December 2019

Automatic and interruptible user interface animation

Tian Liu

Andrey Kulikov

Automatic and interruptible user interface animation

ABSTRACT

User interface layouts can change due to a number of factors, e.g., resizing of windows by users, change in active application, overlay of a window by another, addition or removal of UI elements, etc. Layout changes are typically animated, e.g., there are no jump-cuts from one layout to another. During an ongoing layout animation, there can be additional layout changes that require that the animation adapt to the new target layout. Present UI frameworks or toolkits are not interruptible, e.g., they do not gracefully incorporate the new layout target; instead, they head towards the new layout target with no change in velocity, resulting in visual interruption. Current layout animations are also not automatic, e.g., developers specify the animations for layout changes. This disclosure uses physics-based techniques to effect smooth, automatic, and interruptible layout animation with no extra effort from developers.

KEYWORDS

- User interface
- UI animation
- UI library
- Animation toolkit
- Layout animation
- Interruptible animation
- Layout transition
- Physics-based animation
- Spring animation
- Velocity-aware animation

BACKGROUND

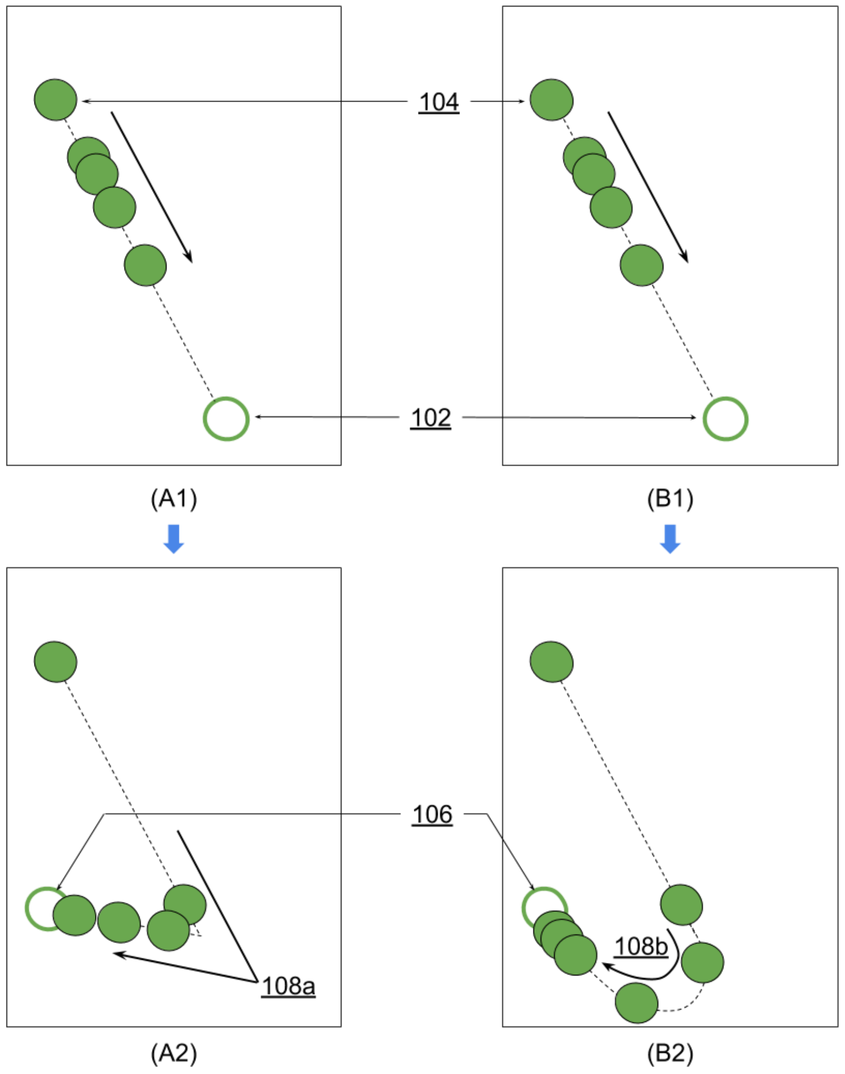

Fig. 1: (A) Non-physics based animation; (B) Physics-based animation

Fig. 1(A) illustrates an example of animation that is not based on physics and Fig. 1(B) illustrates an example of physics-based (B) animation. The animation can be used for user interface transitions. In each animation, a target object (102) is placed at a certain point on the screen and a tracking object (104) moves towards the target. The tracking object is initially at rest and gradually gathers speed. While the tracking object is still in motion, the target is moved to a new position (106) on the screen.

In the animation of Fig. A2 (not based on physics), the tracking object abruptly changes its direction of movement (108a), and, with no change in speed, hits the target object. The animation is unrealistic, e.g., visually discontinuous, due to the sudden movement and change in direction. In a physics-based animation (Fig. B2), for example, a spring-based physics animation, the tracking object is connected to the target via a virtual spring. The equations of spring motion are used to calculate the movement of the tracking object. In accordance with such equations of spring motion, when the target is moved, the tracking object gradually changes its direction of movement, e.g., along a curve (108b), and slows down as it hits the target object. The physics-based animation more gracefully incorporates changes in target positions or other interruptions in animation. The physics-based animation also enables the calculation of the movements, e.g., positions and velocities, of a tracking object independent of other objects on the screen. For example, objects that are subject to physics-based animation track their own positions and velocities autonomously, e.g., without the constraint of inter-object coordination.

DESCRIPTION

Per the techniques of this disclosure, layout changes in a user interface are automatically captured upon the event of any trigger to layout change. Layout changes can be triggered e.g., by a user causing a change in the shape or size of a window, by the overlay of a window by another, by the change of an active application or program resulting in a new active window, by the folding of a foldable screen, by screen transitions, etc. Physics-based animation is used to automatically animate the bounds of the layouts. Objects in a physics-based animation track their own position and velocity, such that a change in animation target is tracked from the current position and velocity of an object. Course correction of an object occurs in a smooth manner.

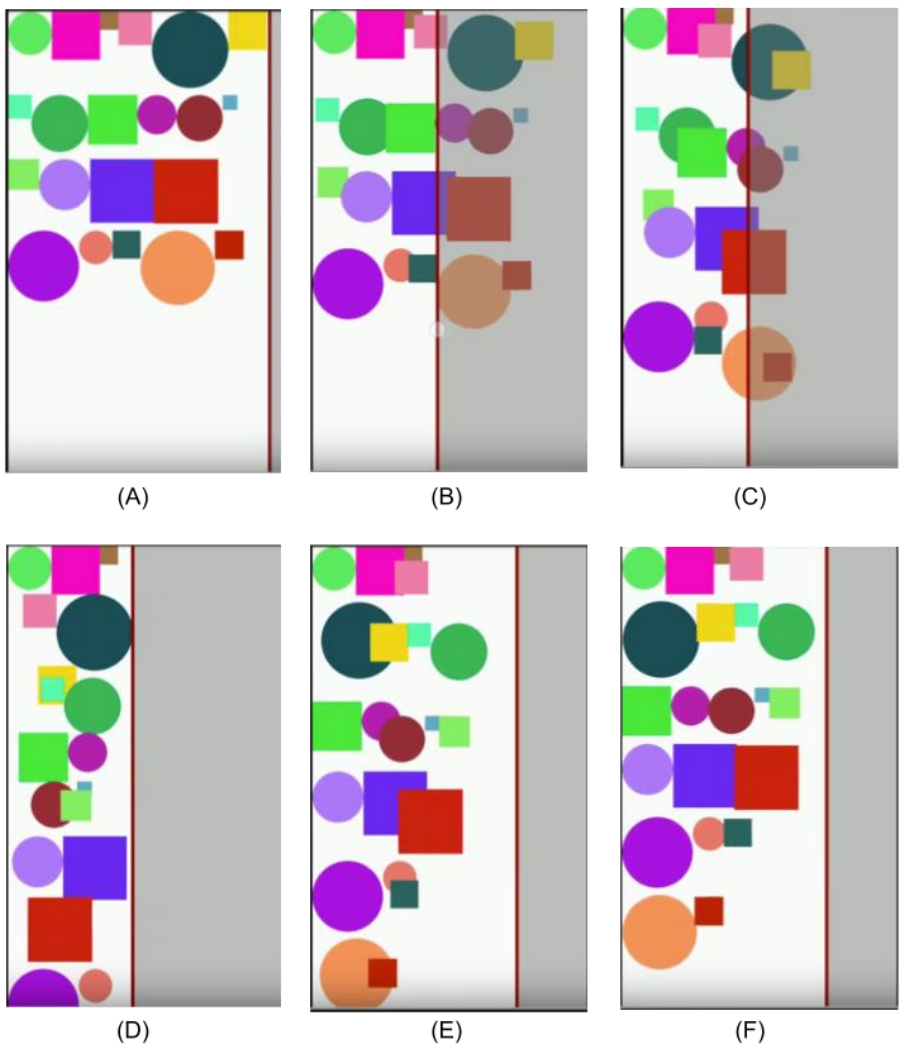

Fig. 2: Smooth, automatic, and interruptible layout animation

Fig. 2 illustrates an example of smooth, automatic, and interruptible layout animation, per techniques of this disclosure. The sequence of sub-figures A-F represent successive screen snapshots. The colored squares and circles represent layout or UI objects. The layout objects are such that they persist through layout transitions. The red vertical line that divides the grey background of the screen from the white background represents a boundary for objects within the layout. Over the sequence of Figs. 2A-D, the red line is moved rapidly and in a discrete step to the left, squeezing the layout objects to a smaller space or container. The red boundary line can be moved, e.g., by a user, by the overlay of one window by another, due to the folding of a foldable screen, etc. The layout objects respond using physics-based animation, e.g., by tracking their own positions and velocities, and by smoothly moving into the smaller space. Over the sequence of Fig. 2E-F, the red line is moved rapidly and in a discrete step to the right, resulting in a larger space for the layout objects. The objects respond using physics-based animation, e.g., by tracking their own positions and velocities, and by smoothly occupying the larger space.

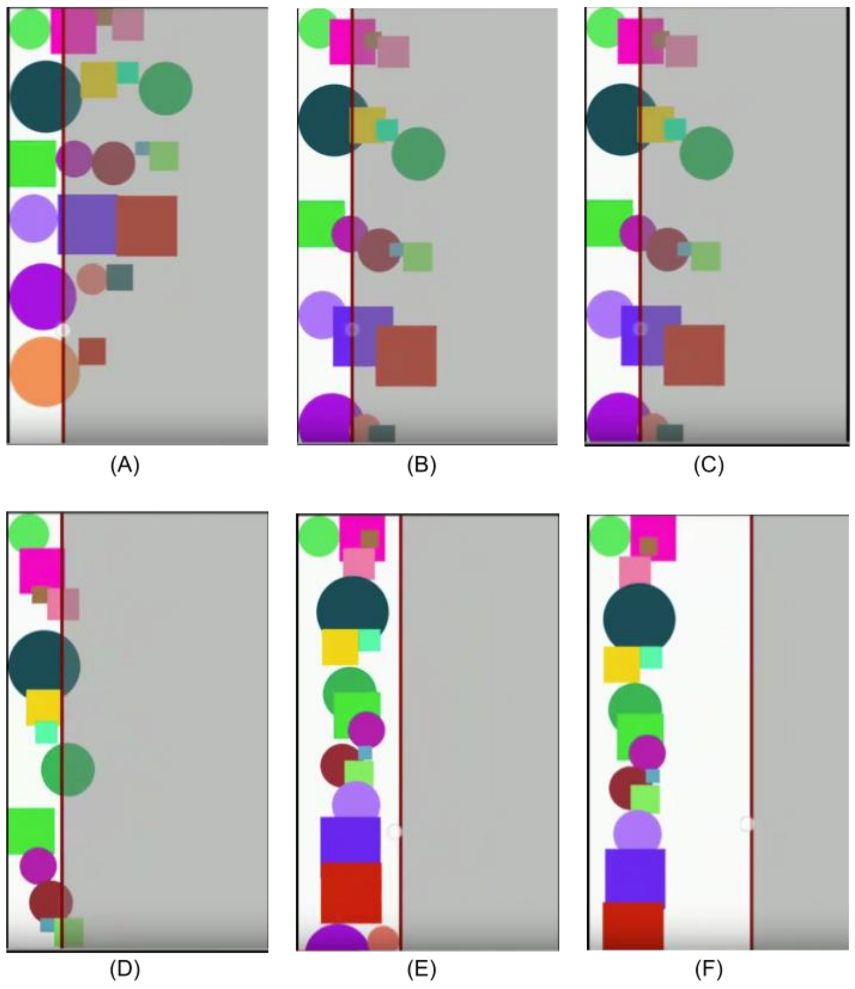

Fig. 3: Smooth, automatic, and interruptible layout animation

Fig. 3 is a continuation in time of the animation of Fig. 2. Over the sequence of Figs. 3A-D, the red line is moved rapidly and in continuous steps to the left, squeezing the layout objects to a smaller space. The layout objects respond using physics-based animation, e.g., by tracking their own positions and velocities, and by smoothly moving into the smaller space. Over Fig. 3E-F, the red line is moved slowly and in continuous steps to the right, resulting in a larger space for the layout objects. The layout objects respond using physics-based animation, e.g., by tracking their own positions and velocities, and by smoothly occupying the larger space.

The automatic and smooth physics-based layout animation disclosed herein gives the animation a unique, polished, and natural look out-of-the-box, with no developer effort. As explained before, an example of physics-based animation is spring-based physics animation, wherein layout objects are connected to their target positions on the screen via virtual springs. Under spring-based physics animation, the stiffness of the virtual springs can be exposed to application developers as a controllable factor, such that application developers can control the speed or snappiness with which layout objects move to their (new) target positions.

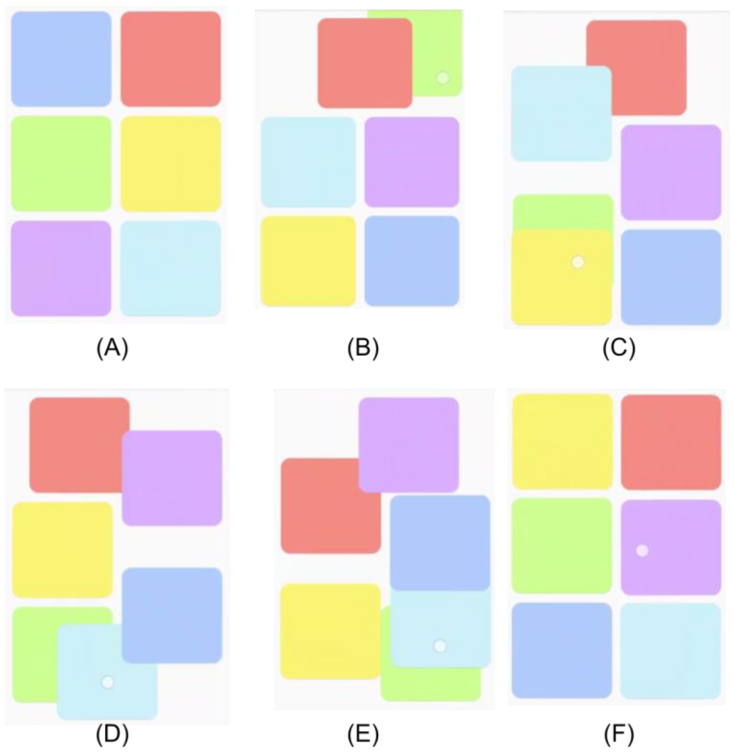

Fig. 4: Smooth, automatic, and interruptible layout animation

Fig. 4 illustrates another example of smooth, automatic, and interruptible layout animation, per techniques of this disclosure. The sequence of sub-figures A-F represents successive screen snapshots. Fig. 4A represents a starting position for six rectangular objects on a screen. In Fig. 4B, the user grabs the green object at the position represented by the white, circular cursor, and in Fig. 4C-E, the user moves the green object around the screen. As the green object is moved around the screen, the other objects continuously change their target positions to make room for the green object. The other objects use spring physics equations to move smoothly, autonomously, and automatically towards their (rapidly changing) target positions. In Fig. 4F, the user grabs the purple object and is about to start a similar motion with the purple object. The other objects animate themselves based on the equations of spring physics and the latest position of the purple object.

The animation techniques described herein can be utilized for user interface animation in any computing device. The techniques can be implemented as part of a user interface library or kit, e.g., as provided by a device operating system or other platform. Application developers can incorporate such a library in their code to provide smooth, autonomous, and interruptible animation within their applications with a polished and natural look consistent with the platform, without coding or other development work. Developers can overwrite the animation if necessary.

CONCLUSION

This disclosure uses physics-based animation to effect smooth, automatic, and interruptible layout animation with no extra effort from developers.

REFERENCES

[1] “Animate movement using spring physics,” https://developer.android.com/guide/topics/graphics/spring-animation, accessed on Nov. 29, 2019.

[2] “Auto animate layout updates,” https://developer.android.com/training/animation/layout accessed on Nov. 29, 2019.

[3] “Transition,” https://developer.android.com/reference/android/transition/Transition accessed on Nov. 29, 2019.

[4] “Manage motion and widget animation … ,” https://developer.android.com/training/constraint-layout/motionlayout accessed on Nov. 29, 2019.

[5] Young, Kenneth L., Steven C. White, Kurt B. Jacob, and Christian Schormann. "Smooth layout animation of continuous and non-continuous properties." U.S. Patent Application 12/405,213, filed March 16, 2009.

[6] Niles, Gregory E., Stephen M. Sheeler, and Guido Hücking. "User Interface for Controlling Animation of an Object." U.S. Patent Application 12/729,912, filed March 23, 2010.