
compose-collapsing-toolbar

A simple implementation of CollapsingToolbarLayout for Jetpack Compose

Installation

You should add `mavenCentral()` repository before installation. Then add the following line to the `dependencies` block in your app level `build.gradle`:

```
1 implementation "me.onebone:toolbar-compose:2.3.5"
```

or `build.gradle.kts`:

```
1 implementation("me.onebone:toolbar-compose:2.3.5")
```

Example

An example can be found [here](#).

Usage

Using CollapsingToolbarScaffold

`CollapsingToolbarScaffold` is a container to help you place composables and move them as a user dispatches scroll. It provides two holes where you can place your components. To use `CollapsingToolbarScaffold` you will need `CollapsingToolbarScaffoldState` which could be retrieved using `rememberCollapsingToolbarScaffoldState()`.

```
1 CollapsingToolbarScaffold(  
2     state = rememberCollapsingToolbarScaffoldState(), // provide the  
3         state of the scaffold  
4     toolbar = {  
5         // contents of toolbar go here...  
6     }  
7 ) {  
8     // main contents go here...  
9 }
```

The toolbar will collapse until it gets as small as the smallest child, and will expand as large as the largest child.

Also note that the content should be scrollable for the `CollapsingToolbarScaffold` to consume nested scroll. For `LazyColumn`, you don't have to care of anything because it is scrollable by default. `Column`, however, is not scrollable by default so you can provide `Modifier.verticalScroll()` to make a content dispatch nested scroll.

```
1 CollapsingToolbarScaffold(  
2     state = rememberCollapsingToolbarScaffoldState(), // provide the  
3     toolbar = {  
4         // contents of toolbar go here...  
5     }  
6 ) {  
7     Column(  
8         modifier = Modifier  
9             .verticalScroll(rememberScrollState()) // main content  
10            should be scrollable for CollapsingToolbarScaffold to  
11            consume nested scroll  
12     ) {  
13         // ...  
14     }  
15 }
```

By default, `CollapsingToolbar` clips content to its bounds. In order to disable it, set `toolbarClipToBounds = false` in `CollapsingToolbarScaffold`.

CollapsingToolbarScaffoldState

`CollapsingToolbarScaffoldState` is a holder of the scaffold state, such as the value of y offset and how much the toolbar has expanded. The field is public so you may use it as you need. Note that the `CollapsingToolbarScaffoldState` is stable, which means that a change on a value of the state triggers a recomposition.

```
1 val state = rememberCollapsingToolbarScaffoldState()  
2 val offsetY = state.offsetY // y offset of the layout  
3 val progress = state.toolbarState.progress // how much the toolbar is  
4     expanded (0: collapsed, 1: expanded)  
5 Text(  
6     text = "Hello World",  
7     textSize = (18 + (30 - 18) * progress).sp // text size depending on  
8     the progress  
9     // recomposed when the value of the progress is changed  
10 )
```

parallax, pin, road

You can tell children of `CollapsingToolbar` how to deal with a collapse/expansion. This works almost the same way to the `collapseMode` in the `CollapsingToolbarLayout` except for the `road` modifier.

```
1 CollapsingToolbar(/* ... */) {
2     Image(
3         modifier = Modifier.parallax(ratio = 0.2f) // parallax, pin,
4         road are available
5     )
6 }
```

road modifier

The `road()` modifier allows you to place a child relatively to the toolbar. It receives two arguments: `whenCollapsed` and `whenExpanded`. As the name suggests, these describe how to place a child when the toolbar is collapsed or expanded, respectively. This can be used to display a title text on the toolbar which is moving as the scroll is fed.

```
1 CollapsingToolbarScaffold(
2     toolbar = {
3         Text(
4             text = "Title",
5             modifier = Modifier
6                 .road(
7                     whenCollapsed = Alignment.CenterStart,
8                     whenExpanded = Alignment.BottomEnd
9                 )
10    )
11 }
12 ) {
13     // ...
14 }
```

The above code orders the title `Text` to be placed at the `CenterStart` position when the toolbar is collapsed and `BottomEnd` position when it is expanded.

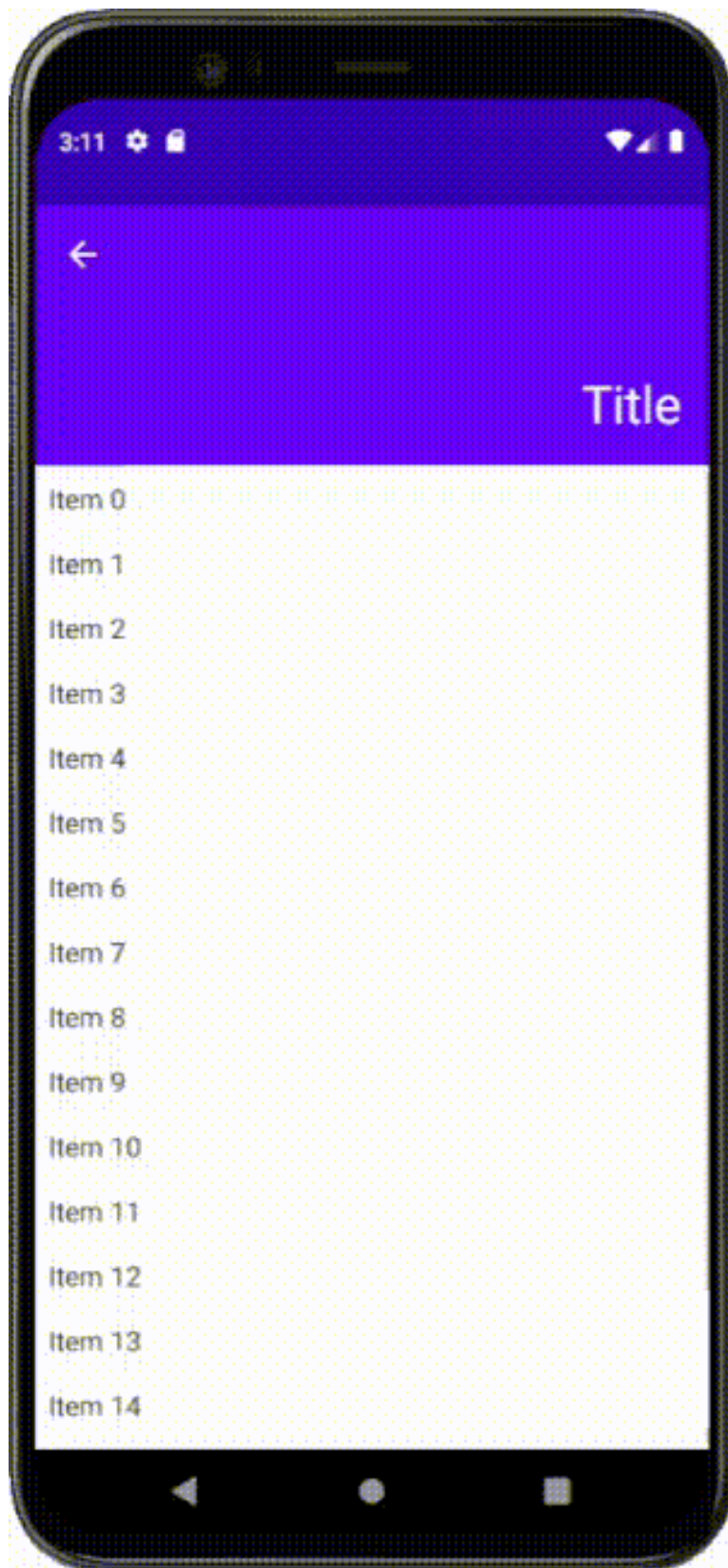
Scroll Strategy

`ScrollStrategy` defines how `CollapsingToolbar` consumes scroll. You can set your desired behavior by providing `scrollStrategy` to `CollapsingToolbarScaffold`:

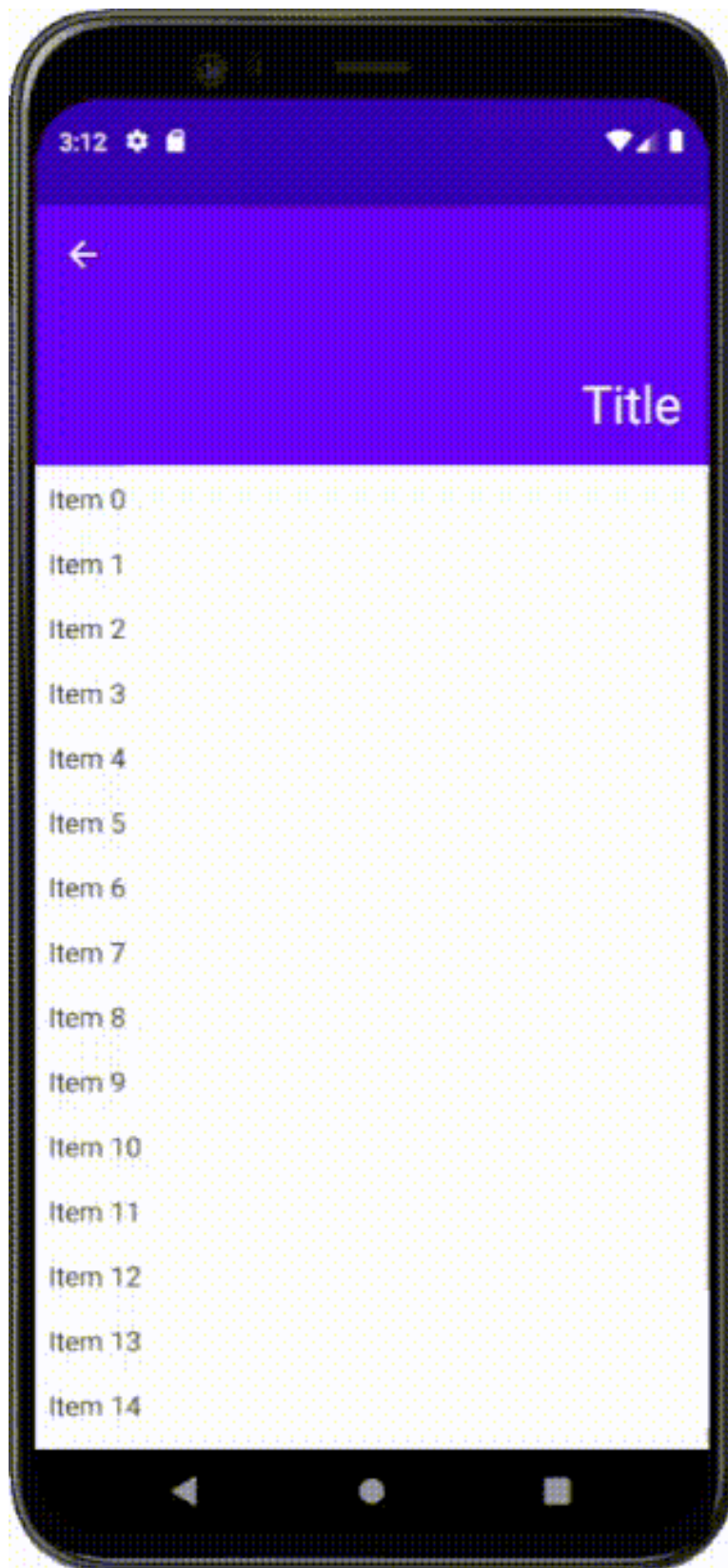
```
1 CollapsingToolbarScaffold(
```

```
2      /* ... */
3      scrollStrategy = ScrollStrategy.EnterAlways // EnterAlways,
          EnterAlwaysCollapsed, ExitUntilCollapsed are available
4  ) {
5      /* ... */
6  }
```

ScrollStrategy.EnterAlways



ScrollStrategy.EnterAlwaysCollapsed



ScrollStrategy.ExitUntilCollapsed

