Moving focus in a LazyList

+1 60 Hotlists (9) Mark as Duplicate

Comments (47) Dependencies Duplicates (1) Blocking (1/2) Resources (24)

Fixed Customer Issue P1 +

STATUS UPDATE No update yet. Edit

DESCRIPTION ra...@google.com created issue #1

Add support for moving focus in a LazyList. When we reach the end of the list, the list should automatically scroll.

Mentioned issues (4) Links (19)

Mentioned issues (4)

P4 BasicTextField crash when typing continuously and conditional modifier is applied to it during recomposition "https://issuetracker.google.com/215186908"

P2 animateScrollBy on LazyColumn not working 1.1.0-rc01 "https://issuetracker.google.com/212982472"

P1 Moving focus in a LazyGrid " ...ru: The code changed over time, and now LazyGrid uses a separate implementation. We need to repeat this implementation for LazyGrid. Filed b/242873216 to tr

P2 TwoDimensionalFocusSearch loads all LazyList content if its children contain nested focus target modifiers "I investigated the issue mentioned in #37, #39, #42, #43 and #44 because it also cr

Links (19)

"https://android-review.googlesource.com/1985706"
"https://android-review.googlesource.com/1987329"
"https://android-review.googlesource.com/1983556"
"https://android-review.googlesource.com/2029144"
"https://android-review.googlesource.com/2025106"

See all related links

COMMENTS

ni...@google.com <ni...@google.com> #2

Please ensure this works with `LazyColumn`, `LazyRow` and `LazyVerticalGrid` when this change is implemented. Thanks!

ra...@google.com <ra...@google.com> #3

Thanks Nick, LazyList is the internal component that all these composables are built out of.

ma...@gmail.com <ma...@gmail.com> #4

If I could add a small additional feature request here:

Hopefully the implementation of focus scrolling in lazy lists leaves room for the easy customization / overriding of the behavior, since TV UI often has unique snapping requirements around t

I've started to come up with my own compose API for this, and while I'm happy with the results so far, I'm hoping that any future built-in scrolling functionality will play nicely with it, or as a be

wa...@reaktor.fi <wa...@reaktor.fi> #5

To add a specific case to #4's comment on configurability:

In View Android apps, scrolling typically occurs once focus is moved to an item outside the viewport. With such behavior, however, the user is given no indication that there are more items to more to see.

In any case, glad to see this being of high priority!

hv...@gmail.com <hv...@gmail.com> #6

Looks like focus support for LazyLists did not make it to `1.1.0-beta01`.

Custom implementation that scrolls a LazyList to bring the focused element to view still almost works, but occasionally the focus moves to a random input node. The random input node can

```
2021-11-01 09:54:51.247 E/MessageQueue-JNI: java.lang.IllegalStateException: KeyEvent can't be processed because this key input node is not active.
    at androidx.compose.ui.input.key.KeyInputModifier.processKeyInput-ZmokQxo(KeyInputModifier.kt:75)
```

```
        at androidx.compose.ui.platform.AndroidComposeView.sendKeyEvent-ZmokQxo(AndroidComposeView.android.kt:439)
        at androidx.compose.ui.platform.AndroidComposeView.dispatchKeyEvent(AndroidComposeView.android.kt:446)
```

na...@vitruvian.me <na...@vitruvian.me> #7

Apologies for pinging this thread. I'm also having the IllegalState exception issue from above. Should this be pulled out into a separate ticket as opposed to part of this one or is it all sort of t

As an aside thanks for all your hard work on this. Compose is a pleasure to work with!

ra...@google.com <ra...@google.com>

Accepted by ra...@google.com.

sz...@gmail.com <sz...@gmail.com> #8

Whoever accepted this issue, you probably have 67 thousand issues, but since the error message is the same, can you please check if this focus changing keyevent stuff problem is the sam
https://issuetracker.google.com/issues/215186908
<3

ap...@google.com <ap...@google.com> #9

Project: platform/frameworks/support
Branch: androidx-main

commit ef81023fc0de83421e4779f6bef8e14d4ef6a158
Author: Ralston Da Silva <ralu@google.com>
Date: Mon Feb 14 18:00:04 2022

    Refactor focus search to accept a lambda

    Refactoring the focus search code so that it runs a lambda
    once it finds the next item. This is needed so that we can
    request focus on the next item when the next item is beyond
    visible bounds. We need this because beyond bounds layout
    requests return a block and the items are guaranteed to be
    available only within the scope of the block.

    For more info, see go/compose-focus-beyondbounds

    Bug: 184670295
    Test: Internal refactoring, existing moveFocus() tests
    Change-Id: I6545ef1a094f5bbe37eb6f861d1ea5ab6a7ec926

M   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/OneDimensionalFocusSearch.kt
M   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/FocusTraversal.kt
M   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/FocusManager.kt
M   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/TwoDimensionalFocusSearch.kt

https://android-review.googlesource.com/1985706

ap...@google.com <ap...@google.com> #10

Project: platform/frameworks/support
Branch: androidx-main

commit 3b4ce572c2fa394843ff0082cc985c8d02228560
Author: Ralston Da Silva <ralu@google.com>
Date: Tue Feb 15 16:57:08 2022

    BeyondBoundsLayout ModifierLocal

    This CL adds a BeyondBoundsLayout modifier local.

    Bug: 184670295
    Test: ./gradlew compose:ui:ui:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.ui.layout.BeyondBoundsLayoutTest
    Relnote: Added a BeyondBoundsLayout modifier local
    Change-Id: If8b51c6e08a375d1c733588e53c9b07474c0855c

A   compose/ui/ui/src/androidAndroidTest/kotlin/androidx/compose/ui/layout/BeyondBoundsLayoutTest.kt
M   compose/ui/ui/api/restricted_current.txt
M   compose/ui/ui/api/current.txt
A   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/layout/BeyondBoundsLayout.kt
M   compose/ui/ui/api/public_plus_experimental_current.txt

https://android-review.googlesource.com/1987329

ap...@google.com <ap...@google.com> #11

Project: platform/frameworks/support
Branch: androidx-main

commit e5cb882c220a13f87054aa5ddcf6410b59c12d9a
Author: Ralston Da Silva <ralu@google.com>
Date: Fri Feb 11 14:11:09 2022

    Adding Focus Group API

    Add an API to specify groups of composables that should be treated
    as a focus group. ie, we give priority to the items within the group
    before we move focus to items outside the group.

    Bug: 213508274
    Bug: 184670295
    Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.focus.FocusGroupTest
    Relnote: Added FocusGroup modifier
    Change-Id: I64bc0b945bf172ad37b64d011d7055f4a99bfeca

M	compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/Focusable.kt
M	compose/foundation/foundation/api/public_plus_experimental_current.txt
A	compose/foundation/foundation/integration-tests/foundation-demos/src/main/java/androidx/compose/foundation/demos/focus/FocusDemos.kt
A	compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/FocusGroupTest.kt
M	compose/foundation/foundation/samples/src/main/java/androidx/compose/foundation/samples/FocusableSample.kt
M	compose/foundation/foundation/integration-tests/foundation-demos/src/main/java/androidx/compose/foundation/demos/FoundationDemos.kt
M	compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/FocusableTest.kt

https://android-review.googlesource.com/1983556

ap...@google.com <ap...@google.com> #12

Project: platform/frameworks/support
Branch: androidx-main

commit fec4977a16aec28d1f56b80a941c4890189e35bc
Author: Ralston Da Silva <ralu@google.com>
Date: Wed Mar 16 01:15:29 2022

    Change BeyondBoundsLayout API

    The current API accepts two lambdas. This simplifies
    it so that we use only one lambda parameter.

    Bug: 184670295
    Test: ./gradlew compose:ui:ui:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.ui.layout.BeyondBoundsLayoutTest
    Relnote: N/A
    Change-Id: Ic41edde9b83cd5f3a602f332e5f441cd00b0be47

M	compose/ui/ui/src/androidAndroidTest/kotlin/androidx/compose/ui/layout/BeyondBoundsLayoutTest.kt
M	compose/ui/ui/api/restricted_current.txt
M	compose/ui/ui/api/current.txt
M	compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/layout/BeyondBoundsLayout.kt
M	compose/ui/ui/api/public_plus_experimental_current.txt

https://android-review.googlesource.com/2029144

ap...@google.com <ap...@google.com> #13

Project: platform/frameworks/support
Branch: androidx-main

commit 137c8716674d40eaf9411819e8e5632a7653cb08
Author: Ralston Da Silva <ralu@google.com>
Date: Mon Mar 14 13:55:18 2022

    Make Scrollable a focusGroup

    This CL makes scrollable a focus group, which ensures that we visit
    all the items in the scrollable before moving to the next focus group.

    Textfield has a scrollable that is added before the focus modifier.
    This causes issues with FocusRequester - The focusRequester is
    associated with the focusGroup instead of the focusModifier. This
    also causes issues with onFocusChanged observers that would see the
    state of the deactivated focus modifier within the focusGroup instead
    of the focus state of the focus modifier. I solved these issues by
    swapping the order of the scrollable and focusModifier in CoreTextField.

    Bug: 184670295
    Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.ScrollableTest
    Change-Id: I149acc64301086f0fd69a99f1e3c816259637367

M	compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/Focusable.kt
M	compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/ScrollableTest.kt

```
M       compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/text/CoreTextField.kt
M       compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/gestures/Scrollable.kt
```

https://android-review.googlesource.com/2025106

ap...@google.com <ap...@google.com> #14

```
Project: platform/frameworks/support
Branch: androidx-main

commit e062417461d8b52a759fab284fdf771c1046f87b
Author: Ralston Da Silva <ralu@google.com>
Date:   Thu Mar 17 15:02:33 2022

    Adding a PinnableParent ModifierLocal

    LazyLists provide a PinnableParent that can be used by components like
    focusable to prevent the currently composed children from being disposed.
    This CL introduces the PinnableParent interface that will later be
    implemented by LazyList, LazyGrid etc.

    Bug: 184670295
    Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.FocusableTest
    Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.lazy.layout.PinRequesterTest
    Relnote: Added a PinnableParent API that allows children of lazy layouts to prevent the currently composed items from being disposed
    Change-Id: Ibbdd02b0d25db2e0de343d5d2278287ab1991831

M       compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/Focusable.kt
A       compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/layout/PinnableParent.kt
M       compose/foundation/foundation/api/public_plus_experimental_current.txt
M       compose/foundation/foundation/api/current.txt
M       compose/foundation/foundation/api/restricted_current.txt
A       compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/lazy/layout/PinnableParentTest.kt
M       compose/foundation/foundation/samples/src/main/java/androidx/compose/foundation/samples/FocusableSample.kt
M       compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/FocusableTest.kt
```

https://android-review.googlesource.com/2030873

ni...@google.com <ni...@google.com> #15

Compose status update: Please update the `Status` and `Status Summary` fields of this bug! (or `Public Status`/`Public Status Summary`)

ap...@google.com <ap...@google.com> #16

```
Project: platform/frameworks/support
Branch: androidx-main

commit 68c019d14d3ab4d1e38dbe927bdcd879c4f1d47b
Author: Ralston Da Silva <ralu@google.com>
Date:   Fri Apr 08 11:58:17 2022

    Updated BeyondBoundsLayout API

    Added a BeyondBoundsLayoutScope that provides access to
    a hasMoreItems property.

    Bug: 184670295
    Test: ./gradlew compose:ui:ui:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.ui.layout.BeyondBoundsLayoutTest
    Relnote: N/A
    Change-Id: I96f1bd707a48d3195bc4ef35948b6de1b02fca7b

M       compose/ui/ui/src/androidAndroidTest/kotlin/androidx/compose/ui/layout/BeyondBoundsLayoutTest.kt
M       compose/ui/ui/api/restricted_current.txt
M       compose/ui/ui/api/current.txt
M       compose/ui/ui/api/public_plus_experimental_current.txt
M       compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/layout/BeyondBoundsLayout.kt
```

https://android-review.googlesource.com/2060011

ap...@google.com <ap...@google.com> #17

```
Project: platform/frameworks/support
Branch: androidx-main

commit a1625c770a6b7a8c22860e7347ebd91626c8e4e0
Author: Ralston Da Silva <ralu@google.com>
Date:   Thu Apr 21 01:43:21 2022

    Focus in LazyList Demo

    Add a demo to demonstrate how a user can move focus
```

within a lazyList by using the DPad.

Bug: 184670295
Test: N/A
Change-Id: I4c83e043d2042b951ec6e852a97bbcef1183f223

M compose/ui/ui/integration-tests/ui-demos/src/main/java/androidx/compose/ui/demos/UiDemos.kt
A compose/ui/ui/integration-tests/ui-demos/src/main/java/androidx/compose/ui/demos/focus/ScrollableLazyRowFocusDemo.kt

https://android-review.googlesource.com/2070449

ap...@google.com <ap...@google.com> #18

Project: platform/frameworks/support
Branch: androidx-main

commit 7979439ccc7bed82fffa252aa4794f37dc8e565d
Author: Ralston Da Silva <ralu@google.com>
Date: Thu Apr 14 17:50:03 2022

Focus search should only consider placed items

Focus search should ignore items that are not placed.
Sometimes, we compose and measure items but don't place
them. These should be excluded from focus search. A
good example of this is when LazyLists reuse layout
nodes. The cached layout nodes are attached to the
hierarchy but should be ignored by focus search if
they are not currently in use.

Bug: 184670295
Test: ./gradlew compose:ui:ui:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.ui.focus.FocusSearchNonPlacedItemsTest
Change-Id: I995990df2a171bb08cc1be2ed1d1db35cd65027a

M compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/OneDimensionalFocusSearch.kt
A compose/ui/ui/src/androidAndroidTest/kotlin/androidx/compose/ui/focus/FocusSearchNonPlacedItemsTest.kt
M compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/FocusTraversal.kt
M compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/TwoDimensionalFocusSearch.kt

https://android-review.googlesource.com/2075962

ap...@google.com <ap...@google.com> #19

Project: platform/frameworks/support
Branch: androidx-main

commit 86f7296f0e6b037823ab57ca32e271bee40c9fd5
Author: Ralston Da Silva <ralu@google.com>
Date: Thu Apr 14 13:53:13 2022

BeyondBoundsLayout modifier local for LazyList

LazyList now provides a BeyondBoundsLayout modifier local
that adds items in response to a request from its children.

LazyList has an optimization where it only places items
that are within visible bounds. This CL removes that
optimization so that we can place items beyond visible
bounds. This is needed for focus search anyway, since it
needs to search through the entire item to find the
focusable modifier.

Bug: 184670295
Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.lazy.list.LazyListBeyondBoundsTest
Relnote: N/A
Change-Id: I01c19b6e92a9eac39bea74a6cf97c50b5f6f1a0c

M compose/foundation/foundation/api/public_plus_experimental_current.txt
M compose/foundation/foundation/api/current.txt
M compose/foundation/foundation/api/restricted_current.txt
A compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/lazy/list/LazyListBeyondBoundsTest.kt
A compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyListBeyondBoundsInfo.kt
A compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyBeyondBoundsModifier.kt
M compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyList.kt
M compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyListMeasure.kt
M compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyListState.kt
M compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyMeasuredItem.kt

https://android-review.googlesource.com/2065330

ap...@google.com <ap...@google.com> #20

Project: platform/frameworks/support
Branch: androidx-main

commit 009c801a23d65b7a9d4d4cfc860386debf05db95
Author: Ralston Da Silva <ralu@google.com>
Date: Thu Apr 21 01:23:39 2022

    Adding LazyList pinning modifier

    Add a modifier that provides a PinnableParent modifier local
    implementation for LazyList.

    Bug: 184670295
    Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.lazy.list.LazyListPinningTest
    Relnote: "Added experimental BeyondBoundsInterval that can be used
     by custom implementations of LazyList when they layout items beyond visible bounds"
    Change-Id: Ifabfbd95ba53bad23ce73bdb74f816c7854222bf

M   compose/foundation/foundation/api/public_plus_experimental_current.txt
M   compose/foundation/foundation/api/current.txt
M   compose/foundation/foundation/api/restricted_current.txt
A   compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/lazy/list/LazyListPinningTest.kt
M   compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyBeyondBoundsModifier.kt
M   compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyList.kt
A   compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyListPinningModifier.kt
M   compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyListState.kt
M   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/layout/SubcomposeLayout.kt

https://android-review.googlesource.com/2070448

ap...@google.com <ap...@google.com> #21

Project: platform/frameworks/support
Branch: androidx-main

commit 43d92d01dd8deac0e51faed7c656adf4196155f1
Author: Ralston Da Silva <ralu@google.com>
Date: Tue Apr 26 16:09:07 2022

    2D Focus Search in a LazyList

    This CL adds support for 2D focus search in a lazylist

    Bug: 184670295
    Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.lazy.list.LazyListFocusMoveTest
    Relnote: N/A
    Change-Id: Id672e565d3c7ea456ada76fe8bded0636c23e166

M   compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/Focusable.kt
M   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/FocusModifier.kt
M   compose/ui/ui/api/restricted_current.txt
M   compose/ui/ui/api/current.txt
A   compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/lazy/list/LazyListFocusMoveTest.kt
A   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/BeyondBoundsLayout.kt
M   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/TwoDimensionalFocusSearch.kt
M   compose/ui/ui/api/public_plus_experimental_current.txt

https://android-review.googlesource.com/2077158

an...@nrk.no <an...@nrk.no> #22

We just updated to Compose 1.2.0-beta01 in our project and it works much better - but we still experience a few issues witht the focus jumping around.

In our setup we have a list of `LazyRow` in a `LazyColumn`. Simplified our setup can be stated like this (the issue is also with this example):

```
LazyColumn {
    items(20) { verticalIndex ->
        LazyRow {
            items(20) { horizontalIndex ->
                var color by remember { mutableStateOf(Color.White) }
                Text(
                    text = "$verticalIndex,$horizontalIndex",
                    fontSize = 50.sp,
                    textAlign = TextAlign.Center,
                    modifier = Modifier
                        .size(100.dp)
                        .border(2.dp, Color.Gray)
                        .onFocusChanged { color = if (it.isFocused) Color.Red else Color.White }
                        .background(color)
                        .focusable()
                )
            }
            }
        }
    }
```

When moving within a row by holding down the d-pad (left or right) it might sometimes lag a bit, but in the end it will end up giving focus to an expected item. However, if we navigate betwee

Are we using the combination of LazyLists wrong or is there still a bug with this custom case?

ra...@google.com <ra...@google.com> #23

Thanks for the example. Yes, this is a bug. The fix is in review.

ap...@google.com <ap...@google.com> #24

Project: platform/frameworks/support
Branch: androidx-main

commit ed9efae72e56ea2e3750ece34f914fa8d0bacd0d
Author: Ralston Da Silva <ralu@google.com>
Date: Tue May 24 12:26:35 2022

    Fix focus search in nested lazylists

    We weren't sending a pin request to grandparents, which is needed
    to support focus search through nested lazylists. This CL fixes
    that so that we can move focus among nested lazylists.

    Bug: 184670295
    Bug: 232033100
    Fixes: 232033100
    Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.lazy.list.LazyListFocusMoveTest
    Change-Id: Ia7603f3844160010ed75aaebd80a7ad83a9c1a73

M   compose/ui/ui/integration-tests/ui-demos/src/main/java/androidx/compose/ui/demos/UiDemos.kt
M   compose/foundation/foundation/src/commonMain/kotlin/androidx/compose/foundation/lazy/LazyListPinningModifier.kt
M   compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/lazy/list/LazyListFocusMoveTest.kt
A   compose/ui/ui/integration-tests/ui-demos/src/main/java/androidx/compose/ui/demos/focus/NestedLazyListFocusSearchDemo.kt
M   compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/TwoDimensionalFocusSearch.kt

https://android-review.googlesource.com/2105544

ar...@gmail.com <ar...@gmail.com> #25

Good day!

What if I need to provide concrete focus order on the page (focusProperties: next, previous) on the device with only two nav buttons left direction and right direction, how does should it look

I've tried to do example with fixes regarding focus in LazyColumn but it doesn't work for this case. Focus is going crazy.

Compose version: `1.2.0-beta02`

Example 1 - Without setting Focus options for each element of list (will fail if go to the 'After btn 1' by pressing left btn direction 2 times and then one more time left)

```
import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.focusable
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.material.OutlinedButton
import androidx.compose.material.Text
import androidx.compose.runtime.*
import androidx.compose.ui.ExperimentalComposeUiApi
import androidx.compose.ui.Modifier
import androidx.compose.ui.focus.*
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.input.key.*
import androidx.compose.ui.platform.LocalFocusManager
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp

@OptIn(ExperimentalComposeUiApi::class)
@Preview
@Composable
fun FocusTestPage() {
    val focusManager = LocalFocusManager.current
    val beforeFocus1 = remember { FocusRequester() }
    val beforeFocus2 = remember { FocusRequester() }
    val afterFocus1 = remember { FocusRequester() }
    val afterFocus2 = remember { FocusRequester() }
    val firstItemFocus = remember { FocusRequester() }
    val lastItemFocus = remember { FocusRequester() }
```

```
Column(
    modifier = Modifier.onKeyEvent {
        if (it.type != KeyEventType.KeyDown)
            return@onKeyEvent false

        return@onKeyEvent when (it.key) {
            Key.DirectionDown, Key.DirectionRight -> {
                focusManager.moveFocus(FocusDirection.Next)
                true
            }
            Key.DirectionUp, Key.DirectionLeft -> {
                focusManager.moveFocus(FocusDirection.Previous)
                true
            }
            else -> false
        }
    }
) {
    Row {
        FocusableBtn(
            text = "Before btn 1",
            modifier = Modifier.focusProperties {
                previous = afterFocus2
                next = beforeFocus2
            },
            focusRequester = beforeFocus1
        )
        FocusableBtn(
            text = "Before btn 2",
            modifier = Modifier.focusProperties {
                previous = beforeFocus1
                next = firstItemFocus
            },
            focusRequester = beforeFocus2
        )
    }

    LazyColumn(
        modifier = Modifier.height(300.dp)
    ) {
        items(20) { i ->
            when (i) {
                0 -> {
                    FocusableBtn(
                        text = "First list btn $i",
                        modifier = Modifier.focusProperties {
                            previous = beforeFocus2
                        },
                        focusRequester = firstItemFocus
                    )
                }
                19 -> {
                    FocusableBtn(
                        text = "Last list btn $i",
                        modifier = Modifier.focusProperties {
                            next = afterFocus1
                        },
                        focusRequester = lastItemFocus
                    )
                }
                else -> {
                    FocusableBtn(
                        text = "List btn $i"
                    )
                }
            }
        }
    }

    Row {
        FocusableBtn(
            text = "After btn 1",
            modifier = Modifier.focusProperties {
                previous = lastItemFocus
                next = afterFocus2
            },
            focusRequester = afterFocus1
        )
        FocusableBtn(
                text = "After btn 2",
                modifier = Modifier.focusProperties {
                    previous = afterFocus1
                    next = beforeFocus1
                },
                focusRequester = afterFocus2
            )
        }
    }
}

@Composable
fun FocusableBtn(
    text: String,
    modifier: Modifier = Modifier,
    focusRequester: FocusRequester = FocusRequester(),
) {
    var color by remember { mutableStateOf(Color.White) }
    OutlinedButton(
        modifier = Modifier
            .focusRequester(focusRequester)
            .onFocusChanged {
                color = if (it.isFocused) Color.Red else Color.White
            }
            .then(modifier)
            .focusable(),
        onClick = { /*TODO*/ },
        border = BorderStroke(2.dp, color)
    ) {
        Text(text = text)
    }
}
```

Example 2 - Setting focus options for each LazyList item (will fail with `java.lang.IllegalStateException: FocusRequester is not initialized.`)

```
package com.augvantis.careviewjetpack.pages

import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.focusable
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.material.OutlinedButton
import androidx.compose.material.Text
import androidx.compose.runtime.*
import androidx.compose.ui.ExperimentalComposeUiApi
import androidx.compose.ui.Modifier
import androidx.compose.ui.focus.*
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.input.key.*
import androidx.compose.ui.platform.LocalFocusManager
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp

@OptIn(ExperimentalComposeUiApi::class)
@Preview
@Composable
fun FocusTestPage() {
    val focusManager = LocalFocusManager.current
    val beforeFocus1 = remember { FocusRequester() }
    val beforeFocus2 = remember { FocusRequester() }
    val afterFocus1 = remember { FocusRequester() }
    val afterFocus2 = remember { FocusRequester() }
    val focusList = remember { List(20) { FocusRequester() } }

    Column(
        modifier = Modifier.onKeyEvent {
            if (it.type != KeyEventType.KeyDown)
                return@onKeyEvent false

            return@onKeyEvent when (it.key) {
                Key.DirectionDown, Key.DirectionRight -> {
                    focusManager.moveFocus(FocusDirection.Next)
                    true
                }
                Key.DirectionUp, Key.DirectionLeft -> {
                    focusManager.moveFocus(FocusDirection.Previous)
                    true
                }
                else -> false
            }
        }
    ) {
        Row {
            FocusableBtn(
                text = "Before btn 1",
                modifier = Modifier.focusProperties {
                    previous = afterFocus2
                    next = beforeFocus2
                },
                focusRequester = beforeFocus1
            )
            FocusableBtn(
                text = "Before btn 2",
                modifier = Modifier.focusProperties {
                    previous = beforeFocus1
                    next = focusList[0]
                },
                focusRequester = beforeFocus2
            )
        }

        LazyColumn(
            modifier = Modifier.height(300.dp)
        ) {
            items(20) { i ->
                when (i) {
                    0 -> {
                        FocusableBtn(
                            text = "First list btn $i",
                            modifier = Modifier.focusProperties {
                                previous = beforeFocus2
                                next = focusList[1]
                            },
                            focusRequester = focusList[0]
                        )
                    }
                    19 -> {
                        FocusableBtn(
                            text = "Last list btn $i",
                            modifier = Modifier.focusProperties {
                                previous = focusList[18]
                                next = afterFocus1
                            },
                            focusRequester = focusList[19]
                        )
                    }
                    else -> {
                        FocusableBtn(
                            text = "List btn $i",
                            modifier = Modifier.focusProperties {
                                previous = focusList[i-1]
                                next = focusList[i+1]
                            },
                            focusRequester = focusList[i]
                        )
                    }
                }
            }
        }

        Row {
            FocusableBtn(
                text = "After btn 1",
                modifier = Modifier.focusProperties {
                    previous = focusList[19]
                    next = afterFocus2
                },
                focusRequester = afterFocus1
            )
            FocusableBtn(
                text = "After btn 2",
                modifier = Modifier.focusProperties {
                    previous = afterFocus1
                    next = beforeFocus1
                },
                focusRequester = afterFocus2
            )
        }
    }
}

@Composable
fun FocusableBtn(
    text: String,
    modifier: Modifier = Modifier,
    focusRequester: FocusRequester = FocusRequester(),
) {
    var color by remember { mutableStateOf(Color.White) }
    OutlinedButton(
        modifier = Modifier
            .focusRequester(focusRequester)
            .onFocusChanged {
                color = if (it.isFocused) Color.Red else Color.White
            }
            .then(modifier)
            .focusable(),
        onClick = { /*TODO*/ },
        border = BorderStroke(2.dp, color)
    ) {
        Text(text = text)
    }
}
```

Example of focus ordering in image attachment

Thanks, Vadim

Message last modified on Jun 6, 2022 09:31PM

Mon Jun 06 2022 14:11:44 GMT+0300 (Eastern European Summer Time).png
63 KB View Download

ap...@google.com <ap...@google.com> #26

Marked as fixed.

Project: platform/frameworks/support
Branch: androidx-main

commit b1fbc771c5675eed692fd69fda1653f5147f9594
Author: Ralston Da Silva <ralu@google.com>
Date: Thu Jun 02 16:10:27 2022

    Add 1D Focus Search support for Lazylists

    1D focus search currently traverses items in the order they
    were composed. However this doesn't work for lazylists which
    reuses composed items. So we use placement order instead.

    Bug: 184670295
    Fixes: 184670295
    Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.lazy.list.LazyListFocusMoveTest
    Change-Id: I77bfa2a3017934726d3c4149ccfd30d8f682a5ae

M compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/OneDimensionalFocusSearch.kt
M compose/ui/ui/integration-tests/ui-demos/src/main/java/androidx/compose/ui/demos/focus/ScrollableLazyRowFocusDemo.kt
M compose/foundation/foundation/src/androidAndroidTest/kotlin/androidx/compose/foundation/lazy/list/LazyListFocusMoveTest.kt
M compose/ui/ui/lint-baseline.xml

https://android-review.googlesource.com/2115574

ra...@google.com <ra...@google.com> #27

Hi Vadim,

In your example above, the issue seems to be that you are adding a focusable to OutlinedButton. Buttons use a focusable internally, so you don't need to add another one. I tried out your exa

However we can simplify your sample a little, after you switch to a build that includes aosp/2115574. To get the order you want, you would have to make the Column and Row focusGroup

```
@OptIn(ExperimentalComposeUiApi::class, ExperimentalFoundationApi::class)
@Preview
@Composable
fun FocusTestPage2() {
    val focusManager = LocalFocusManager.current
    fun Modifier.only1DFocusSearch() = onKeyEvent {
        if (it.type != KeyDown) return@onKeyEvent false
        when (it.key) {
            DirectionDown, DirectionRight -> { focusManager.moveFocus(Next); true }
            DirectionUp, DirectionLeft -> { focusManager.moveFocus(Previous); true }
            else -> false
        }
    }

    Column(Modifier.only1DFocusSearch().focusGroup()) {
        Row(Modifier.focusGroup()) {
            FocusableBtn("Before btn 1")
            FocusableBtn("Before btn 2")
        }

        LazyColumn(Modifier.height(300.dp)) {
            items(20) {
                FocusableBtn("List btn $it")
            }
        }

        Row(Modifier.focusGroup()) {
            FocusableBtn("After btn 1")
            FocusableBtn("After btn 2")
        }
    }
}

@Composable
fun FocusableBtn(text: String, modifier: Modifier = Modifier) {
    var color by remember { mutableStateOf(White) }
    OutlinedButton(
        modifier = modifier.onFocusChanged { color = if (it.isFocused) Red else White },
        onClick = { /*TODO*/ },
        border = BorderStroke(2.dp, color)
    ) {
        Text(text = text)
    }
}
```

Note: An additional unrelated suggestion: Outlined Button already provides focused indication so you don't have to add the border around the box when it is focused, (unless of course, you ha

na...@vitruvian.me <na...@vitruvian.me> #28

Hi Ralston,

Apologies for pinging this issue. Not sure if its the right place but I just wanted to follow up what control we have on the scroll behaviour / if I have missed something. This is based on the Ar

We have a layout similar to below (i've also attached a sample at the end + a screenshot)

```
LazyColumn(state = lazyColumnListState) {
    items(COLUMN_ITEM_COUNT) { columnIndex ->
        Column(
            modifier = Modifier
        ) {
            Text(
                "I am section $columnIndex"
            )
            LazyRow(horizontalArrangement = Arrangement.spacedBy(4.dp)) {
                items(ROW_ITEM_COUNT) { rowIndex ->
                                       FocusableItem()

                                       .........................
```

If we rely on the bringinto view provided by the lazy list then when scrolling up only the row is brought into view (without the title).

We then naturally end up trying to use animateScrollToItem which then runs into the issues of cancelled scrolls caused by bring into view. This / workarounds are talked about here https://iss

Is there better way to control the behaviour of lazy list scrolling that I have missed or is the best workaround to copy focusable and clickable (clickable is harder as it references package inte

Examples of all the stuff mentioned above with comments available here https://github.com/nathan-castlehow/Lazyscrollingissue

Thanks! Nathan

desired_layout.png
98 KB View Download

ra...@google.com <ra...@google.com> #29

Hi Nathan,

Thanks for raising this issue. Instead of trying to cancel the default bringIntoView behavior and start a new animation, you can intercept the bringIntoView request instead:

```
@OptIn(ExperimentalFoundationApi::class)
@Composable
fun BringTitleIntoView(){
    LazyColumn {
        items(10) { columnIndex ->
            val bringIntoViewRequester = remember { BringIntoViewRequester() }
            val bringIntoViewResponder = remember { CustomBringIntoViewResponder(bringIntoViewRequester) }
            Column(
                modifier = Modifier
                    .bringIntoViewRequester(bringIntoViewRequester)
                    .bringIntoViewResponder(bringIntoViewResponder)
            ) {
                Text("I am section $columnIndex")
                LazyRow(horizontalArrangement = Arrangement.spacedBy(4.dp)) {
                    items(10) { rowIndex ->
                        Text(text = "$columnIndex $rowIndex", Modifier.focusable())
                    }
                }
            }
        }
    }
}

@OptIn(ExperimentalFoundationApi::class)
private class CustomBringIntoViewResponder(private val parent: BringIntoViewRequester) : BringIntoViewResponder {
    @ExperimentalFoundationApi
    override fun calculateRectForParent(localRect: Rect): Rect {
        return localRect
    }

    @ExperimentalFoundationApi
    override suspend fun bringChildIntoView(localRect: Rect) {
        parent.bringIntoView()
    }
}
```

an...@nrk.no <an...@nrk.no> #30

Hi Ralston.

Thanks for your replies here and in the other thread. In our case we want to center the focused row (so whatever is focused is in the middle of the screen), but struggle to do so. Seems like C

We have tried to dig a bit into the implementation and think it might be related to that the underlying scrollable starts the scroll and thus `localRect` isn't up to date when we receive it (or wh of adding bugs (or not getting the future bugfixes).

We've tried to return various rects for `calculateRectForParent` without any luck. We'll be happy to provide more info if helps.

Best regards Anders

an...@nrk.no <an...@nrk.no> #31

Hi again.

Just tried to make a simple example that shows our issue. It's really a simple version of it, but it still happens in this case. Here we're only using a lazy row, but happens for columns. If you ur

Hope this shows what our issue is.

Best regards Anders

```
@OptIn(ExperimentalFoundationApi::class)
@Composable
private fun BringIntoViewCentered() {
    val rowListState = rememberLazyListState()
    val bringIntoViewResponder = remember(rowListState) { BringIntoViewCenterResponder(rowListState) }
    LazyRow(
        state = rowListState,
        modifier = Modifier.bringIntoViewResponder(bringIntoViewResponder)
    ) {
        listOf(1.until(10), 8.downTo(1)).flatten().forEachIndexed { index, itemFraction ->
            item {
                val interactionSource = remember { MutableInteractionSource() }
                val isFocused by interactionSource.collectIsFocusedAsState()
                Text(
                    text = "#$index",
                    textAlign = TextAlign.Center,
                    modifier = Modifier
                        .fillParentMaxWidth(fraction = itemFraction.toFloat() / 10)
                        .focusable(interactionSource = interactionSource)
                        .background(if (isFocused) Color.Green else if (itemFraction % 2 == 0) Color.Red else Color.Blue)
                        .padding(vertical = 16.dp)
                )
            }
        }
    }
}
```

```
@OptIn(ExperimentalFoundationApi::class)
class BringIntoViewCenterResponder(private val listState: LazyListState) : BringIntoViewResponder {
    @ExperimentalFoundationApi
    override fun calculateRectForParent(localRect: Rect): Rect {
        return localRect
    }

    @ExperimentalFoundationApi
    override suspend fun bringChildIntoView(localRect: Rect) {
        val parentWidth = listState.layoutInfo.viewportSize.width
        val destinationLeft = (parentWidth / 2) - (localRect.width / 2)
        val scrollDelta = localRect.left - destinationLeft

        listState.stopScroll() // Just for debug purposes, animateScrollBy would normally do this
        // When scrolling to item #4 (zero-indexed) this clearly shows that the list has been scrolled after localRect was calculated
        // LOG - expected left offset: "Local rect left: ${localRect.left}"
        // LOG - actual left offset: "Visible items offset after scroll cancel: ${listState.layoutInfo.visibleItemsInfo.joinToString("; ") { "#${it.index}
        listState.animateScrollBy(scrollDelta)
    }
}
```

fa...@gmail.com <fa...@gmail.com> #32

With 1.2.0-rc03, we are also running into the same D-Pad focus issues with nested LazyLists as described in #22. I'm guessing that the fix described in #23 hasn't landed in a release yet?

is...@fubo.tv <is...@fubo.tv> #33

Comment has been deleted.

Message last modified on Jul 28, 2022 06:28PM

ap...@google.com <ap...@google.com> #34

Project: platform/frameworks/support
Branch: androidx-main

commit ec0b0c6c04008d708ffa52cd6413a48ff50ff34a
Author: Ralston Da Silva <ralu@google.com>
Date: Mon Aug 01 16:30:34 2022

    Fix Incorrect Traversal Order in 1D Focus Search

    1D Focus search visits children in composition order. However,
    if items are re-used (Eg. LazyList), we use placement order.
    When we added a fix for LazyLists, we ended up introducing a
    bug where we have an incorrect traversal order if focusable
    siblings have different layoutnode parents. This CL fixes this
    issue by using the placement order relative to the nearest
    common ancestor instead of using the placement order of the
    layout node the focus modifier is connected to.

    Bug: 238210250
    Bug: 184670295
    Fixes: 238210250
    Test: ./gradlew compose:f:f:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.foundation.lazy.list.LazyListFocusMoveTest
    Test: ./gradlew compose:ui:ui:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.ui.focus.OneDimensionalFocusSearchNextTest
    Test: ./gradlew compose:ui:ui:cC -P android.testInstrumentationRunnerArguments.class=androidx.compose.ui.focus.OneDimensionalFocusSearchPreviousTest
    Change-Id: I400efc722521516aef9449db6bd3ef403fa710e0

M compose/ui/ui/src/commonMain/kotlin/androidx/compose/ui/focus/OneDimensionalFocusSearch.kt
M compose/ui/ui/src/androidAndroidTest/kotlin/androidx/compose/ui/focus/OneDimensionalFocusSearchPreviousTest.kt
M compose/ui/ui/src/androidAndroidTest/kotlin/androidx/compose/ui/focus/OneDimensionalFocusSearchNextTest.kt

https://android-review.googlesource.com/2169776

vr...@gmail.com <vr...@gmail.com> #35

Hi there,

I'm trying to use `LazyVerticalGrid` with Dpad and noticing that the items do not scroll past the visible items. I assume as per comment #3 this behavior should work with `LazyVerticalGr`

Using compose version: 1.3.0-alpha03

Here is an example to test the focus behavior:

```
@Composable
fun GridExample() {
    LazyVerticalGrid(columns = GridCells.Adaptive(minSize = 128.dp)) {
        items(100) { index ->
            FocusableBox() {
                Text("$index")
            }
        }
    }
}

@Composable
private fun FocusableBox(
    modifier: Modifier = Modifier,
    content: @Composable BoxScope.() -> Unit = {}
) {
    var borderColor by remember { mutableStateOf(Color.Black) }
    Box(
        modifier = modifier
            .size(100.dp)
            .padding(2.dp)
            .onFocusChanged { borderColor = if (it.isFocused) Color.Red else Color.Black }
            .border(2.dp, borderColor)
            .focusable(),
        content = content
    )
}
```

Message last modified on Aug 17, 2022 08:30AM

ra...@google.com <ra...@google.com> #36

@vrkovvuru: The code changed over time, and now LazyGrid uses a separate implementation. We need to repeat this implementation for LazyGrid. Filed b/242873216 to track this.

Message last modified on Aug 18, 2022 03:41AM

do...@gmail.com <do...@gmail.com> #37

Hi there,

I'm trying to use LazyRow in LazyColumn with Dpad and I saw some delay with move focus to down. It seems to be due to the recomposition of all items in the LazyRow from which the focus

Using compose version: 1.2.0

Here is an example to test the focus behavior:

```
LazyColumn {
    items(20) { verticalIndex ->
        LazyRow{
            items(500) { horizontalIndex ->
                println("There is recomposition item with index=$verticalIndex,$horizontalIndex")
                var color by remember { mutableStateOf(Color.White) }
                Text(
                    text = "$verticalIndex,$horizontalIndex",
                    fontSize = 50.sp,
                    textAlign = TextAlign.Center,
                    modifier = Modifier
                        .size(100.dp)
                        .border(2.dp, Color.Gray)
                        .onFocusChanged { color = if (it.isFocused) Color.Red else Color.White }
                        .background(color)
                        .focusable()
                )
            }
        }
    }
}
```

Message last modified on Aug 19, 2022 07:17PM

sa...@gmail.com <sa...@gmail.com> #38

Hello, I'm trying to replicate Leanback focus behavior and failed. What I'm came up with is that I totally abandoned LazyColumn DPad handling and overrided keyEvent. I store focus requeste

```
        .onKeyEvent {
                            // completeley custom focus handling
                            // the relevant issue https://issuetracker.google.com/issues/184670295
                            // it may be better in the future
                            val code = it.nativeKeyEvent.keyCode
                            var pos = focusedItemPosition.value
                            var move = false
                            var isUp = false
                            var direction = FocusDirection.Next
```

```
                    if (it.nativeKeyEvent.action == KeyEvent.ACTION_DOWN) {
                        when (code) {
                            KeyEvent.KEYCODE_DPAD_LEFT -> {
                                direction = FocusDirection.Left
                                move = true
                                pos--
                            }
                            KeyEvent.KEYCODE_DPAD_RIGHT -> {
                                direction = FocusDirection.Right
                                move = true
                                pos++
                            }
                            KeyEvent.KEYCODE_DPAD_UP -> {
                                direction = FocusDirection.Up
                                move = true
                                pos -= columnsCount
                            }
                            KeyEvent.KEYCODE_DPAD_DOWN -> {
                                direction = FocusDirection.Down
                                move = true
                                pos += columnsCount
                            }
                            else -> Unit
                        }
                        isUp = pos < 0
                        pos = pos.coerceIn(0, collectionItems.lastIndex)
                        focusedItemPosition.value = pos
                    }
                    if (move) {
                        scope.launchSafe {
                            if (focusedItemPosition.value != pos) return@launchSafe
                            if (isUp) {
                                lazyGridState.animateScrollToItem(0, 0)
                                focusManager.moveFocus(FocusDirection.Up)
                            } else {
                                val animateTo = (pos / columnsCount) + 1
                                lazyGridState.animateScrollToItem(animateTo, -150)
                                val fr = focusRequesterByPosition[pos]?.invoke()
                                if (fr == null) {
                                    focusManager.moveFocus(direction) // possible fix crash
                                } else {
                                    fr.requestFocus()
                                }
                            }
                        }
                    }
                    return@onKeyEvent move
                }
```

Sometimes, however, focus jumps to unknown location and causing a crash :(

pi...@gmail.com <pi...@gmail.com> #39

I've hit exactly the same performance issue as #37. Even weirder: scrolling down is painfuly slow. Butscrolling back up is fine.

Compose 1.3.2.

ra...@google.com <ra...@google.com> #40

Thanks for reporting this issue. It should be fixed by https://android-review.googlesource.com/c/platform/frameworks/support/+/2255266

Message last modified on Oct 21, 2022 04:32AM

na...@google.com <na...@google.com> #41

The following release(s) address this bug:

- `androidx.compose.foundation:foundation:1.3.0`
- `androidx.compose.ui:ui:1.3.0`

pi...@gmail.com <pi...@gmail.com> #42

It still looks broken/slow in `1.4.0-alpha01`.

jo...@svt.se <jo...@svt.se> #43

Very slow in `1.4.0-alpha02`. The issue seems to be when BringIntoViewRequester is used

ni...@svt.se <ni...@svt.se> #44

Keyboard navigation lags a lot when using FocusGroup and FocusProperties, and I've not been able to get the functionality that I want without them.
Problem seems to be happening directly when using FocusGroup and FocusProperties on both LazyRow an TvLazyRow.

el...@gmail.com <el...@gmail.com> #45

I investigated the issue mentioned in #37, #39, #42, #43 and #44 because it also critically impacts me and created a new bug: b/269627309. I hope the details aid in fixing this issue faster. M

Message last modified on Feb 18, 2023 05:34AM

wa...@gmail.com <wa...@gmail.com> #46

I am using androidx.compose.ui:ui:1.5.0. I found focus will jump to unexpected item when using LazyVerticalGrid.

```
@Composable
fun GridView() {
    LazyVerticalGrid(columns = GridCells.Adaptive(minSize = 128.dp)) {
        items(100) { index ->
            FocusableBox(index = index) {
                Text("$index")
            }
        }
    }
}

@Composable
private fun FocusableBox(
    modifier: Modifier = Modifier,
    index: Int,
    content: @Composable BoxScope.() -> Unit = {}
) {
    var borderColor by remember { mutableStateOf(Color.Black) }
    Box(
        modifier = modifier
            .size(100.dp)
            .padding(2.dp)
            .onFocusChanged {
                Log.d("Test", "onFocusChanged index: $index  $it")
                borderColor = if (it.isFocused) Color.Red else Color.Black
            }
            .border(2.dp, borderColor)
            .focusable(),
        content = content
    )
}
```

Each row has 7 items. Current focus item is 0. After press Dpad key down, the expected focus item should be 7. But it moved to 8.

Output log:

```
2023-08-29 14:52:01.493 23514-23514  onFocusChanged index: 0  Inactive
2023-08-29 14:52:01.504 23514-23514  onFocusChanged index: 7  Active
2023-08-29 14:52:01.550 23514-23514  onFocusChanged index: 7  Inactive
2023-08-29 14:52:01.551 23514-23514  onFocusChanged index: 8  Active
```

Message last modified on Aug 29, 2023 05:22PM

mb...@gmail.com <mb...@gmail.com> #47

Noticed the same thing with 1.5.0 as above. Looks like regression.