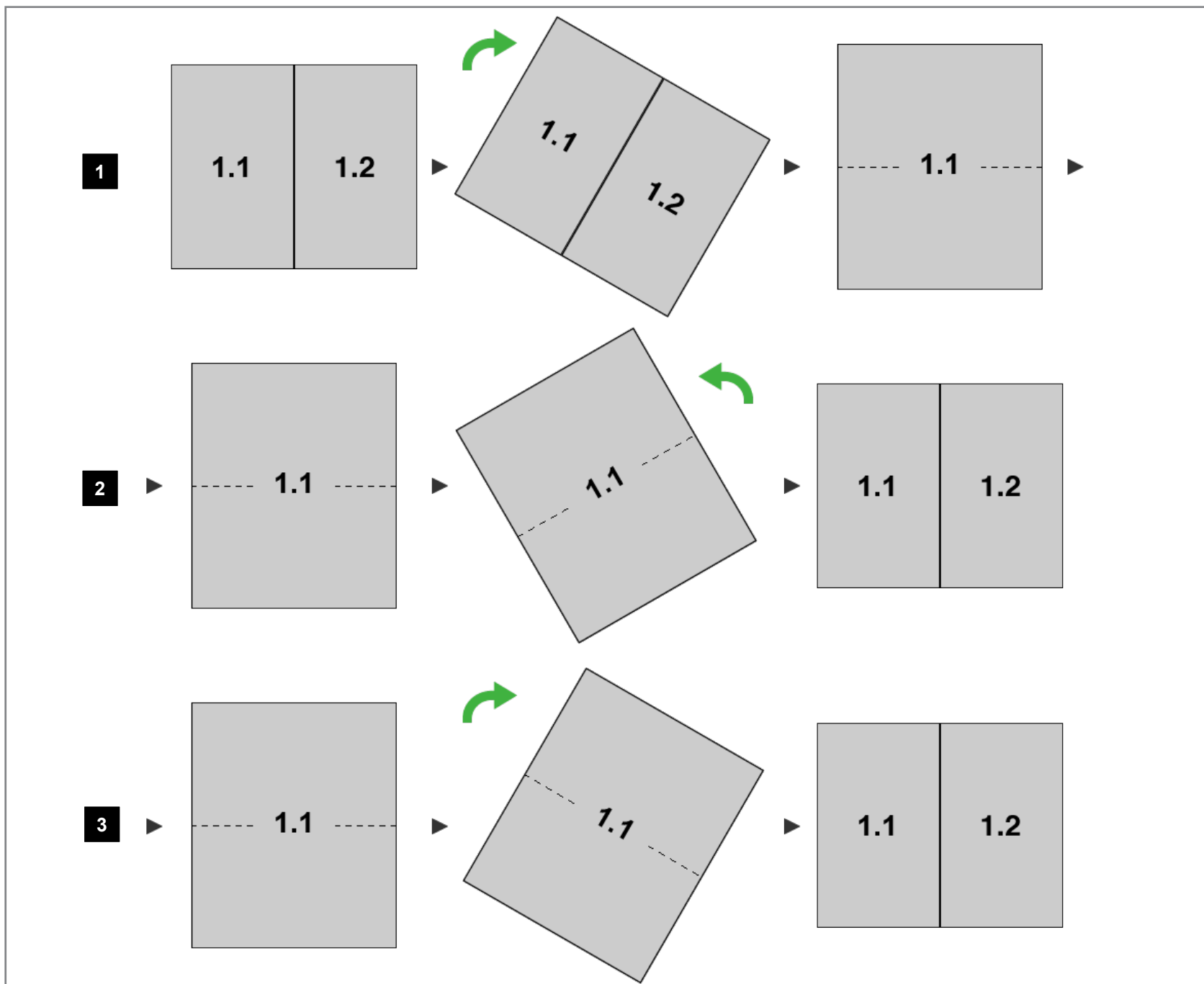
5.1 Introduction to rotation rules

Whenever rotation takes place, the applications try to preserve their state and position in the window stack.

In the example on the left, two single-screen applications are rotated 360 degrees. The primary screen is outlined. Note that after a rotation of 180 degrees, the device is upside down but the applications are rotated within their respective screens.

The motion for rotating the device is:

MOT.1.1 Rotate device



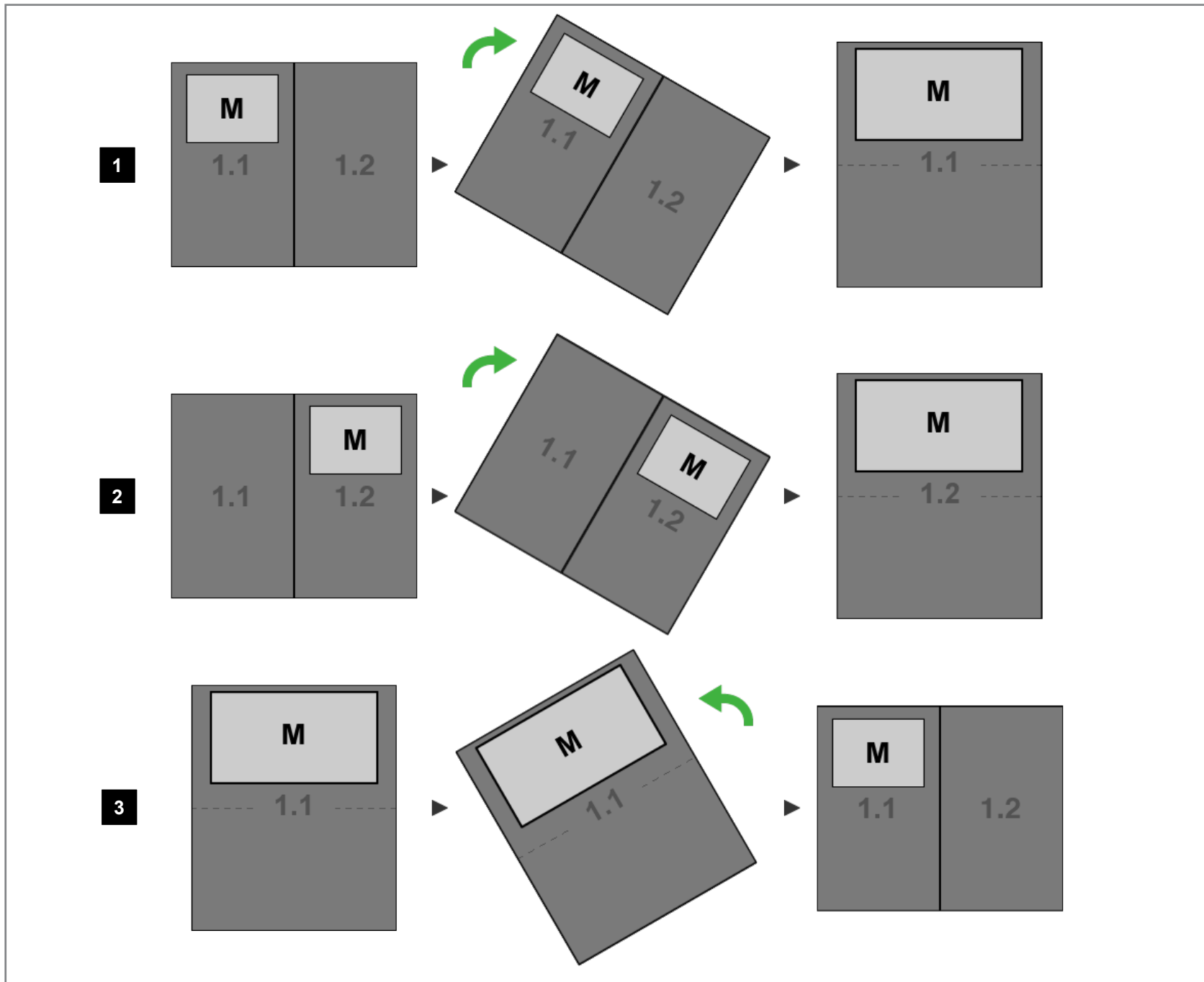
5.2 Rotating dual-screen applications

Parent-child dual-screen applications maximize across both screens when the device is rotated to LD orientation. When rotating back to PD, the maximized view is retained whenever possible.

1. A dual-screen parent-child application is rotated from PD to LD. The top view is maximized across both screens. This is known as the *Gravity drop*.
2. The device is rotated back to PD. The top view retains its position on the primary screen (left) and the application is kept in a maximized state. In this example, the parent-child application follows the maximization rules as defined.
3. The device is rotated further to PD. The top view retains its position on the primary screen (right — the device is now upside down) and the application is kept in a maximized state. In this example, the parent-child application follows the maximization rules as defined.

The motion for the Gravity drop is:

MOT.1.2 Gravity drop



5.3 Modals when rotating dual-screen applications

As a rule of thumb, modals remain on the screen they were launched on during and after rotation, whenever possible.

1. A modal on the top screen of a dual-screen application after rotation remains on the maximized view.
2. A modal on the bottom screen of a dual-screen application alters the behavior by forcing the bottom screen to expand. If the bottom screen cannot exist alone, or the view changes completely, the modal is cancelled.
3. A modal on a maximized LD dual-screen application remains on its respective view after rotation to PD.

6 Keyboard

6.1 PD keyboard behavior during WPC events



6.2 PD keyboard during minimization



6.3 LD keyboard behavior during WPC events



6.4 Keyboard rotations

TBD