Notification blocking code update

Jun Mukai <mukai@chromium.org>
2013/08/29

Background

In ChromeOS, multiple sources create notifications to notify the user some new things happening on the system (see the list of system notifiers). Some of those system notifications should appear regardless of the system's status, i.e. in the lock screen or in the fullscreen status. However, the current notification code blocks any notifications in such situations. See http://crbug.com/256404

Another thing is immersive mode in ChromeOS. Immersive mode is similar to Mac's fullscreen app, the browser controls are hidden by default so that the user see the web content fully, but the controls slide in when the user moves the mouse cursor at the top.
In this situation, the shelf is auto-hidden but appears when the user moves the cursor at the bottom of the screen and the message center is available. It does not make much sense to block the notifications in this mode. See http://crbug.com/260060

To deal with those situations, it's better to revise the current mechanism of blocking new notifications.

Note that we still have the old balloon notification code which are used in Gtk-Linux. We'll have the message center in Aura-Linux but we haven't fixed the actual schedule of Aura-transition for Linux. The new blocking mechanism should not break the balloon notification code.

How the current code works

There are multiple players in the notification areas and multiple classes have roles on the notifications.
NotificationUIManagerImpl:

- the superclass of MessageCenterNotificationManager and BalloonNotificationUIManager
- checks the current status (locked or fullscreen) and updates it. block the notifications
- polling base: check the status for every second

MessageCenter:

- nothing to do with blockings

WebNotificationTray / MessageCenterTrayBridge (cocoa):

- manages the icon appearing in the shelf / launcher / task bar
- disappears when the state has changed to locking
- block to open message center in such state
- block to create popups in such state too

The new design

NotificationUIManagerImpl

- Extract the checking code of the user state as UserStateObserver
    - Make subclass for UserStateObserverChromeOS
- Do not do anything about the user state

BalloonNotificationUIManager

- Use the UserStateObserver to block the notifications, as to keep the current behavior

MessageCenter

- Declare an interface for blocking. It should have two methods:
    - bool ShouldShowMessageCenter(): returns true if the WebNotificationTray should show its icon and should show the notification upon the user action
    - bool ShouldShowNotificationAsPopup(const NotifierId& id): returns true if the notification from the id can appear as a popup
- Add a new method: OnBlockingStateChanged()

MessageCenterNotificationUIManager

- Owns UserStateObserver and bridges it to the interfaces of blocking for MessageCenter

WebNotificationTray / MessageCenterTrayBridge

- Do not implement the conditions / logic for the blocking. Just invokes MessageCenter's blocking interface

How this design works?

When a new notification is created in the lock screen:

1. The notification arrives to MessageCenter
2. ShouldShowMessageCenter returns false, so the message center UI isn't visible
3. It creates the popup collection code or updates it
4. GetPopupNotifications() runs over the notification lists, but ShouldShowNotificationAsPopup() returns false to those notification, so it doesn't appear as a popup either
5. If the notification is an important system-level notification which should appear even in the lock screen, the platform returns true to ShouldShowNotificationAsPopup() so it appears as a popup

When the screen is locked:

1. The message center should be closed, so no notifications will be visible anymore.
2. MessageCenter::OnBlockingStateChanged() is called, which runs through the popup notifications with the new ShouldShowNotificationAsPopup() and marks-as-shown the notifications which returns false
3. So, only the important notifications remain the transition, other normal notifications are closed as a popup

When the screen is unlocked:

1. Same as a locked

2. When at least one notifications are detected to be shown as a popup, invokes UpdateNotification() to let the WebNotificationTray / popup collection class updates the list of popups, this enables to show the notifications created during locked.

Polling vs event-based

Current user-state code does the polling to check the state. This is okay but inefficient for ChromeOS. Per-platform class allows to do the checking event-based.

Brief sketch for the new interfaces

```
namespace message_center {
class NotificationBlocker {
 public:
  virtual bool ShouldShowMessageCenter() = 0;
  virtual bool ShouldShowNotificationAsPopup(const NotifierId& notifier_id) = 0;
};

// List of blockers
class FullscreenNotificationBlocker;
class FullscreenNotificationBlockerChromeos;  // for immersive mode
class ScreenlockNotificationBlocker;
class LoginNotificationBlockerChromeos;  // for start-screen in ChromeOS
// ChromeOS: FullscreenNotificationBlockerChromeos, ScreenlockNotificationBlocker, and
//     LoginNotificationBlockerChromeOS
// others: FullscreenNotificationBlocker and ScreenlockNotificationBlocker


class MessageCenter {
 public:
  void AddNotificationBlocker(NotificationBlocker* blocker);
  void RemoveNotificationBlocker(NotificationBlocker* blocker);
  void OnBlockingStateChanged();

  // Aggregates the result of blockers.
  bool ShouldShowMessageCenter();
  // seems unnecessary to aggregate ShouldShowNotificationAsPopup(), it would be inside of
  // MessageCenter class.

 private:
  std::vector<NotificationBlocker*> blockers_;
};

class NotificationList {
 public:
```

```
  // GetPopupNotifications should accept the list of blockers.
  PopupNotifications GetPopupNotifications(const std::vector<NotificationBlocker*>& blockers);
};
}  // namespace message_center

class UserStateObserver : public message_center::NotificationBlocker {
 public:
  bool IsUserActive();
  void CheckUserState();  // for polling
};
```