一 上滑进入应用抽屉界面

1.入口

quickstep/com/android/launcher3/uioverrides/touchcontrollers/PortraitStatesTouchController.java -> canInterceptTouch(MotionEvent ev)

return false即可屏蔽该功能；

二 Launcher显示定制

launcher的显示流程：如果数据库为空则读取xml配置 -> 写入数据库 -> 绘制UI；如果有数据库，则直接从数据库中读取，再绘制。

1.基本数据定制

在res/xml/device_profiles.xml进行修改定制（所有功能均可在代码中动态修改），可以定制的功能有：workspace应用行列数、workspace文件夹行列数、数据库文件名、采用的布局文件、Hotseat应用数量、应用名文字大小、应用图标大小、应用显示区域最小宽高。

```
<grid-option
    launcher:name="m8_launcher"
    launcher:numRows="3" //workspace应用行数
    launcher:numColumns="6" //workspace应用列数
    launcher:numFolderRows="0" //workspace文件夹应用行数
    launcher:numFolderColumns="0" //workspace文件夹应用列数
    launcher:numHotseatIcons="5" //Hotseat应用数量
    launcher:dbFile="launcher_6_by_3.db" //数据库文件名
    launcher:defaultLayoutId="@xml/default_workspace_6x3" //采用的布局文件
    >

    //以下数据代码中均可以动态修改
    <display-option
        launcher:name="Super Short Stubby"
        launcher:minWidthDps="1388"
        launcher:minHeightDps="812"
        launcher:minCellHeight="30" //一个显示格子占用的最小高度，单位 dp
        launcher:minCellWidth="40" //一个显示格子占用的最小宽度，单位 dp
        launcher:iconImageSize="28" //应用图标大小
        launcher:iconTextSize="10.0" //应用名文字大小
        launcher:canBeDefault="true" //是否默认采用该布局配置 />
</grid-option>
```

1)在代码中修改minCellHeight及minCellWidth

分为三种方式：

(1)全局修改：

在src/com/android/launcher3/CellLayout.java中修改mCellWidth和mCellHeight的值：

```
public void setCellDimensions(int width, int height) {
    mFixedCellWidth = mCellWidth = xxx;
    mFixedCellHeight = mCellHeight = yyy;
    mShortcutsAndWidgets.setCellDimensions(mCellWidth, mCellHeight, mCountX,
mCountY,
            mBorderSpace);
}
```

(2)部分页面修改：

在src/com/android/launcher3/ShortcutAndWidgetContainer.java中修改mCellWidth和mCellHeight的值：

```
public void setCellDimensions(int cellWidth, int cellHeight, int countX, int
countY,
        Point borderSpace) {
    mCellWidth = xxx;
    mCellHeight = yyy;
    mCountX = countX;
    mCountY = countY;
    mBorderSpace = borderSpace;
}
```

(3)部分元素修改：

在src/com/android/launcher3/ShortcutAndWidgetContainer.java中：

```
public void measureChild(View child) {
    CellLayout.LayoutParams lp = (CellLayout.LayoutParams)
child.getLayoutParams();
    final DeviceProfile dp = mActivity.getDeviceProfile();
    if (child instanceof NavigableAppWidgetHostView) {
        ...
        lp.setup(xxx, yyy, invertLayoutHorizontally(), mCountX, mCountY,
                dp.appWidgetScale.x, dp.appWidgetScale.y, mBorderSpace,
mTempRect);
    } else {
        lp.setup(xxx, yyy, invertLayoutHorizontally(), mCountX, mCountY,
                mBorderSpace, null);
        ...
    }
}
```

或

```
public void layoutChild(View child) {
    ...
    int childLeft = lp.x;
    int childTop = lp.y;

    //此处还可定制元素的显示位置
    //childLeft = 27;
    //childTop = 110;

```

```
    lp.width = xxx;
    lp.height = yyy;

    child.layout(childLeft, childTop, childLeft + lp.width, childTop +
lp.height);
    ...
}
```

2.Workspace布局定制

在res/xml/default_workspace_xxx.xml中进行定制，可以定制的功能有：

(1)appwidget的位置、大小、第几屏；

(2)应用图标（favorite）的位置、第几屏；

(3)文件夹（folder）的位置、第几屏；

(4)快捷方式图标（shortcut）的位置、第几屏；

(5)resolve的位置、第几屏；

1)appwidget

```
<appwidget
    launcher:className="com.android.messaging.widget.BugleWidgetProvider"
    launcher:packageName="com.android.messaging"
    launcher:screen="0" //第几屏
    launcher:spanX="1" //X轴需要占用的格子数量；如何计算实际大小？上一步在
device_profiles中定义的minCellWidth × spanX即为该appwidget的实际占用大小
    launcher:spanY="1" //同上
    launcher:x="4" //起始X轴位置
    launcher:y="2" //起始Y轴位置 />

//其他图标或 widget 的预置位置不能与本 widget 所预置的区域冲突，否则会导致加载失败。
//例：
//A起始位置为x=0,y=0，X轴占用2,Y轴占用3，如果B的起始位置为x=1,Y=2，此时就会存在显示冲
突，若A先加载，会导致B无法加载，若B先加载，会导致A无法加载；
```

2)favorite

```
<favorite
    launcher:className="com.autonavi.amapauto.MainMapActivity" //类名
    launcher:packageName="com.autonavi.amapauto" //包名
    launcher:screen="1" //第几屏
    launcher:x="2" //x位置
    launcher:y="4" //y位置/>
//x和y可以用负值，如果用负数的话，右下角为(-1,-1)，如果用正数的话，左上角为(0,0)。
```

3)folder

```
<folder
    launcher:title="@string/folder_name" //文件夹名
    launcher:screen="0" //第几屏幕
    launcher:x="0" //x位置
    launcher:y="3" //y位置>
    <favorite
        launcher:packageName="com.android.settings"
        launcher:className="com.android.settings.Settings"
        launcher:x="0" //第几个/>
    <favorite
        launcher:packageName="com.android.settings"
        launcher:className="com.android.settings.Settings"
        launcher:x="1" />
</folder>
```

4)shortcut

```
<shortcut
    launcher:icon="@drawable/app_icon" //快捷方式图标
    launcher:title="@string/app_name" //标题
    launcher:uri="http://www.baidu.com/" //链接，可以是fileUri
    launcher:screen="0" //第几屏
    launcher:x="0" //x位置
    launcher:y="0" //y位置/>
```

5)resolve

没用过，不知道有什么用；

```
<resolve
    launcher:container="-101" //确定该快捷方式的显示位置 -100为workspace，-101为hotseat，不设置的话默认为-100
    launcher:screen="1" //第几屏
    launcher:x="1" //x位置
    launcher:y="0" //y位置>
    <favorite

 launcher:uri="#Intent;action=android.intent.action.MAIN;category=android.intent.category.APP_MESSAGING;end" />
    <favorite launcher:uri="sms:" />
    <favorite launcher:uri="smsto:" />
    <favorite launcher:uri="mms:" />
    <favorite launcher:uri="mmsto:" />
</resolve>
```

3.Hotseat布局定制

1)需要显示的快捷方式定制

在res/xml/default_workspace_xxx.xml中进行定制，可以定制的功能有各个应用快捷方式的位置定制：

```
<favorite
    launcher:container="-101" //确定该快捷方式的显示位置 -100为workspace，-101为hotseat，不设置的话默认为-100
    launcher:className="com.android.settings.Settings" //类名
    launcher:packageName="com.android.settings" //包名
    launcher:screen="0" //第几个位置，从左到右递增
    launcher:x="0" //因为默认情况下为横屏，且在底部，所以和screen相同
    launcher:y="0" //由于hotseat默认只有一行，所以固定为0/>

//特殊情况下，hotseat会改动到屏幕左侧或右侧，可以将屏幕进行旋转，使hotseat视觉上在屏幕下方，左侧第一个为screen=0；此时hotseat默认只有一列，所以此时x固定为0，y跟着实际走，可能递增，可能递减到0；
```

2)位置及大小定制

竖屏时，默认在屏幕下方，横屏时，默认在屏幕右侧，此方案只能让其在竖屏时从屏幕下方移动到上方，横屏时从屏幕右侧移动到屏幕左侧：

(1)在res/layout/launcher.xml中修改hotseat的位置（可能不需要，有待验证）：

```
<include
    android:id="@+id/hotseat"
    layout="@layout/hotseat"
+   android:layout_gravity="left"/>
```

(2)在src/com/android/launcher3/Hotseat.java修改hotseat的大小：

```
@Override
public void setInsets(Rect insets) {
    FrameLayout.LayoutParams lp = (FrameLayout.LayoutParams) getLayoutParams();
    DeviceProfile grid = mActivity.getDeviceProfile();
    if (grid.isVerticalBarLayout()) {
        mQsb.setVisibility(View.GONE);
        //hotseat宽高自定义
        lp.height = 600;
        lp.gravity = Gravity.LEFT | Gravity.CENTER;
        lp.width = 200;
//          if (grid.isSeascape()) {
//          } else {
//              lp.gravity = Gravity.RIGHT;
//              lp.width = grid.hotseatBarSizePx + insets.right;
//          }
        ......
    }
    ......
}
```

4.搜索框客制化

1)移除谷歌搜索框

在src\com\android\launcher3\config\BaseFlags.java中将QSB_ON_FIRST_SCREEN置为false即可；

2)移除Smartspace

(1)在src\com\android\launcher3\config\FeatureFlags.java中将QSB_ON_FIRST_SCREEN置为false；

(2)vendor\partner_gms\apps\SearchLauncher\res\xml\launcher_preferences.xml删除以下代码：

```
<androidx.preference.PreferenceScreen
    android:key="pref_smartspace"
    android:persistent="false"
    android:summary="@string/smartspace_preferences_in_settings_desc"
    android:title="@string/smartspace_preferences_in_settings"/>
```

3)置顶搜索框

SearchLauncherQuickStep 待机界面搜索框默认显示在底部 Hotseat 区域，如需搜索框在顶部显示，移除顶部固定的日期小部件和底部搜索框，并默认使用 Launcher3 中的搜索框，则需删除如下文件：

```
vendor/partner_gms/apps/SearchLauncher 目录下
// 删除底部搜索框的布局
deleted:quickstep/res/layout/search_container_all_apps.xml
// 删除顶部日期小部件的容器布局
deleted:quickstep/res/layout/search_container_workspace.xml
// 删除顶部日期小部件的布局
deleted:quickstep/res/layout/smart_space_date_view.xml
// 删除overlay的Hotseat大小以及搜索框的配置值
deleted:quickstep/res/values/dimens.xml
// 删除设置界面overlay的配置
deleted:quickstep/res/values/settings_overrides.xml

// 删除布局对应的java文件
deleted: quickstep/src/com/android/searchlauncher/HotseatQsbWidget.java
deleted:
quickstep/src/com/android/searchlauncher/QuickstepSettingsFragment.java
deleted: quickstep/src/com/android/searchlauncher/SmartSpaceHostView.java
deleted: quickstep/src/com/android/searchlauncher/SmartspaceQsbWidget.java
```

备注：删除文件可能由于 GMS 包的版本不同而导致修改失败，修改思路就是移除SearchLauncherQuickStep中对搜索框进行 overlay 的相关文件，然后修改 Hotseat 的高度。

5.屏蔽新安装应用添加到屏幕

在src/com/android/launcher3/SessionCommitReceiver.java中对ADD_ICON_PREFERENCE_KEY为false处理即可：

```
    public static boolean isEnabled(Context context) {
        /*屏蔽新安装应用添加到屏幕*/
        /*如果不让用户修改可以直接return false*/
        return
Utilities.getPrefs(context).getBoolean(ADD_ICON_PREFERENCE_KEY, false);
    }
```

三 动画相关

1.抽屉提示动画

在 Launcher 待机界面 Workspace 下面中间区域有一个小箭头，该箭头提示用户可以通过向上滑动进入应用抽屉界面。目前 Android 13.0 版本上，该箭头默认只在横屏模式下或者通过“设置 > 无障碍”，开启无障碍功能菜单的情况下才会显示，且在单层桌面下不允许显示。

如需默认显示，修改如下：

在Launcher3\src\com\android\launcher3\views\ScrimView.java中

```
+private static final boolean ALLWAYS_SHOW_DRAGHANDLE = true;
private void updateDragHandleVisibility(Drawable recycle) {
    boolean visible = mLauncher.getDeviceProfile().isVerticalBarLayout() ||
mAM.isEnabled();
+   visible |= ALLWAYS_SHOW_DRAGHANDLE;
    visible &= !MultiModeController.isSingleLayerMode();
    ......
}
```

2.缩小动画

workspace的元素进行拖动时会缩小workspace的显示区域，该动画的屏蔽方式如下（此处修改会引发一个bug，解决方案见“屏蔽缩小动画后出现闪退”）：

在src/com/android/launcher3/dragndrop/DragController.java中：

```
protected void callOnDragStart() {
    if (TestProtocol.sDebugTracing) {
        Log.d(TestProtocol.NO_DROP_TARGET, "6");
    }
    if (mOptions.preDragCondition != null) {
        mOptions.preDragCondition.onPreDragEnd(mDragObject, true /*
dragStarted*/);
    }
    mIsInPreDrag = false;
    mDragObject.dragView.onDragStart();
+   /*移除屏幕缩小动画*/
+   if (true) {
+       return;
+   }
```

```
    for (DragListener listener : new ArrayList<>(mListeners)) {
        listener.onDragStart(mDragObject, mOptions);
    }
}

```

四 点击事件相关

1.widget长按事件

在src/com/android/launcher3/touch/ItemLongClickListener.java中修改：

```
private static boolean onWorkspaceItemLongClick(View v) {
    TestLogging.recordEvent(TestProtocol.SEQUENCE_MAIN,
"onWorkspaceItemLongClick");
    Launcher launcher = Launcher.getLauncher(v.getContext());
    if (!canStartDrag(launcher)) return false;
    if (!launcher.isInState(NORMAL) && !launcher.isInState(OVERVIEW)) return
false;
    if (!(v.getTag() instanceof ItemInfo)) return false;
    /*屏蔽widget长按事件*/
    if (v.getTag() instanceof LauncherAppWidgetInfo) {
        int itemType = ((LauncherAppWidgetInfo) v.getTag()).itemType;
        if (itemType == LauncherSettings.Favorites.ITEM_TYPE_APPWIDGET ||
itemType == LauncherSettings.Favorites.ITEM_TYPE_CUSTOM_APPWIDGET) {
            return false;
        }
    }
    ...

    launcher.setWaitingForResult(null);
    beginDrag(v, launcher, (ItemInfo) v.getTag(),
launcher.getDefaultWorkspaceDragOptions());
    return true;
}
```

2.hotseat长按事件

在src/com/android/launcher3/touch/ItemLongClickListener.java中修改：

```
private static boolean onWorkspaceItemLongClick(View v) {
    ...

    /*屏蔽hotseat长按事件*/
    if (v.getTag() instanceof WorkspaceItemInfo) {
        int itemType = ((WorkspaceItemInfo) v.getTag()).container;
        if (itemType == LauncherSettings.Favorites.CONTAINER_HOTSEAT) {
            return false;
        }
    }

    //根据v.getTag()去屏蔽对应类型的点击事件；
```

```
    //LauncherAppWidgetInfo包含所有的Widget类型，其中的itemtype中又分别对应不同的widget类型
    //WorkspaceItemInfo包含hotseat和workspace中所有的对象，其中container用于区分hotseat和workspa
    ...
}
```

3.workspace空白处长按事件

在src/com/android/launcher3/touch/WorkspaceTouchListener.java中：

```
@Override
public void onLongPress(MotionEvent event) {
    if (mLongPressState == STATE_REQUESTED) {
        /*屏蔽空白处长按事件*/
        if (true) {
            return;
        }
        ...
    }
    ...
}
```

4.长按显示应用快捷菜单

在src/com/android/launcher3/util/ShortcutUtil.java中：

```
public static boolean supportsShortcuts(ItemInfo info) {
    /*屏蔽长按APP显示快捷菜单*/
    return false;
    //return isActive(info) && (isApp(info) || isPinnedShortcut(info));
}
```

五 workspace配置

1.屏蔽桌面文件夹创建

在src/com/android/launcher3/Workspace.java中：

```
boolean createUserFolderIfNecessary(View newView, int container, CellLayout target,
            int[] targetCell, float distance, boolean external, DragObject d) {
        /*取消文件夹创建*/
        if (true) return false;
    ...
}
```

2.屏蔽文件夹新增item

在src/com/android/launcher3/Workspace.java中：

```
boolean addToExistingFolderIfNecessary(View newView, CellLayout target, int[]
targetCell,
        float distance, DragObject d, boolean external) {
    /*取消文件夹可以新增item*/
    if (true) return false;
    ...
}
```

3.代码创建新的屏幕

在src/com/android/launcher3/model/BaseLoaderResults.java中：

```
public void bindWorkspace(boolean incrementBindId) {
    ...
    final IntArray orderedScreenIds = new IntArray();
    ...
    synchronized (mBgDataModel) {
        ...
        orderedScreenIds.add(0);
        orderedScreenIds.addAll(mBgDataModel.collectWorkspaceScreens());
        ...
    }
    ...
}
```

由于在代码自动创建过程中，如果某个屏幕在配置文件中显示没有任何元素，比如第0屏和第2屏各有一个APP图标，但是第1屏没有任何元素，那么在屏幕创建的时候，会忽略第1屏，只创建第0屏和第2屏，且对应的ID仍旧为0和2，导致需要展示的空白屏幕第1屏不创建，从而无法在后续过程中通过代码手动添加某些元素到第1屏上，且该屏幕顺序和id无关，只和orderedScreenIds中的排列顺序有关，比如{0,3,1,2}，那么手机上从左到右依次为第0屏、第3屏、第1屏、第2屏，且在创建过程中不能使用重复ID，后面会有重复ID检测，会crash。

遇到的BUG

1.部分应用图标无法加载

在定制过程中，可能存在部分应用的快捷方式无法正常加载，明明classname和packagename都是对的，但就是xml解析异常，无法自动写入数据库，此时可以绕过xml读取，直接写入数据库，实现方法为：

在src/com/android/launcher3/AutoInstallsLayout.java中进行配置：

```
protected class AppShortcutParser implements TagParser {
    @Override
    public int parseAndAdd(XmlPullParser parser) {
        final String packageName = getAttributeValue(parser,
ATTR_PACKAGE_NAME);
        final String className = getAttributeValue(parser, ATTR_CLASS_NAME);

        if (!TextUtils.isEmpty(packageName) &&
!TextUtils.isEmpty(className)) {
            ActivityInfo info;
```

```
            ComponentName cn;
            try {
                ......
            } catch (PackageManager.NameNotFoundException e) {
+               if (className.contains("com.baidu.baidumaps.WelcomeScreen"))
{
+                   cn = new ComponentName(packageName, className);
+                   final Intent intent = new Intent(Intent.ACTION_MAIN,
null)
+                           .addCategory(Intent.CATEGORY_LAUNCHER)
+                           .setComponent(cn)
+                           .setFlags(Intent.FLAG_ACTIVITY_NEW_TASK
+                                   |
Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED);
+                   return addShortcut("百度地图",
+                           intent, Favorites.ITEM_TYPE_APPLICATION);
                }
                Log.e(TAG, "Favorite not found: " + packageName + "/" +
className);
            }
            return -1;
        } else {
            return invalidPackageOrClass(parser);
        }
    }
    ......
}
```

2.屏蔽缩小动画后出现闪退

在缩小动画一节中屏蔽掉相关功能后，在拖动桌面元素时，超9成概率会出现launcher闪退，解决方案为屏蔽如下：

在src/com/android/launcher3/dragndrop/DragView.java中屏蔽如下代码：

```
public void move(int touchX, int touchY) {
    /*屏蔽该处代码避免拖动应用图标时闪退*/
    //if (touchX > 0 && touchY > 0 && mLastTouchX > 0 && mLastTouchY > 0
    //        && mScaledMaskPath != null) {
    //    mTranslateX.animateToPos(mLastTouchX - touchX);
    //    mTranslateY.animateToPos(mLastTouchY - touchY);
    //}
    mLastTouchX = touchX;
    mLastTouchY = touchY;
    applyTranslation();
}
```

待验证功能

1.桌面图标大小

在src/com/android/launcher3/BubbleTextView.java中 mIconSize

2.修改默认加载的布局名称

在src/com/android/launcher3/DeviceProfile.java