Android Application Development, COMP 10073

Mohawk College, Winter 2021

Navigation Drawer

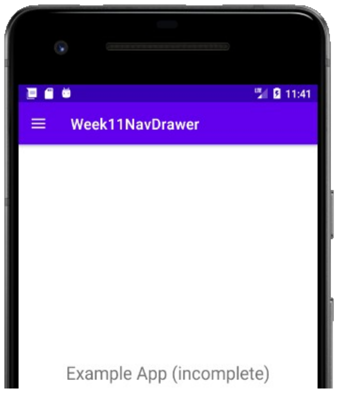
The Navigation drawer widget has changed in recent releases Like the SMS demonstration it is important that you use Marshmallow for both your SDK AND your virtual Android device. Several people in the course are testing their applications with the latest version of the Android OS. Examples from the course will not necessarily behave as described in the notes if you experiment with them using other versions.

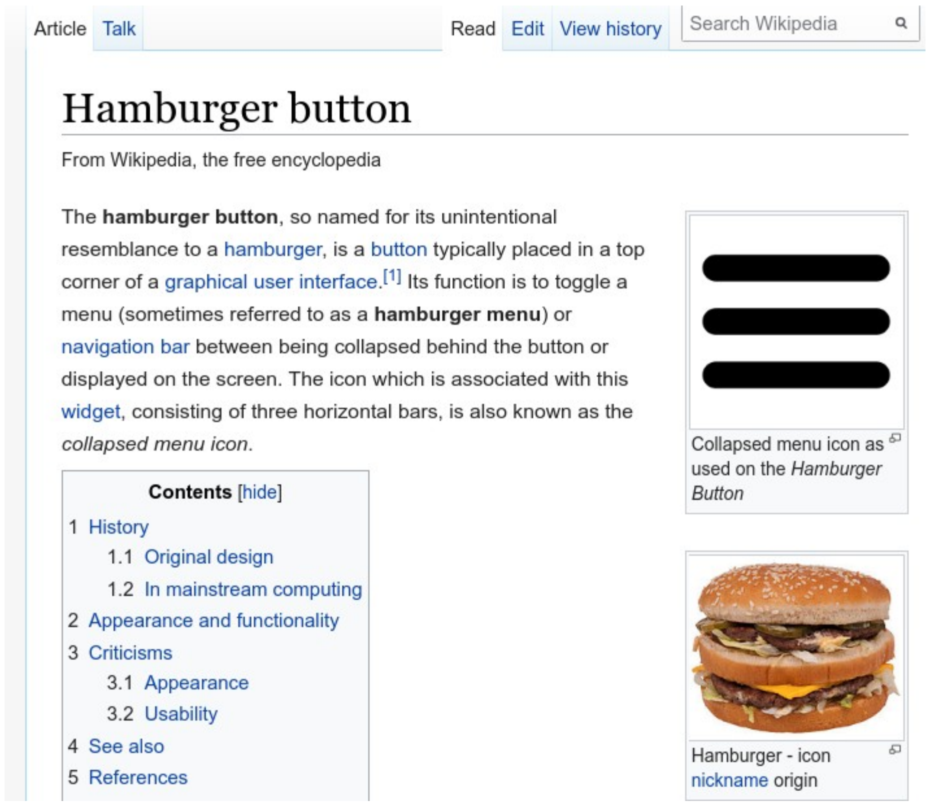
Hamburger Menu is Deprecated

Q: What is a hamburger menu?

A: An ancient idea. Origin 1981, revived 2009 to support limited screen area in mobile apps. Lately in decline.

https://en.wikipedia.org/wiki/Hamburger_button





Article Talk Read Edit View history Search Wikipedia

Hamburger button

From Wikipedia, the free encyclopedia

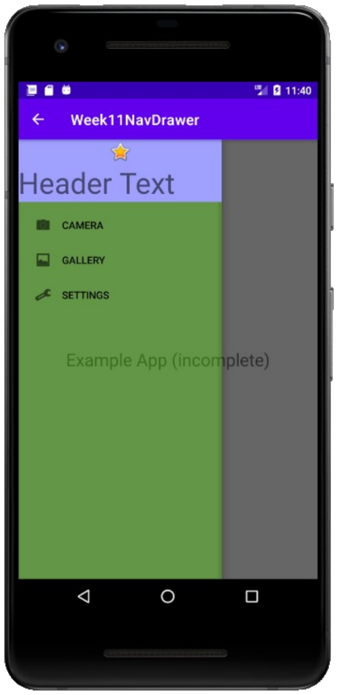
The **hamburger button**, so named for its unintentional resemblance to a hamburger, is a button typically placed in a top corner of a graphical user interface.[1] Its function is to toggle a menu (sometimes referred to as a **hamburger menu**) or navigation bar between being collapsed behind the button or displayed on the screen. The icon which is associated with this widget, consisting of three horizontal bars, is also known as the *collapsed menu icon*.

Contents [hide]

1 History
  1.1 Original design
  1.2 In mainstream computing
2 Appearance and functionality
3 Criticisms
  3.1 Appearance
  3.2 Usability
4 See also
5 References

Collapsed menu icon as used on the *Hamburger Button*

Hamburger - icon nickname origin



activity_main.xml

You will not find the drawerlayout widget in the Android Studio menus. Start with this layout. Within the drawer layout embed the NavigationView layout, which defines the drawer view menu, and also define a standard constraint layout for the main activity portion of the interface that is normally present. In this case we just give a smiple text view.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/drawer_layout"
    tools:context=".MainActivity">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Example App (incomplete)"
            android:textSize="24sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>

    <com.google.android.material.navigation.NavigationView
        android:background="#5060FF00"
        android:id="@+id/nav_view"
        app:menu="@menu/drawer_menu"
        app:headerLayout="@layout/header"
        android:layout_gravity="start"

        android:layout_width="wrap_content"
        android:layout_height="match_parent">

    </com.google.android.material.navigation.NavigationView>

</androidx.drawerlayout.widget.DrawerLayout>
```

The navigation view defines a header, which includes “Header Text” and the star icon in this example. Create an XML layout file titled header.xml.

header.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:background="#A0A0FF"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:srcCompat="@android:drawable/btn_star_big_on" />
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Header Text"
        android:textSize="40sp"
        tools:text="My Menu" />
</LinearLayout>
```

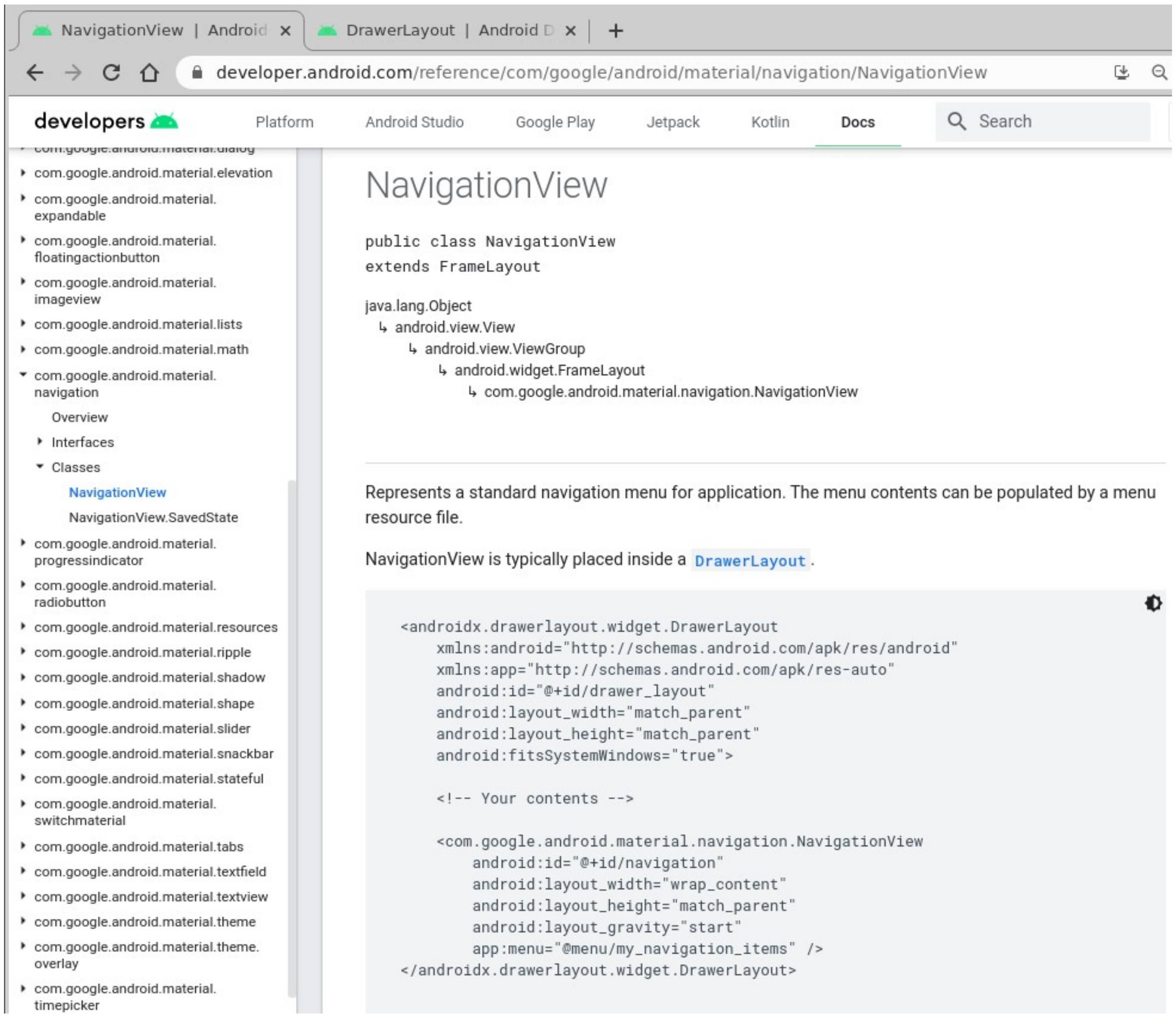
To create the menu, right click on the res directory, select new resource file, and choose menu as the file type. This will also create a menu directory.

drawer_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_camera"
            android:icon="@android:drawable/ic_menu_camera"
            android:title="CAMERA" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@android:drawable/ic_menu_gallery"
            android:title="GALLERY" />
        <item
            android:id="@+id/nav_settings"
            android:icon="@android:drawable/ic_menu_preferences"
            android:title="SETTINGS" />
    </group>
</menu>
```

NavigationView

https://developer.android.com/reference/com/google/android/material/navigation/NavigationView


NavigationView | Android
DrawerLayout | Android D
developer.android.com/reference/com/google/android/material/navigation/NavigationView
developers
Platform
Android Studio
Google Play
Jetpack
Kotlin
Docs
Search
com.google.android.material.elevation
com.google.android.material.expandable
com.google.android.material.floatingactionbutton
com.google.android.material.imageview
com.google.android.material.lists
com.google.android.material.math
com.google.android.material.navigation
Overview
Interfaces
Classes
NavigationView
NavigationView.SavedState
com.google.android.material.progressindicator
com.google.android.material.radiobutton
com.google.android.material.resources
com.google.android.material.ripple
com.google.android.material.shadow
com.google.android.material.shape
com.google.android.material.slider
com.google.android.material.snackbar
com.google.android.material.stateful
com.google.android.material.switchmaterial
com.google.android.material.tabs
com.google.android.material.textfield
com.google.android.material.textview
com.google.android.material.theme
com.google.android.material.theme.overlay
com.google.android.material.timepicker

NavigationView

```
public class NavigationView
extends FrameLayout
```

java.lang.Object
↳ android.view.View
  ↳ android.view.ViewGroup
    ↳ android.widget.FrameLayout
      ↳ com.google.android.material.navigation.NavigationView

Represents a standard navigation menu for application. The menu contents can be populated by a menu resource file.

NavigationView is typically placed inside a `DrawerLayout`.

```
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true">

    <!-- Your contents -->

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/navigation"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:menu="@menu/my_navigation_items" />
</androidx.drawerlayout.widget.DrawerLayout>
```

DrawerLayout

https://developer.android.com/reference/androidx/drawerlayout/widget/DrawerLayout

DrawerLayout acts as a top-level container for window content that allows for interactive "drawer" views to be pulled out from one or both vertical edges of the window. Drawer positioning and layout is controlled using the android:layout_gravity attribute on child views corresponding to which side of the view you want the drawer to emerge from: left or right (or start/end on platform versions that support layout direction.) Note that you can only have one drawer view for each vertical edge of the window.

To use a DrawerLayout, position your primary content view as the first child with width and height of match_parent and no layout_gravity. Add drawers as child views after the main content view and set the layout_gravity appropriately. Drawers commonly use match_parent for height with a fixed width.

NavigationView Recipe

Start with a template for onCreate. We need to implement the NavigationView Listener, include an empty template for this method.

```
public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {
    public static final String TAG = "==MainActivity==";
    private DrawerLayout myDrawer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // TODO Access myDrawer

        // TODO Access the ActionBar, enable "home" icon

        // TODO Add an ActionBarDrawerToggle element

        // TODO set up callback method for Navigation View

        Log.d(TAG, "onCreate");
    }

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        return false;
    }
}
```

First thing is to lookup the drawer_layout. This is the id of our toplevel component in the main_activity.xml file.

```
// TODO Access myDrawer
myDrawer = (DrawerLayout)
        findViewById(R.id.drawer_layout);
```

DrawerLayout.DrawerListener can be used to monitor the state and motion of drawer views. Avoid performing expensive operations such as layout during animation as it can cause stuttering; try to perform expensive operations during the STATE_IDLE state. DrawerLayout.SimpleDrawerListener offers default/no-op implementations of each callback method.

Set up the ActionBar

https://developer.android.com/reference/android/app/ActionBar

```java
// TODO Access the ActionBar, enable "home" icon
ActionBar myActionBar = getSupportActionBar();
myActionBar.setDisplayHomeAsUpEnabled(true);
```

The Action bar is a primary toolbar within the activity that may display the activity title, application-level navigation affordances, and other interactive items. The action bar appears at the top of an activity's window when the activity uses the AppCompat's AppCompat theme.

Beginning with API level 21, the action bar may be represented by any Toolbar widget within the application layout. The application may signal to the Activity which Toolbar should be treated as the Activity's action bar.

ActionBarDrawerToggle

This element is necessar to make the drawer do something.

https://developer.android.com/reference/androidx/appcompat/app/ActionBarDrawerToggle

```java
// TODO add an ActionBarDrawerToggle element
ActionBarDrawerToggle myactionbartoggle = new
        ActionBarDrawerToggle(this, myDrawer,
        (R.string.open), (R.string.close));

myDrawer.addDrawerListener(myactionbartoggle);
myactionbartoggle.syncState();
```

The given Activity will be linked to the specified DrawerLayout and its Actionbar's Up button will be set to a custom drawable. This drawable shows a Hamburger icon when drawer is closed and an arrow when drawer is open. It animates between these two states as the drawer opens. String resources must be provided to describe the open/close drawer actions for accessibility services.

```
<resources>
    <string name="app_name">Week11NavDrawer</string>
    <string name="open">\"Drawer Is Open\"</string>
    <string name="close">Drawer Is Closed</string>
</resources>
```

This class provides a handy way to tie together the functionality of DrawerLayout and the framework ActionBar to implement the recommended design for navigation drawers.

To use ActionBarDrawerToggle, create one in your Activity and call through to the following methods corresponding to your Activity callbacks:

- onConfigurationChanged
- onOptionsItemSelected

Call syncState() from your Activity's onPostCreate to synchronize the indicator with the state of the linked DrawerLayout after onRestoreInstanceState has occurred.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Find out the current state of the drawer (open or closed)
    boolean isOpen = myDrawer.isDrawerOpen(GravityCompat.START);

    // Handle item selection
    switch (item.getItemId()) {
        case android.R.id.home:
            // Home button - open or close the drawer
            if (isOpen == true) {
                myDrawer.closeDrawer(GravityCompat.START);
            } else {
                myDrawer.openDrawer(GravityCompat.START);
            }
            return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Setup the CallBack for the Menus

```
// TODO set up the callback method for Navigation View
NavigationView myNavView = (NavigationView)
        findViewById(R.id.nav_view);
myNavView.setNavigationItemSelectedListener(this);
```

In order to respond to menu clicks we have added a method to launch simple toasts. Very similar to the “onClickListener”, works like a collection of buttons.

```
// Respond to Navigation Drawer item selected
@Override
public boolean onNavigationItemSelected(@NonNull
                              MenuItem item) {

    // Show visual for selection
    item.setChecked(true);

    // Close the Drawer
    myDrawer.closeDrawers();

    switch (item.getItemId()) {
        case R.id.nav_camera:
            Toast.makeText(this, "CAMERA!",
                  Toast.LENGTH_SHORT).show();
            break;
        case R.id.nav_gallery:
            Toast.makeText(this, "GALLERY!",
                  Toast.LENGTH_SHORT).show();
            break;
        case R.id.nav_settings:
            Toast.makeText(this, "SETTINGS!",
                  Toast.LENGTH_SHORT).show();
            break;
    }
    return false;
}
```