[Используйте отладчик в Android Studio](https://developer.android.com/codelabs/basic-android-kotlin-compose-intro-debugger?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-2-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-intro-debugger)

## [**3. Получите стартовый код**](https://developer.android.com/codelabs/basic-android-kotlin-compose-intro-debugger?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-2-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-intro-debugger#2)

Откройте свой проект DiceRoller

## [**4. Запустите отладчик**](https://developer.android.com/codelabs/basic-android-kotlin-compose-intro-debugger?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-2-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-intro-debugger#3)

Есть два способа запустить отладчик вместе с вашим приложением:

* Присоедините отладчик к существующему процессу приложения, который выполняется на устройстве или в эмуляторе.
* Запустите приложение с помощью отладчика.

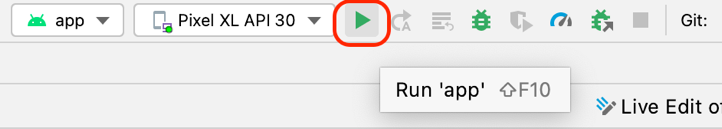
Оба пути в определенной степени достигают одного и того же. Освоив оба способа, вы сможете выбрать тот, который вам больше нравится, или тот, который вам нужен.

## **Присоедините отладчик к процессу приложения**

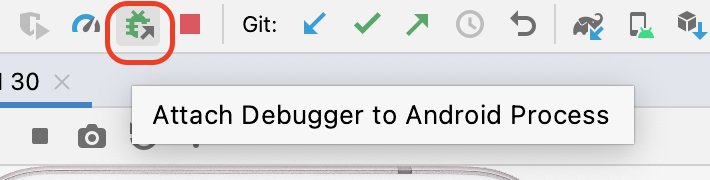
Если ваше приложение уже запущено, вы можете подключить к нему отладчик.

Чтобы подключить отладчик к процессу приложения, выполните следующие действия:

1. Нажмите  **«Run app»** .

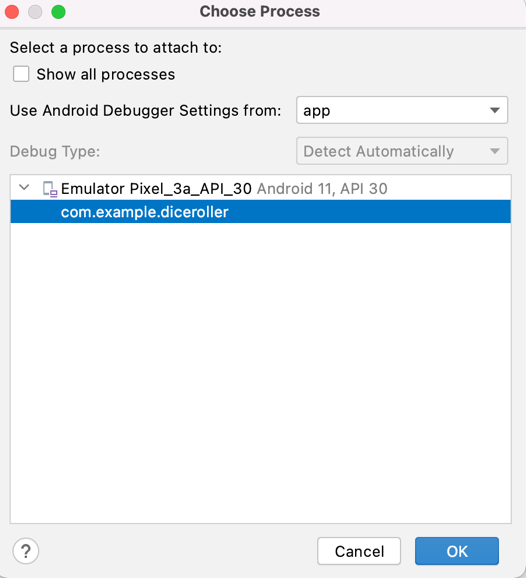


1. После запуска приложения на устройстве или в эмуляторе нажмите  **«Attach Debugger to Android** Process».

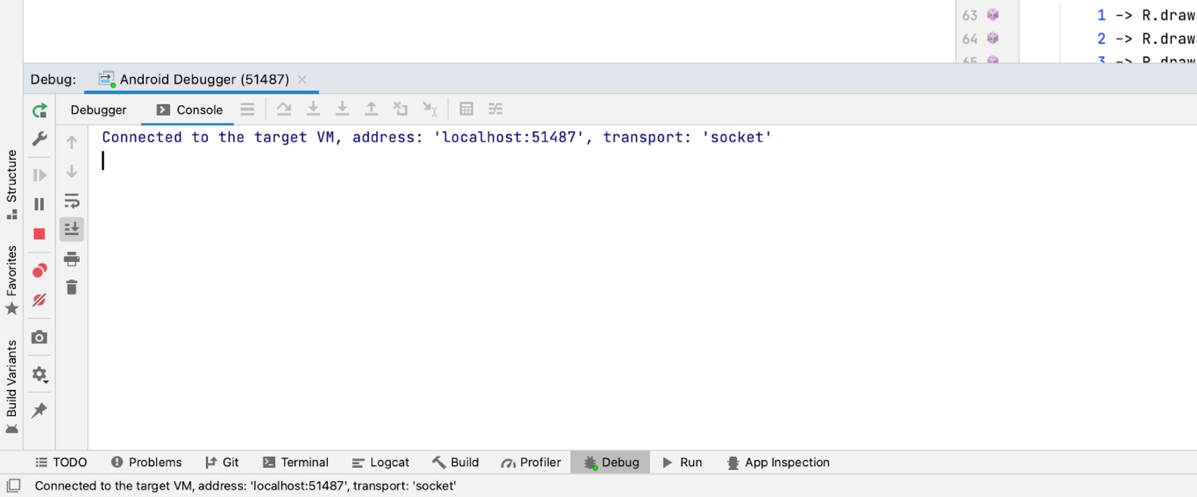


Откроется диалоговое окно **«Choose Process»** , в котором вы можете выбрать процесс, к которому вы хотите подключить отладчик.

1. Выберите com.example.diceroller и нажмите **«ОК»** .



В нижней части Android Studio появится панель **«Debug»** с сообщением о том, что отладчик подключен к целевому устройству или эмулятору.



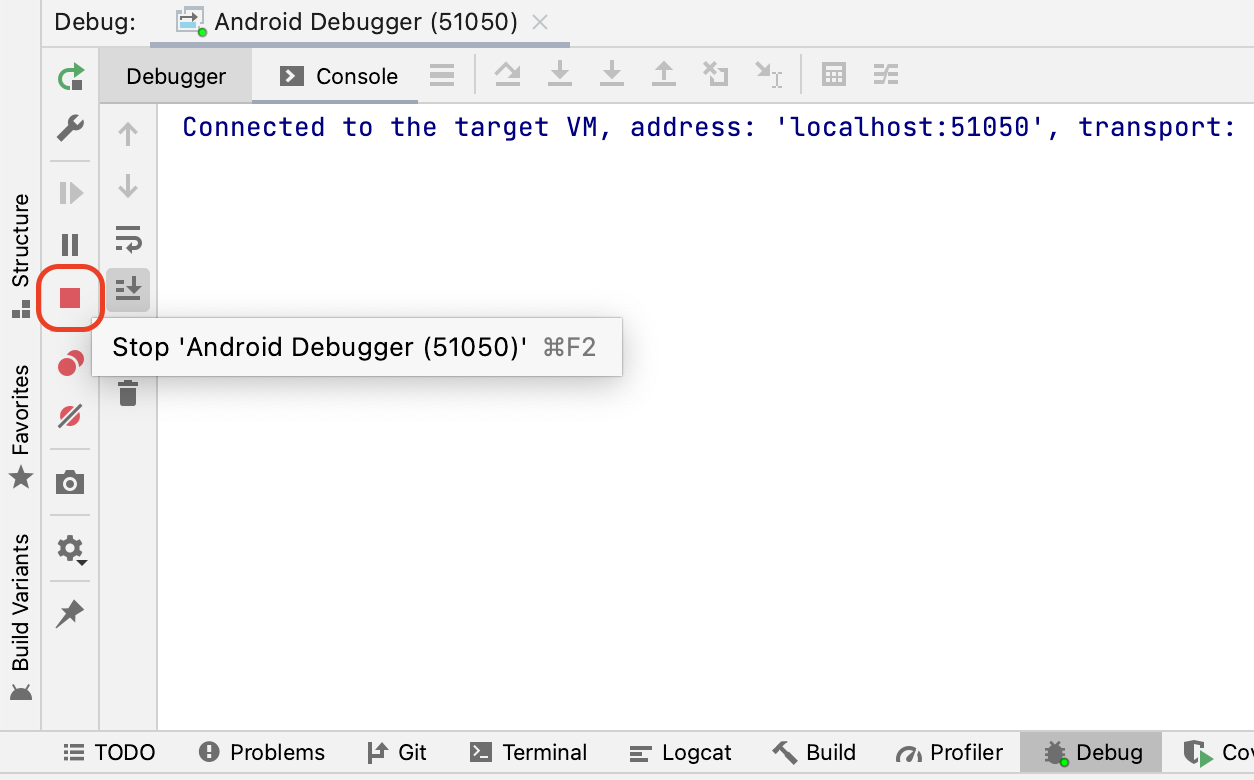
Вы подключили отладчик к своему приложению! Не беспокойтесь о том, что это означает или о том, что вы можете сделать с помощью отладчика, поскольку это будет описано позже в этой лаборатории. Далее вы узнаете, как запустить приложение с уже подключенным отладчиком.

## **Запустите приложение с помощью отладчика**

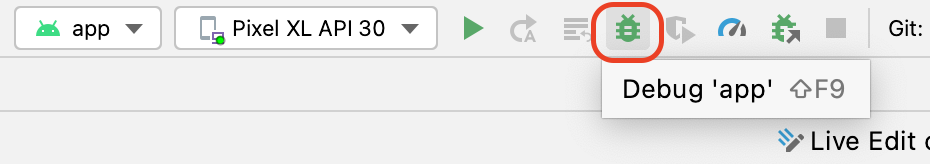
Если вы с самого начала знаете, что хотите использовать отладчик, вы можете сэкономить время, запустив приложение с помощью отладчика. Более того, если вы хотите отладить код, который запускается только при запуске приложения, вам необходимо запустить приложение с уже подключенным отладчиком.

Чтобы запустить приложение с помощью отладчика, выполните следующие действия:

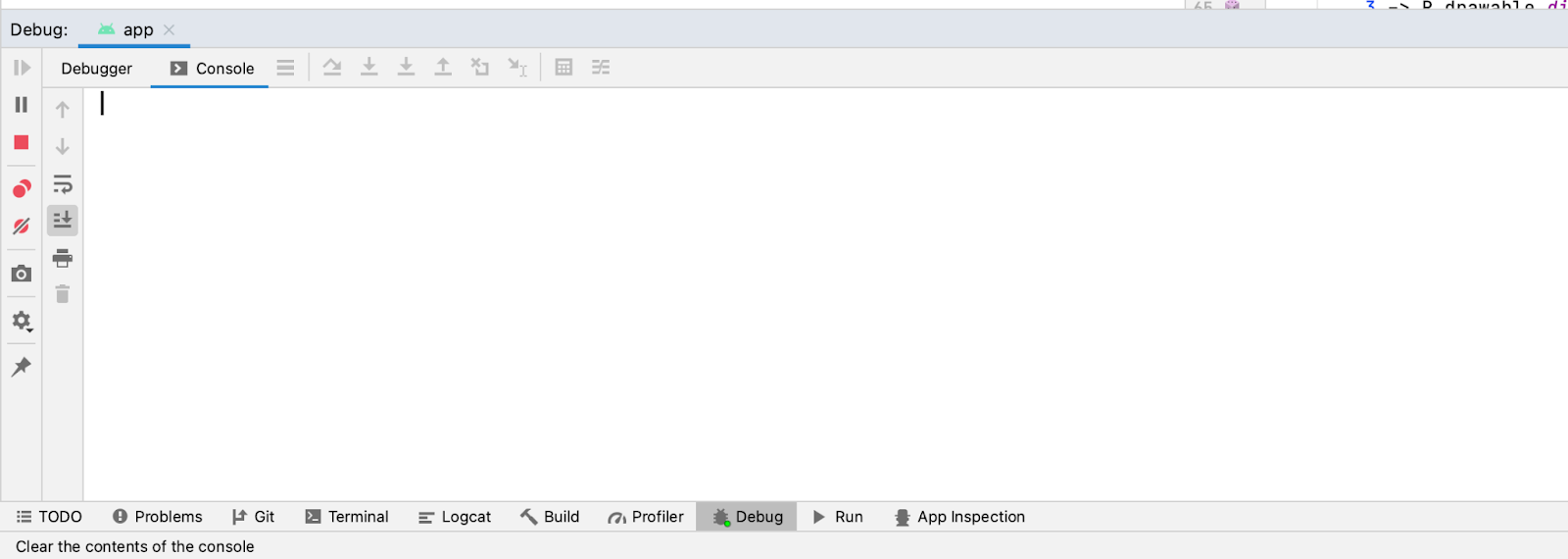
1. На панели **«Debug»** нажмите  **«Stop Android Debugger»** , а затем закройте приложение на устройстве или в эмуляторе.



1. Нажмите  **«Debug app»** .



Та же самая панель **Debug** появляется в нижней части Android Studio с некоторым выводом на консоль.



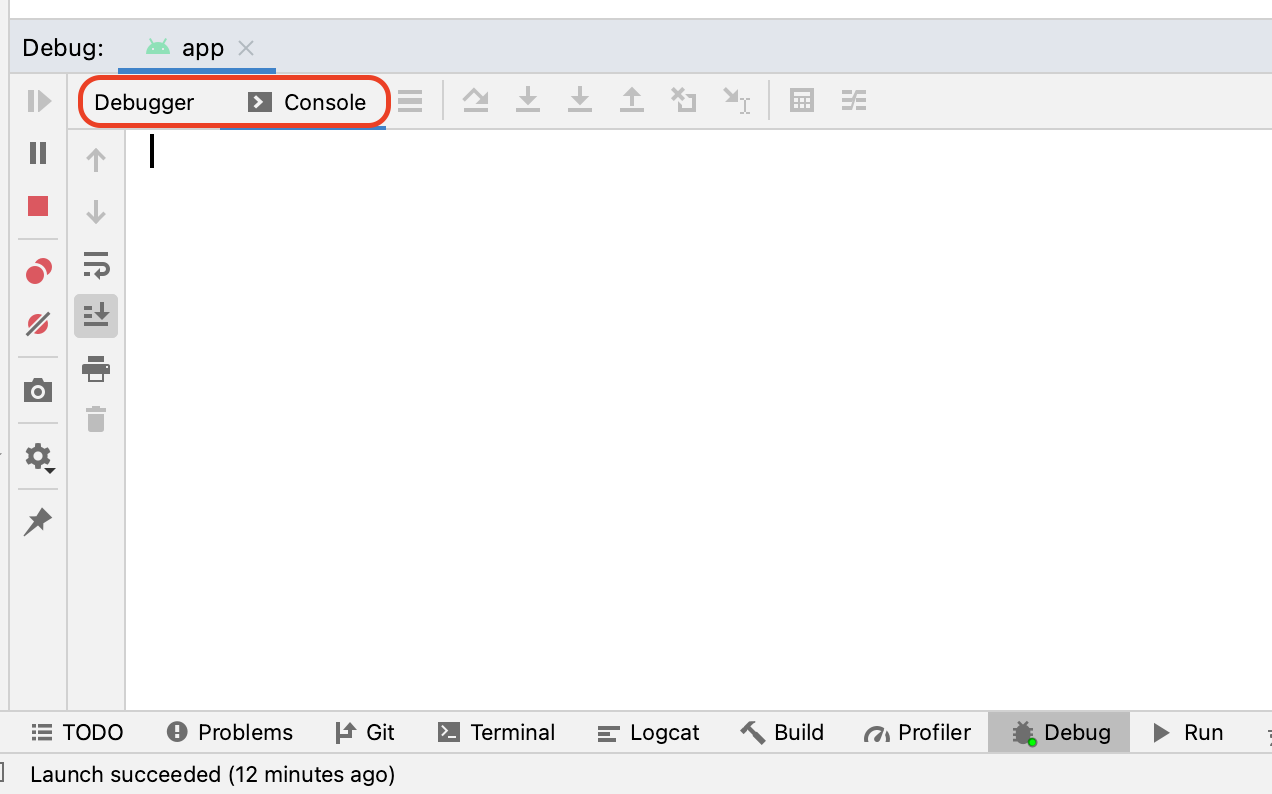
Теперь вы знаете, как запустить отладчик! Далее вы научитесь им пользоваться.

## [**5. Используйте отладчик**](https://developer.android.com/codelabs/basic-android-kotlin-compose-intro-debugger?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-2-pathway-2%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-intro-debugger#4)

## **Панель «Debug »**

**Вы, вероятно, заметили, что в верхней части панели «Debug»** имеется довольно много кнопок , но на данный момент эти кнопки мало что значат, и большинство из них неактивны и недоступны для нажатия. В этом разделе описаны наиболее часто используемые функции отладчика.

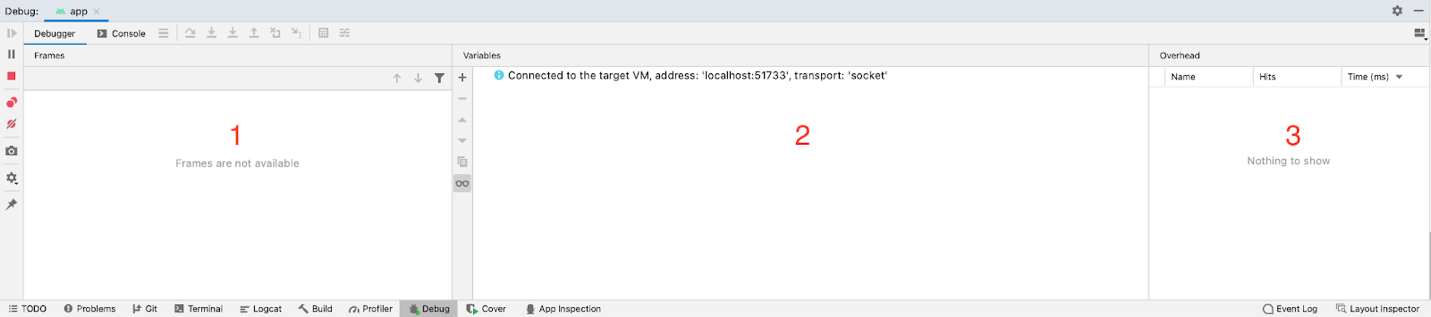
**При первом запуске отладчика на панели «Debug»** вы увидите несколько кнопок . В верхней части панели **«Debug»** вы видите кнопки **«Debugger»** и **«Console»** .



Кнопка **«Console»** отображает выходные данные приложения. Если в вашем коде есть операторы журнала, выходные данные отображаются по мере выполнения этого фрагмента кода.

Кнопка **«Debugger»** отображает три отдельные панели, которые сейчас пусты, поскольку вы не используете отладчик:

1. Frames
2. Variables
3. Overhead



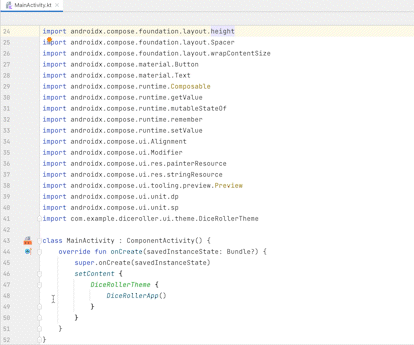
## **Используйте общие функции отладчика**

### **Установить точку останова**

Одной из основных особенностей отладчика является то, что он позволяет остановить выполнение определенной строки кода с помощью **точки останова**.

Чтобы установить точку останова в Android Studio, вам нужно перейти к определенной строке кода, а затем щелкнуть поле рядом с номером строки. Чтобы отменить установку точки останова, вам нужно щелкнуть существующую точку останова в желобе, чтобы она исчезла.

* Чтобы попробовать это самостоятельно, установите точку останова там, где установлена переменная imageResource.

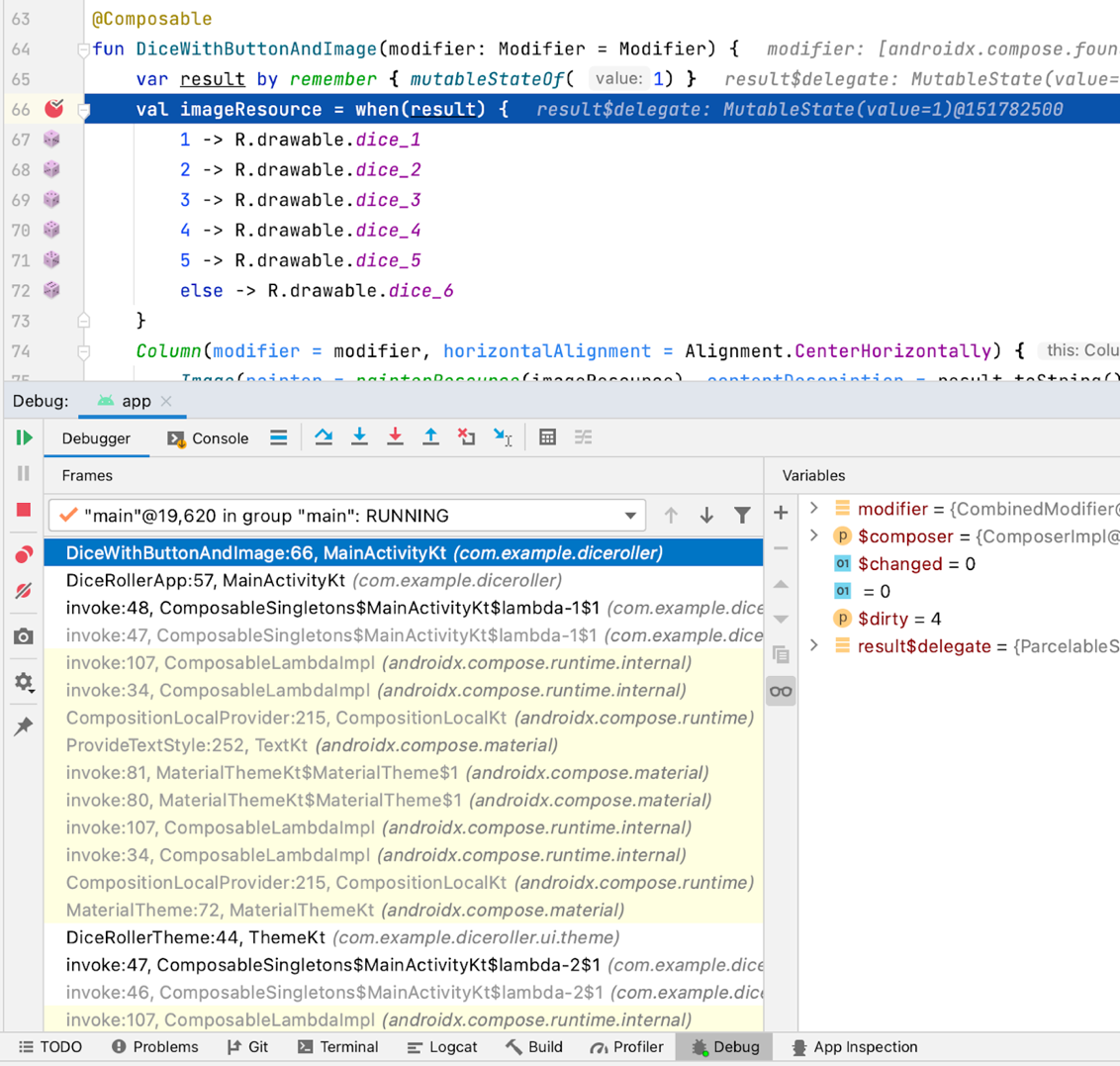


### **Используйте кнопку «Resume Program»**

Точка останова приводит к приостановке выполнения инструкции. Если выполнение кода приостанавливается с помощью отладчика, вам, вероятно, придется продолжить выполнение, чтобы продолжить работу приложения. Самый прямой способ сделать это — использовать кнопку **«Resume program»** .

Чтобы возобновить работу программы, выполните следующие действия:

1. Нажмите  **«Debug app»** . После запуска приложения вы должны увидеть что-то вроде этого изображения:



Прежде чем возобновить работу программы, важно объяснить кое-что из того, что вы видите на экране, когда отладчик приостанавливает выполнение:

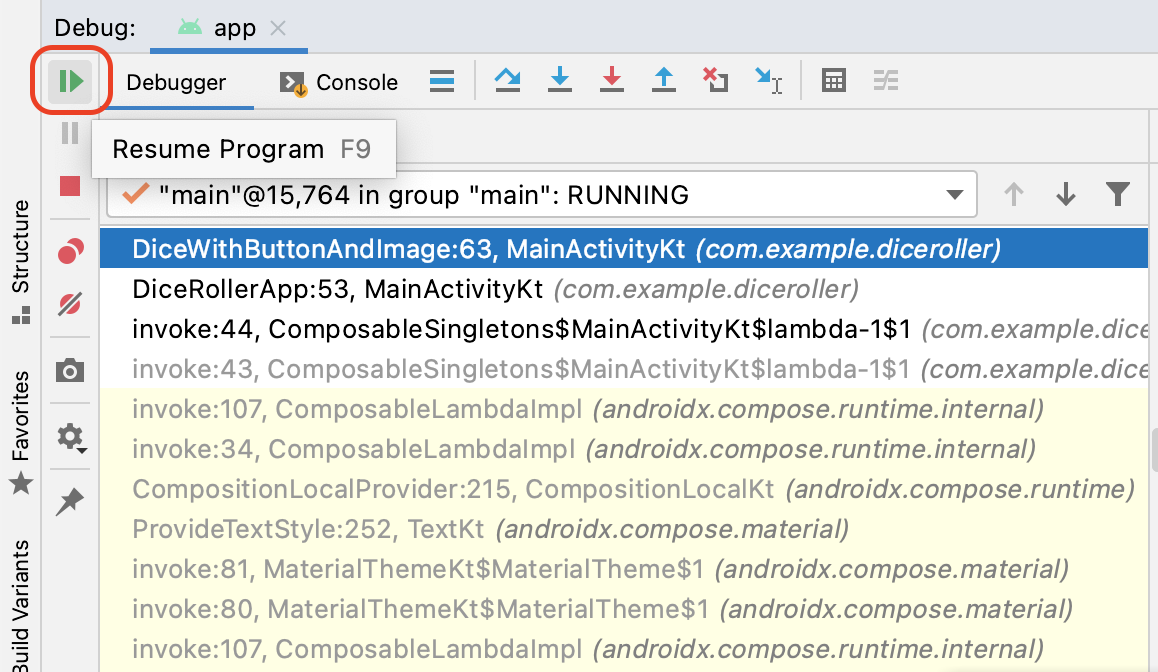
* Многие кнопки на панели **«Debug»** теперь доступны для нажатия.
* На панели **«Frames»** отображается много информации, включая выделенную ссылку на строку, где была установлена точка останова.
* На панели « **Variables»** отображается несколько элементов, но в этом приложении не так много переменных, поэтому на данный момент не так много информации, актуальной для этой лаборатории кода. Однако возможность проверки переменных является важной особенностью отладчика, поскольку она позволяет понять, что происходит в коде во время выполнения.

Если вы посмотрите на приложение на своем устройстве или в эмуляторе, вы заметите, что экран пуст, потому что приложение приостановлено на строке кода. Точнее, выполнение остановлено в точке останова, а пользовательский интерфейс еще не отрисован.

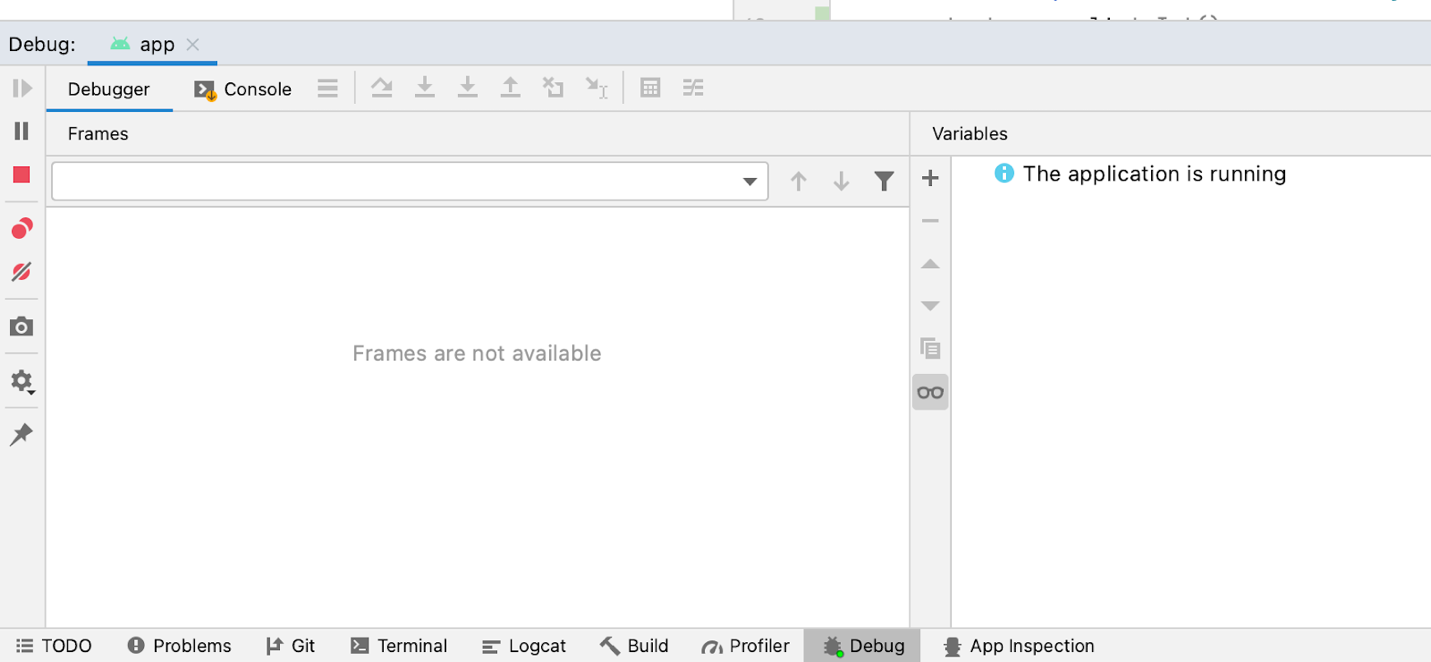
Имейте в виду, что приложение не всегда останавливается немедленно только потому, что была установлена точка останова. Это зависит от того, где вы разместите точку останова в коде. В данном случае вы поместили его в строку, которая выполняется при запуске приложения.

Важно помнить, что приложение приостанавливается только в точке останова, когда происходит попытка выполнить строку, в которой была установлена точка останова. Существует несколько способов продвинуть отладчик вперед, но сейчас вы используете кнопку **«Resume Program»** .

1. Нажмите  **«Resume Program»** .



Теперь вы должны увидеть что-то похожее на это изображение:



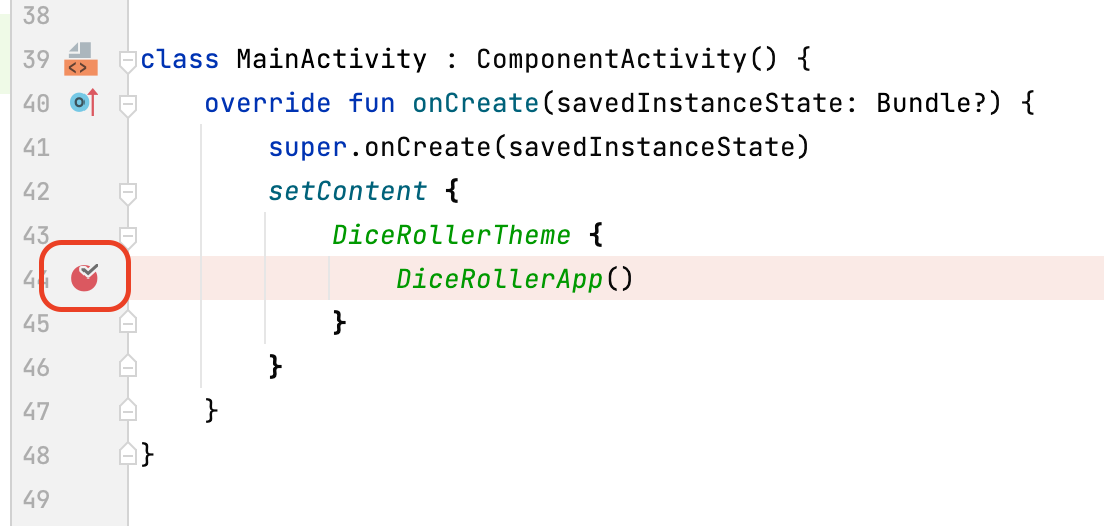
Большая часть информации исчезает, и кнопки снова становятся недоступными для нажатия. Приложение также отображается как обычно на вашем устройстве или в эмуляторе. Причина этого в том, что код больше не приостанавливается в точке останова и приложение находится в нормальном рабочем состоянии. Отладчик подключен, но он мало что делает, пока не будет предпринята попытка выполнить строку кода, для которой установлена точка останова. Оставьте эту точку останова на месте, поскольку она будет полезна в следующих примерах.

### **Используйте кнопку «Step Into»**

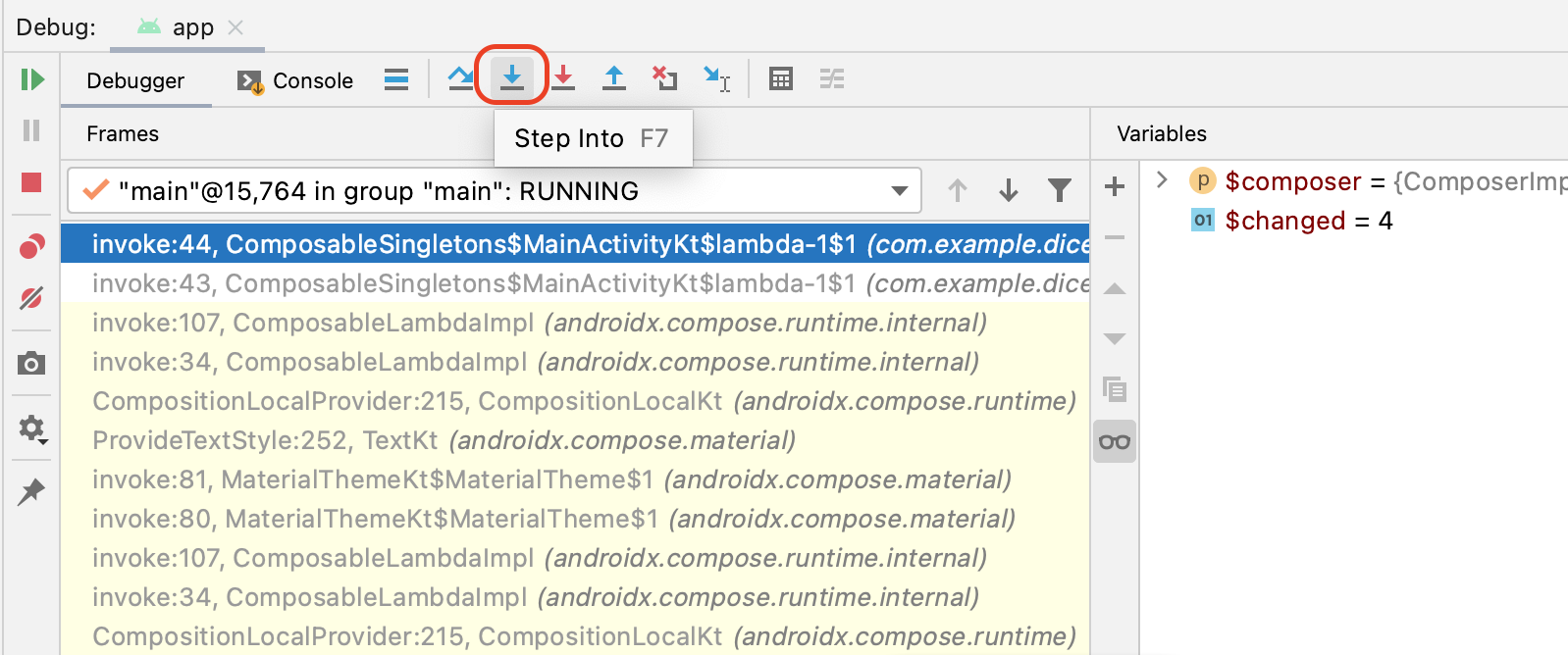
Кнопка **«Step Into»** в отладчике — это удобный способ глубже изучить код во время выполнения. Если инструкция вызывает метод или другой фрагмент кода, кнопка **«Step Into»** позволяет ввести код без необходимости переходить туда вручную перед запуском отладчика для установки точки останова.

Чтобы использовать кнопку **«Step Into»** , выполните следующие действия:

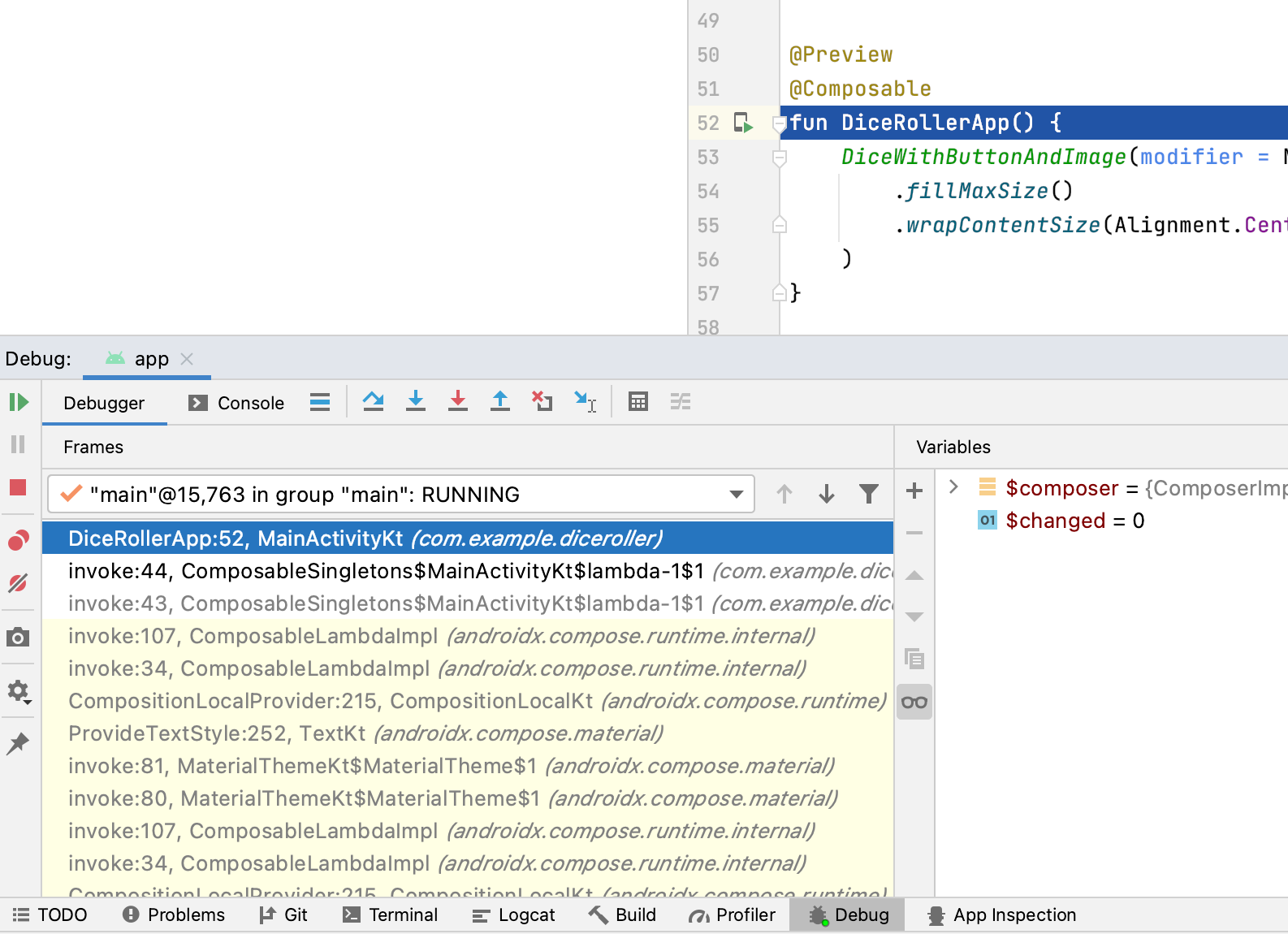
1. Создайте точку останова в теле лямбда setContent - функции onCreate() класса MainActivity, в котором вызывается функция DiceRollerApp().



1. Нажмите  **«Debug app»** , чтобы повторно запустить приложение с помощью отладчика. Выполнение приостанавливается на строке, где вызывается функция DiceRollerApp().
2. Нажмите шаг в **«Step Into»** .



Теперь строка 52 выделена, а панель **«Frames»** на панели **«Debug»** указывает, что код приостановлен в строке 52.

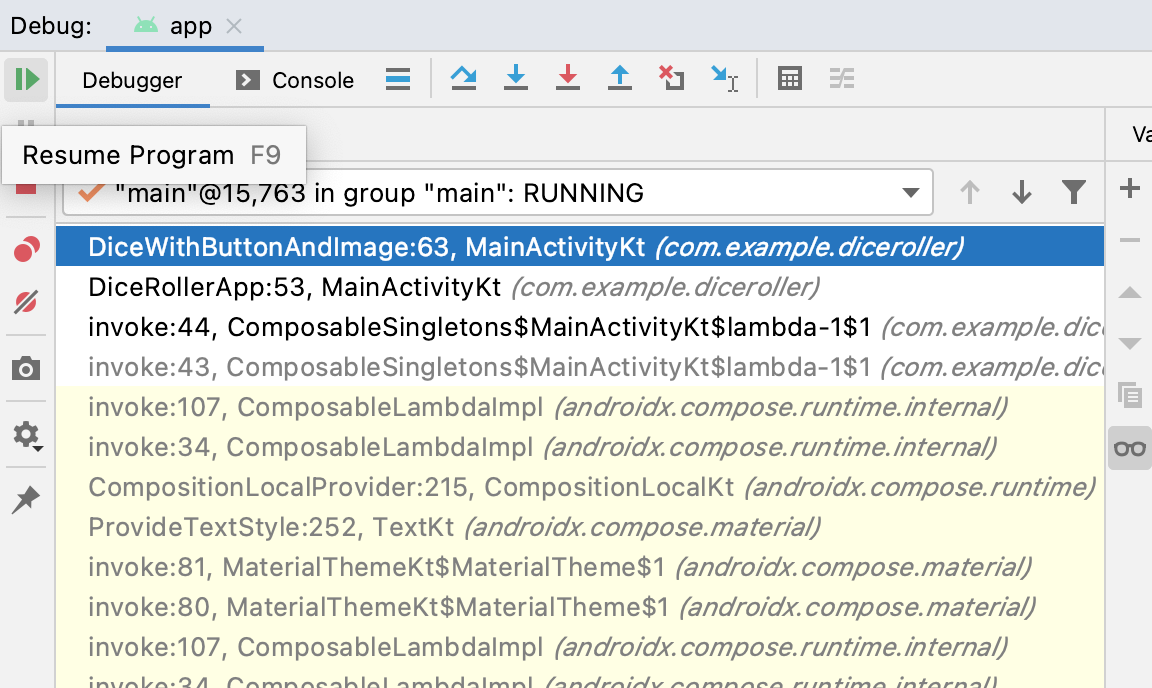


Если вы развернете панель **«Frames»** , вы увидите, что строка после выделенной строки начинается с номера строки invoke:, который на предыдущем изображении равен 44. Это то, что называется **стеком вызовов**. По сути, он показывает цепочку вызовов, которые приводят выполнение кода к текущей строке. В этом случае строка 44 содержит инструкцию, вызывающую функцию DiceRollerApp().

Когда вы нажимаете кнопку **«Step Into»**, когда отладчик останавливается в точке останова, установленной для вызова этой функции, отладчик переходит к этой функции, что приводит выполнение к строке 52, где функция объявлена. Выделенная строка указывает, где выполнение приостановлено. Если строки после выделенной строки имеют связанные с ними номера строк, это указывает путь выполнения. В данном конкретном случае отладчик указывает, что инструкция в строке 44 привела вас к строке 52.

1. Нажмите  **«Resume app»** .

Это должно привести вас к исходной точке останова, которую вы установили. Возможно, вы поймете немного больше о том, что увидели, когда остановили выполнение в первом примере. Это та самая картинка из шестого шага лекции выше :



В стеке вызовов вы можете видеть, что функция DiceWithButtonAndImage() приостановлена в строке 63 и функция была вызвана из строки 53 в функции DiceRollerApp(), которая была вызвана в строке 44. Функция стека вызовов может помочь вам понять путь выполнения. Это очень полезно, когда функция вызывается из разных мест приложения.

Кнопка **Step Into** позволяет войти в функцию и приостановить ее выполнение без установки точки останова в самой функции. В этом случае вы устанавливаете точку останова при вызове функции DiceRollerApp(). Когда вы нажимаете кнопку **«Step Into»** , выполнение функции приостанавливается DiceRollerApp().

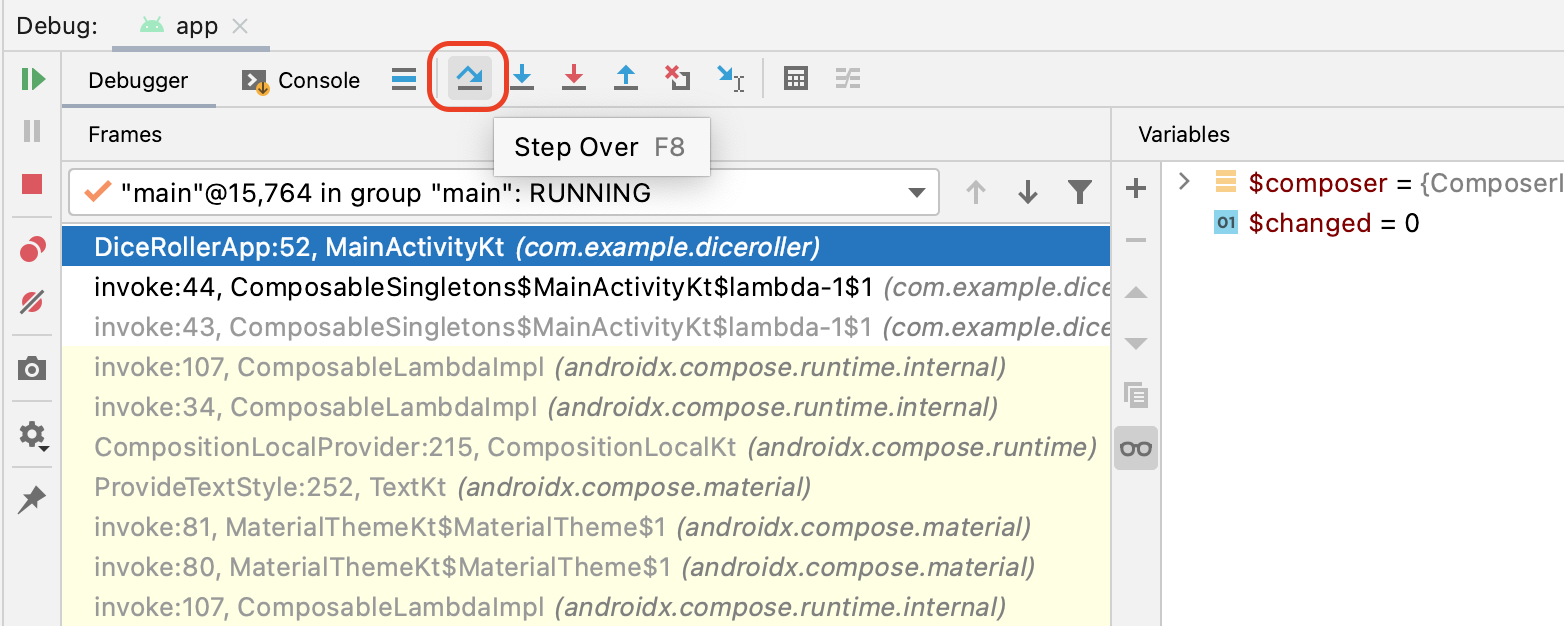
Dice Roller — довольно маленькое приложение, поскольку в нем не так много файлов, классов и функций. Когда вы работаете с более крупными приложениями, функция отладчика **Step Into** становится более полезной, поскольку она дает вам возможность углубляться в код без необходимости самостоятельно перемещаться по коду.

### **Используйте кнопку** «Step Over»

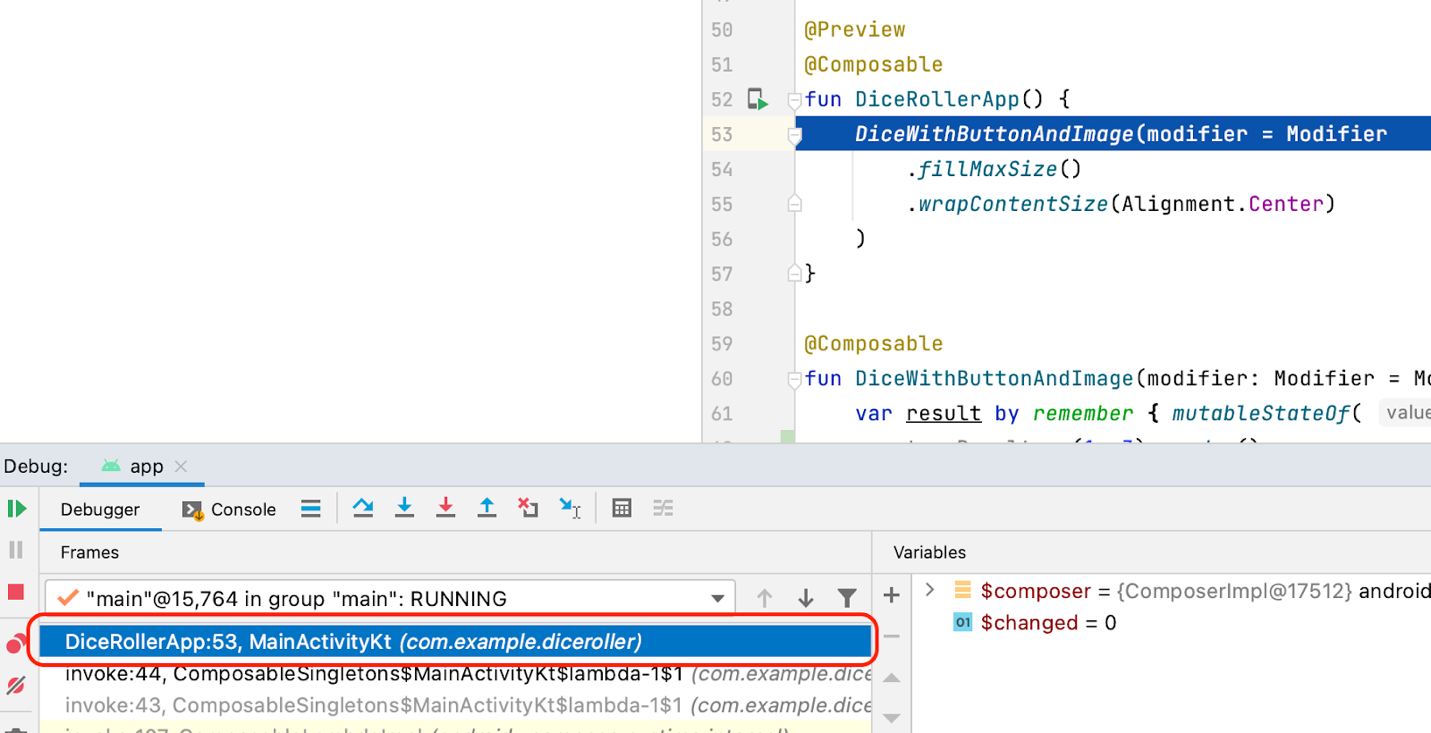
Кнопка **«Step Over»** предоставляет еще один способ пошагового выполнения кода приложения во время выполнения. Он перемещает выполнение на следующую строку кода и продвигает отладчик.

Чтобы использовать кнопку **«Step Over»** , выполните следующие действия:

* Нажмите  **«Step Over»** .



Теперь вы видите, что отладчик приостановил выполнение кода на следующей строке выполнения, то есть строке 53. Вы можете последовательно переходить к каждой строке.



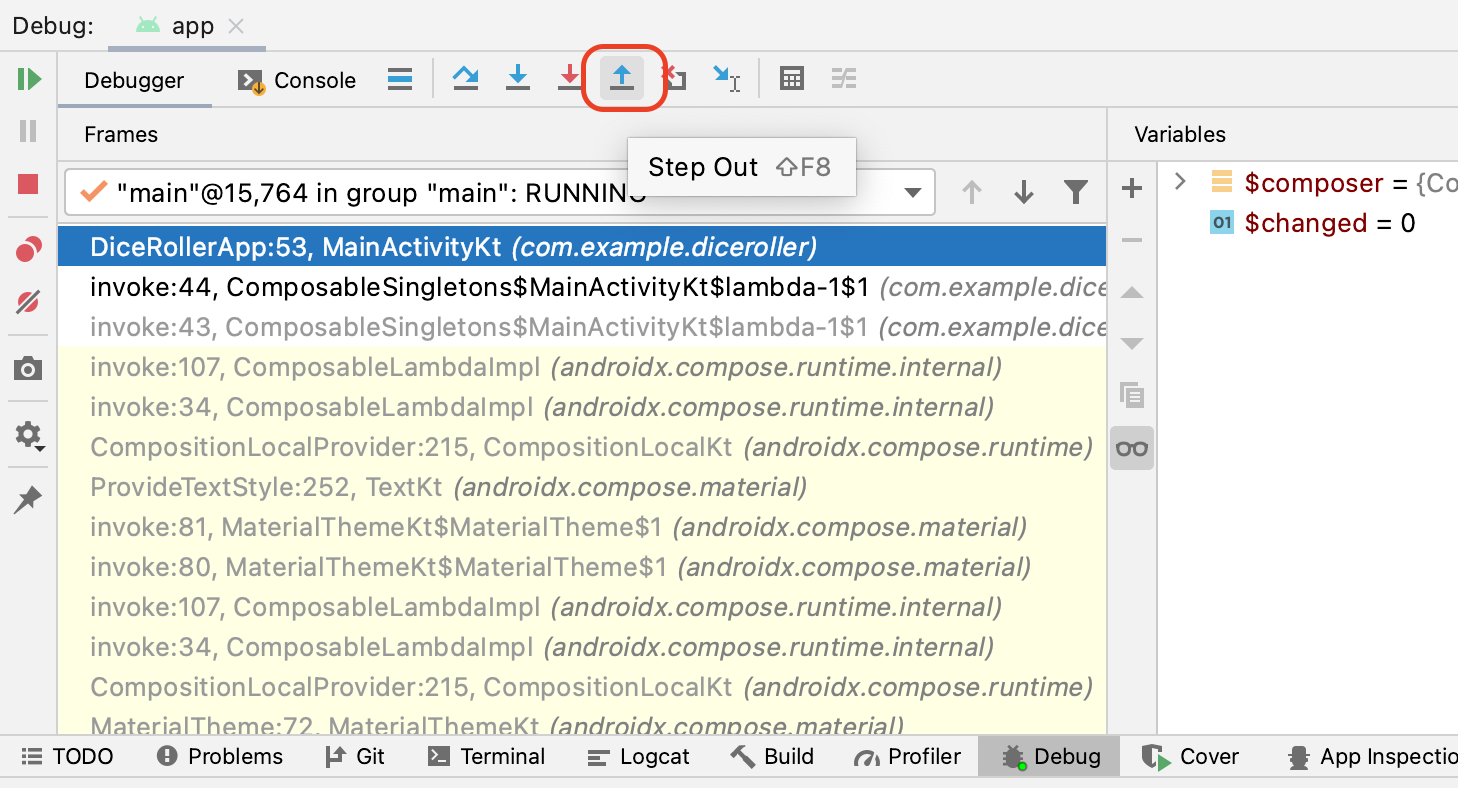
### **Используйте кнопку «Step Out»**

Кнопка **«Step Out»** выполняет противоположное действие кнопки **«Step Into»** . Вместо того, чтобы углубляться в стек вызовов, кнопка **«Step Out»** позволяет перейти к стеку вызовов вверх.

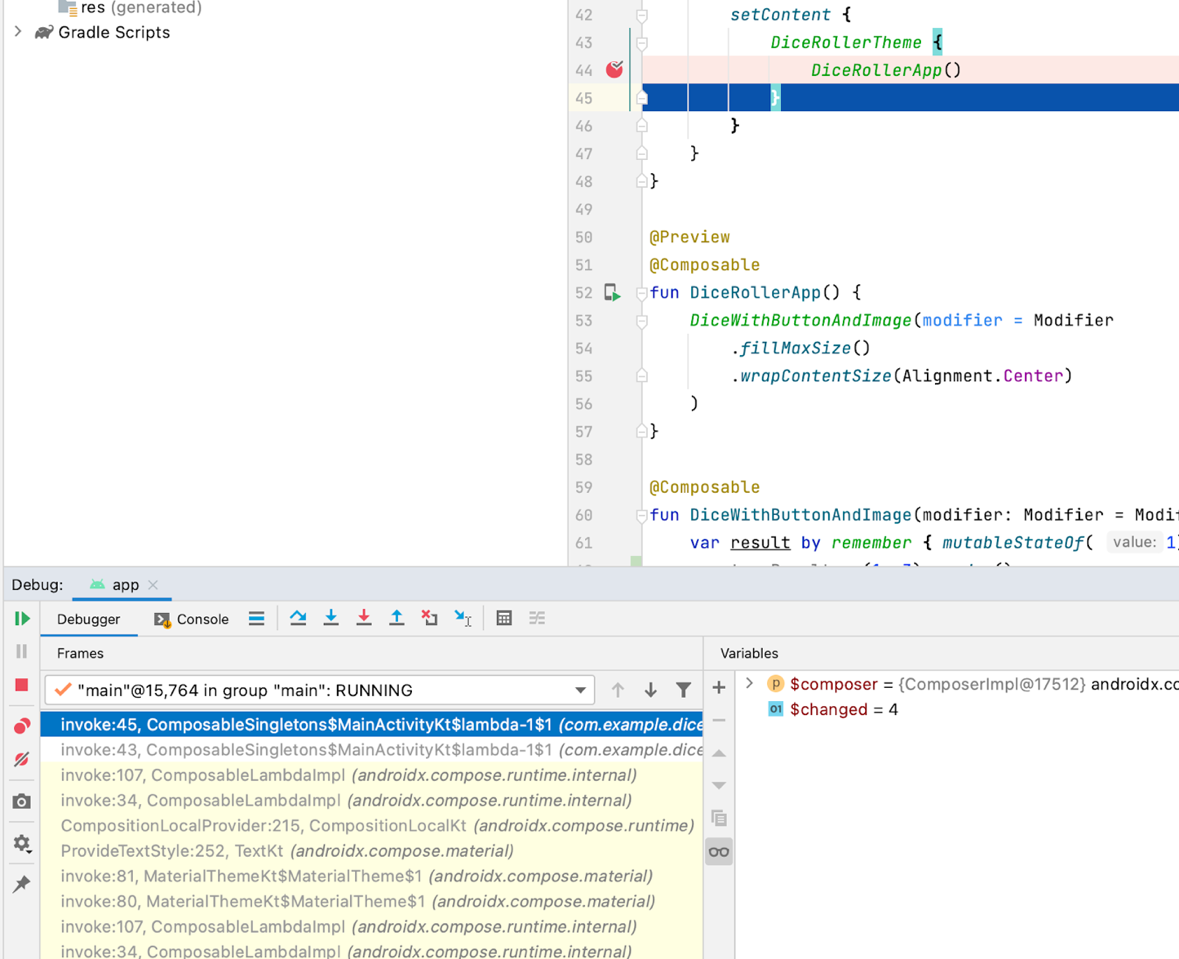
Чтобы использовать кнопку **«Step Out»** , выполните следующие действия:

1. Нажмите  **«Step Out»** .

Сможете ли вы угадать, на какой строке программа зависает?



1. Обратите внимание, что отладчик вышел из функции DiceRollerApp() и перешел на следующую строку после выполнения функции в строке 45. Выход не возвращает к строке 44, поскольку эта строка уже выполнена. Таким образом, выход приведет вас к строке 45.



Кнопка **«Step Out»** — полезный инструмент, когда вы слишком глубоко запутались в стеке вызовов методов. Это позволяет вам продвигаться вверх по стеку вызовов без необходимости пошагового выполнения всего кода для каждого метода, в который вы вошли.

## **Проверка переменных**

Ранее было краткое описание панели **«Variables»** . Рассмотрим подробнее.

Чтобы проверить переменные, выполните следующие действия:

1. Щелкните точку останова, чтобы удалить ее из того места, где вызывается функция DiceRollerApp(), но оставьте точку останова, где установлена переменная imageResource.
2. Нажмите  **«Debug app»**. Вы должны увидеть, что переменная result$delegate имеет MutableState значение 1. Это связано с тем, что когда переменная определена, ее экземпляр создается со значением mutableStateOf = 1.  Это означает, что результирующая переменная содержит состояние MutableState, которое можно изменить.

**Примечание.** Переменная result$delegate на панели **«Variable»** относится к  переменной result в коде. Это обозначение $delegate существует, потому что переменная result является делегатом remember.



1. Нажмите  **«Resume Program»** .
2. В приложении нажмите **«Roll»** . Ваш код снова приостанавливается в точке останова, и вы можете увидеть другое значение переменной result$delegate.

На этом изображении изменяемое состояние переменной result$delegate имеет значение 6. Это демонстрирует, как можно проверять переменные во время выполнения с помощью отладчика. В более полнофункциональном приложении значение переменной потенциально может привести к сбою. Используя отладчик для проверки переменных, вы можете получить более подробную информацию о сбое и исправить ошибку.

