Known problems and solutions

Unresolved:

JDK-8280982 java.awt.Robot taking screenshots

Problem: The java.awt.Robot class provides createScreenCapture() method which currently uses X11-specific APIs

Discussion:

It is doable as of now, but not in a standard and convenient way. It can be done using some shell specific APIs, e.g. org.gnome.Shell.Screenshot DBUS API (which will not work for KDE, it has another API)

It has several drawbacks:

- There is no direct access to the screenshot data. Screenshot is saved to `png` file, after that you have to read it, decode it.
- Filesystem footprint. You need to save it to some file(its filesystem can be slow). It can be workarounded by using shared memory object though.
- Using DBUS API in this case is really slow. For instance, just a sync DBUS call to take a 1000x1000 screenshot and save it to the shared memory object file is ~14 time slower(it is just saving, without decoding and extracting data) comparing to full X11 implementation.

So, it would be great to get some standard API to get screenshot data in memory. I see that there is an open bug against this issue.

Solution: Use xdg-desktop-portal, a cross-platform stable D-Bus API which works on GNOME, KDE and wlroots-based compositors.

Solution #1.

Specifically, it provides org.freedesktop.portal.Screenshot#Screenshot which can be used to create a screenshot.

However, it has some drawbacks(at least on Gnome):

- It does not allow to take a specified screen area, only whole screen.
- ~~It asks a confirmation on each screenshot request, so it is unusable for automated testing.~~ (note, this was fixed for GNOME 43, not yet delivered to most distros)
- It saves a screenshot to disk, which must be read and deleted(may be slow depending on drive).
- A user can deny the request. In this case as a possible solution we can throw a SecurityException for Robot#createScreenCapture (javadoc update required). Currently it is thrown only if readDisplayPixels permission is not granted(which is not suitable here).
- There is no way to disable the "screen flash" after screenshot

There is also org.freedesktop.portal.Screenshot#PickColor API which can't be used for Robot#getPixelColor needs. It does not allow to take a pixel color at specified location, but interactively asks a user to pick a color with mouse cursor.

Solution #2.

Finally, another possibility is to use the Screencast portal.

This might be a bit of overkill, but it avoids several of the problems mentioned in the screenshot and color picker portals.

- implementation is more complicated comparing to Screenshot API
- A user can deny the screenshot request too.
  - Permission to make screenshots can be stored with restore_token (string), that needs to be stored safely somewhere.

    - new permission required if display set is changed or rearranged, might be an issue in case of remote automated testing.
- no intermediate file, screenshot data can be obtained from memory
- each display has its own stream, in case if requested screenshot area covers several displays resulting screenshot need to be combined from pieces.

Each solution is viable, but #2 seems to be more preferable.

However we may want to implement both of approaches:

If some new display got or disconnected, new permission request from user is required. In case of automated testing it may be a stopper. But we can fallback to a solution #1.

JDK-8280983 java.awt.Robot emulating keyboard/mouse events

Problem: The java.awt.Robot class provides methods to emulate input; keyPress(), keyRelease(), mousePress(), mouseRelease()

Discussion:

It mostly works for X11 compatibility mode (except when trying to reach outside XWayland and windows are not restacked on emulated mouse click).

Implementing this via libEI will also fix some of the X11 compatibility issues:

JDK-8280995 X11 compatibility: Robot.mouseMove does not visually move mouse cursor

JDK-8280990 X11 compatibility: XTest emulated mouse click does not bring window to front.

JDK-8280988 X11 compatibility: Click on title to request focus test failures

Solution: ~~This will probably be implemented be libEI. However, its shipping time(even estimated) is not yet known.~~

You can do most of this using the RemoteDesktop portal. Once permission is given, you can use it to completely control the keyboard and mouse.

JDK-8280984 Showing a splash screen

Problem: splash screens displays somewhere in top left

Discussion:

This is due to Wayland not allowing windows to position themselves.

This will need a Wayland protocol extension to add a role for an output-only, centered surface.

it'd just need a fallback implementation for compositors that doesn't support it, e.g. a dialog

Add input to this existing issue : https://gitlab.freedesktop.org/wayland/wayland-protocols/-/issues/67

Implementation will probably be similar to this MR for Picture-In-Picture video: https://gitlab.freedesktop.org/wayland/wayland-protocols/-/merge_requests/132

Solution: None yet (see discussion in the GitLab issue).

JDK-8280997 X11 compatibility: HiDPI.

Problem: Applications running on XWayland looks blurry on HiDPI screens

Discussion:

Quote from Maxim Kartashev's e-mail:

> *There's a quality-of-service problem with running via the compatibility layer as under certain circumstances native X windows look blurry. Users with small(ish) HiDPI displays tend to enable fractional scaling and with that enabled (regardless of the actual scale), XWayland pretends that the screen size is smaller and then pixel-stretches the resulting window according to the global scale. This works as a temporary solution, but people get quickly tired of looking at text that is blurry.*
>
> *See https://gitlab.gnome.org/GNOME/mutter/-/issues/402 and https://github.com/swaywm/sway/issues/5917 for some more details.*

Solution: None for XWayland (yet). Wayland-native applications support HiDPI

JDK-8280995 X11 compatibility: Robot.mouseMove does not visually move mouse cursor

Problem:

Javadoc for `java.awt.Robot#mouseMove` says:

```
    /**
     * Moves mouse pointer to given screen coordinates.
     * @param x         X position
     * @param y         Y position
     */
    public synchronized void mouseMove(int x, int y) {
```

Despite the fact that this mouse movement is successfully emulated within the XWayland server(following mouse press emulation will be in a right place), the mouse cursor does not visually move.

With current specification of `java.awt.Robot#mouseMove` it may be a conformance issue.

Solution:

Preferred: see earlier point on working with RemoteDesktop portal

Workaround: None

JDK-8280990 X11 compatibility: XTest emulated mouse click does not bring window to front.

Problem: window does not come to front on emulated mouse click. Window focus transfer happens without issues.

Discussion:

windowDoesNotComeOnTop.c

```
// gcc windowDoesNotComeOnTop.c -o windowDoesNotComeOnTop -lX11 -lXtst && ./windowDoesNotComeOnTop
#include <stdio.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <unistd.h>

#include <X11/extensions/XTest.h>

static Display* display;

/*
 * We have two windows, one overlapping other.
 * Call to XTestFakeButtonEvent on lower window should bring it above the other one, but this doesn't happen
on XWayland.
 * Real user mouse click works fine.
 */

static void mapNewWindow(Window *win, int x, int y, int width, int height)
{
    unsigned long background = WhitePixel(display, DefaultScreen(display));

    *win = XCreateSimpleWindow(display, DefaultRootWindow(display), x, y, width, height, 0, 0, background);

    XSizeHints    hints = {0};

    hints.flags  = PPosition | PSize;
    hints.x      = x;
    hints.y      = y;
    hints.width  = width;
    hints.height = height;

    XSetNormalHints(display, *win, &hints);
```

```
    XSelectInput(display, *win, ButtonPressMask | ButtonReleaseMask | StructureNotifyMask);
    XMapWindow(display, *win);
}

void clickOnLeftWindow(Display* display) {
    int x = 225;
    int y = 150;

    printf("XWarpPointer to %d %d\n", x, y);

    XWarpPointer(display, None, DefaultRootWindow(display), 0, 0, 0, 0, x, y);
    XSync(display, False);

    printf("XTestFakeButtonEvent\n");
    XTestFakeButtonEvent(display, 1, True, CurrentTime);
    XSync(display, False);

    XTestFakeButtonEvent(display, 1, False, CurrentTime);
    XSync(display, False);
}

int main(void) {
    XInitThreads();
    display = XOpenDisplay(NULL);

    int width = 200;
    int height = 100;

    Window w1, w2;
    mapNewWindow(&w1, 200, 50, width, height);
    mapNewWindow(&w2, 350, 50, width, height);

    XFlush(display);

    int mappedCount = 0;

    while (mappedCount < 2) {
        XEvent e;
        XNextEvent(display, &e);

        switch (e.type) {
            case MapNotify: {
                printf("%lx: MapNotify\n", e.xmap.window);
                mappedCount++;
                break;
            }
            default:
                printf("%lx: Received event type: %i\n",  e.xany.window,  e.type);

        }
    }

    sleep(2);
    clickOnLeftWindow(display);

    while (True) {
        XEvent e;
        XNextEvent(display, &e);

        switch (e.type) {
            case ButtonPress:
            case ButtonRelease: {
                printf("%lx: Button%s\n",
                       e.xbutton.window,
                       e.type == ButtonRelease ? "Release" : "Press"
                );
                break;
            }
            default:
                printf("%lx: type: %i\n",  e.xany.window, e.type);
```

```
        }
    }
}
```

> *Yes, the reason is this is using XTEST, an X11 protocol which will not work outside of X11.*
>
> *In other words, the emulated input event reaches the X11 clients, but not the Wayland compositor which is the actual display server but also the X11 window manager in Wayland, the component which is in charge of moving/resizing/stacking the windows.*
>
> *You can easily observe this using xdotool and xev in Xwayland. If the emulated click occurs within the xev window, the X11 event is logged by xev, but the window is not restacked by the Wayland compositor.*

Solution:

Preferred: emulate mouse click with libei

Workaround: modify test to use another API to bring window to front(e.g XMapRaised)

JDK-8283278 X11 compatibility: XOR rendering issue

Problem: window is not rendered correctly(wrong colors) using setXORMode in Virtual Box with 3D acceleration enabled.

Discussion:

Reproducers: java and native

E.g. how should native reproducer be displayed:

And how it actually looks like:

Reported as #1333

Solution: none yet.

Fix in progress / workaround known:

JDK-8280993 X11 compatibility: Popup menu dismiss in X11 compatibility mode.

Problem: We are using `XGrabPointer` to get mouse input and dismiss popup menus on mouse click. Obviously, it does not work outside of XWayland server.

E.g. if we have shown some popup menu, it will be closed upon clicking on any of XWayland's windows.

But it will not be closed if we click on some other window from Wayland.

```
diff --git a/src/java.desktop/unix/classes/sun/awt/X11/XPopupMenuPeer.java b/src/java.desktop/unix/classes/sun
/awt/X11/XPopupMenuPeer.java
index a19f56249ae..db88ef49f37 100644
--- a/src/java.desktop/unix/classes/sun/awt/X11/XPopupMenuPeer.java
+++ b/src/java.desktop/unix/classes/sun/awt/X11/XPopupMenuPeer.java
@@ -111,6 +111,16 @@ public class XPopupMenuPeer extends XMenuWindow implements PopupMenuPeer {
         // Get menus from the target.
         Vector<MenuItem> targetItemVector = getMenuTargetItems();
         if (targetItemVector != null) {
+            //TODO: add focus listener only for XWayland
+            target.addFocusListener(new FocusAdapter() {
+                @Override
+                public void focusLost(FocusEvent e) {
+                    target.removeFocusListener(this);
+                    if (isShowing()) {
+                        hide();
+                    }
+                }
+            });
             reloadItems(targetItemVector);
             //Fix for 6287092: JCK15a: api/java_awt/interactive/event/EventTests.html#EventTest0015 fails,
mustang
```

For a first look this workaround seems to work reliably except only one case:

clicking on window's title containing origin component does not lead to focus change, thus we don't hiding a popup.

I see a behavior similar to this workaround in some other third party application running on XWayland.

Solution: Can be not perfectly workarounded by dismissing menu on focus lost event

JDK-8280988 X11 compatibility: Click on title to request focus test failures

Problem: Some tests are trying to get focus of a window by clicking on its title bar.

Solution:

Preferred: use libEI

Workaround: Click on its body instead of decorations.

JDK-8280987 X11 compatibility: crash when mapping a lot of windows

Problem: GUI are crashing if you are trying to map a lot(> ~260) of windows at once.

Discussion:

Filed as xorg/xserver/-/issues/1222.

```
// gcc windowSpawnDoS.c -o windowSpawnDoS -lX11 && ./windowSpawnDoS
#include <X11/Xlib.h>
#include <unistd.h>

int main(void) {
    XInitThreads();
    Display* dpy = XOpenDisplay(NULL);

    Window root = DefaultRootWindow(dpy);
    for (int i = 0; i < 500; ++i) {
        Window ww = XCreateSimpleWindow(dpy, root, 50, 50, 50, 50, 0, 0, 0);
        XMapWindow(dpy, ww);
    }

    for(;;) {
        XEvent e;
        XNextEvent(dpy, &e);
    }
}
```

It may be faced when running some tests which are trying to map a lot of windows, e.g. one for every GraphicsConfiguration.

If you have 210 GraphicsConfigurations per display on Xwayland, then running such test in 2 display configuration will lead to crash.

Solution: None (yet), but known upstream at xorg/xserver/-/issues/1222.

JDK-8281612 X11 compatibility: Drag and Drop, drop performs to a wrong window from X11 client

Can't perform a Drag and Drop operation from X11 client to a Wayland client if there is an intermediate window during the drag.
Reported as #2136

Solution: waiting for a bug fix.

Fixed:

JDK-8280985 X11 compatibility: Wayland crash on huge window

```
// gcc hugeFrame.c -o hugeFrame -lX11 && ./hugeFrame
#include <X11/Xlib.h>
#include <unistd.h>

static Display* display;

int main(void) {
    int width = 22000;
    int height = 25000;

    display = XOpenDisplay(NULL);

    Window win = XCreateSimpleWindow(display, DefaultRootWindow(display), 0, 0, width, height, 0, 0L, 0L);
    XMapWindow(display, win);
    XFlush(display);

    sleep(1);
}
```

`/var/log/syslog` shows:

```
gnome-shell[1248]: WL: error in client communication (pid 1248)
gnome-shell[1298]: (EE)
gnome-shell[1298]: Fatal server error:
gnome-shell[1298]: (EE) wl_shm@5: error 1: invalid size (-2094967296)
gnome-shell[1298]: (EE)
```

where -2094967296 looks like an integer overflow of 4 * 22000 * 25000

So it fails to create a pool, but doesn't handle it gracefully.

Solution: Resolved by 1 and 2. Waiting for its propagation to Linux distros.

JDK-8280994 X11 compatibility: Drag and Drop java Wayland does not work

Drag and drop from java to wayland apps does not work with current JDK implementation.

Solution: Might be solved by updating to the latest versions of Mutter, see https://gitlab.gnome.org/GNOME/mutter/-/merge_requests/2124

JDK-8280992 X11 compatibility: Custom cursor might be displayed in wrong color

Problem: Custom cursor(e.g. loaded from GIF) set by Component#setCursor might be displayed in wrong color.

For example on following screenshot X shaped cursor should be displayed in yellow:

Reported as /xorg/xserver/-/issues/1303

Solution: Resolved by this MR. Waiting for its propagation.

JDK-8280991 X11 compatibility: fake configure event for XRandR emulation

Problem: We are not getting updates from the system after "changing" screen resolution via `XRRSetScreenConfigAndRate`.

Discussion:

Excerpt from Olivier's mail:

> *Also worth noting that the XRandR emulation is per window/X11 client, whereas the root window is shared between all X11 clients, but maybe we could send a fake ConfigureNotify event to the given client, I would need to check if that's doable.*

This notification would be helpful.

This MR adds notification, however it has several issues:

1. There is no event when you are trying to change to native resolution from another.
2. `XRRScreenChangeNotifyEvent` always has native resolution reported. (we are not using it though)
3. It might be a minor one, but on X11 session there is no events reported if you are trying to change to the same resolution.

Reported as xorg/xserver/-/issues/1305

Solution: fix released. Waiting for its propagation to verify.