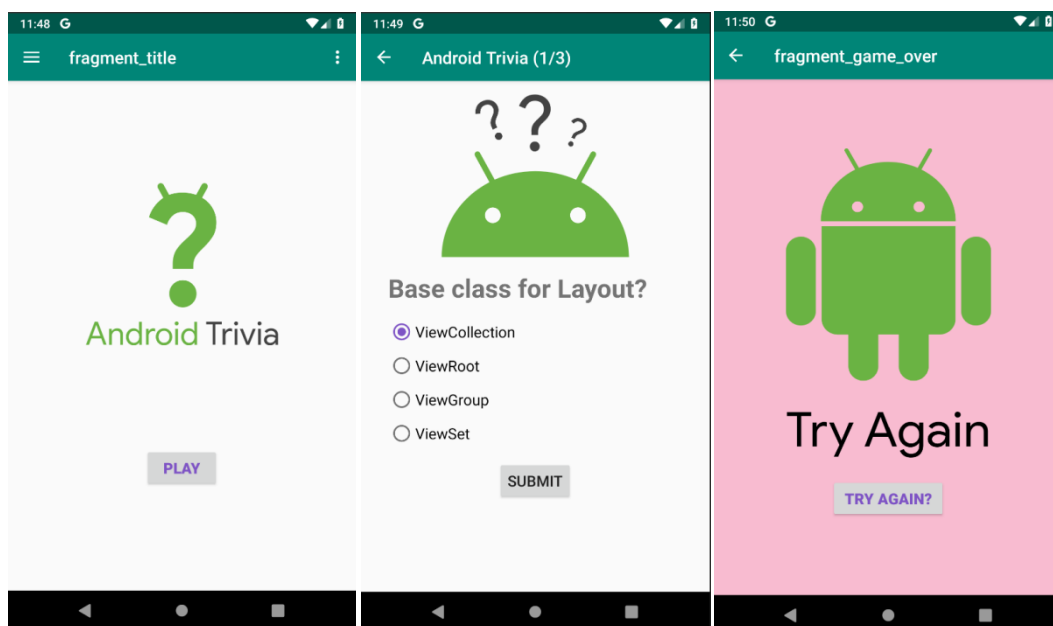


## 2. Ciąg dalszy Nawigacja po fragmentach

Aplikacja AndroidTrivia to gra, w której użytkownicy odpowiadają na pytania dotyczące rozwoju Androida. Jeśli użytkownik odpowie poprawnie na trzy pytania, wygrywa grę.

W tym ćwiczeniu aktualizujesz aplikację AndroidTrivia:

- Tworzysz graf nawigacyjny dla aplikacji (navigation graph)
- Dodajesz nawigację do ekranu tytułowego i ekranu gry.
- Połączysz ekrany z akcjami i dajesz użytkownikowi możliwość przejścia do ekranu gry, dotykając **Play**.
- Dodajesz przycisk w górę, który jest wyświetlany jako strzałka w lewo u góry niektórych ekranów oraz menu.



## 3. Zadanie: dodaj komponenty nawigacyjne do projektu

### Krok 1: Dodaj zależności nawigacyjne (navigation dependencies)

Aby korzystać z biblioteki nawigacji, musisz dodać zależności nawigacji do plików Gradle.

W okienku Project: Android otwórz folder Gradle Scripts. Kliknij dwukrotnie plik build.gradle na poziomie projektu, aby go otworzyć

1. W okienku Project: Android otwórz folder **Gradle Scripts** Kliknij dwukrotnie plik **build.gradle** na poziomie projektu.



2. Na górze pliku **build.gradle** na poziomie projektu, wraz z innymi zmiennymi **ext** dodaj zmienną dla **navigationVersion**. Aby znaleźć najnowszy numer wersji nawigacji, zobacz [Declaring dependencies](#) w dokumentacji dla programistów Androida.

```
ext {  
    ...  
    navigationVersion = '1.0.0-rc02'  
    ...  
}
```

4. W folderze **Gradle Scripts** otwórz plik **build.gradle** na poziomie modułu (module-level). Dodaj zależności dla **navigation-fragment-ktx** i **navigation-ui-ktx**, jak pokazano poniżej:

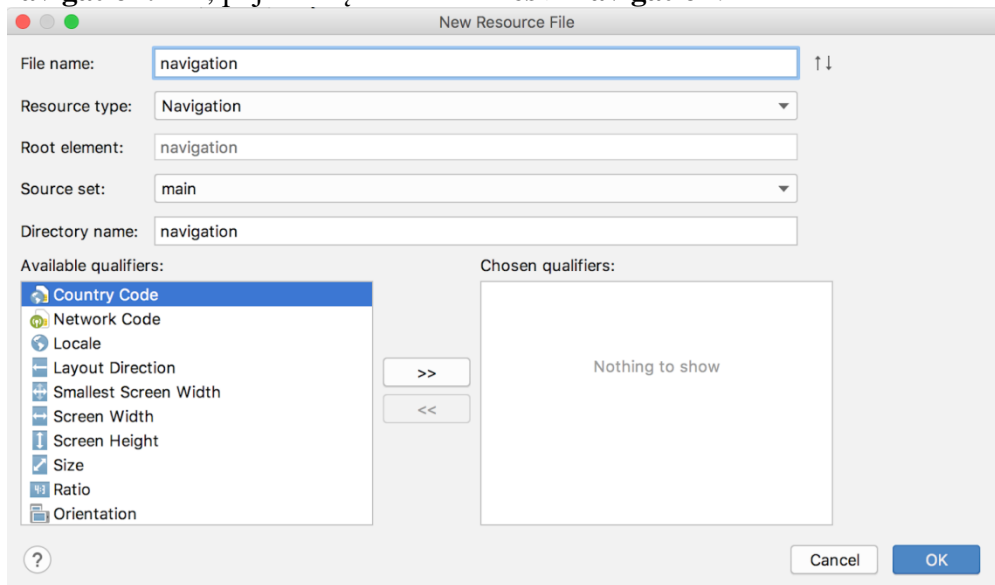
```
dependencies {  
    ...  
    implementation"android.arch.navigation:navigation-fragment-  
ktx:$navigationVersion"  
    implementation "android.arch.navigation:navigation-ui-  
ktx:$navigationVersion"  
    ...  
}
```

5. Przebuduj projekt.

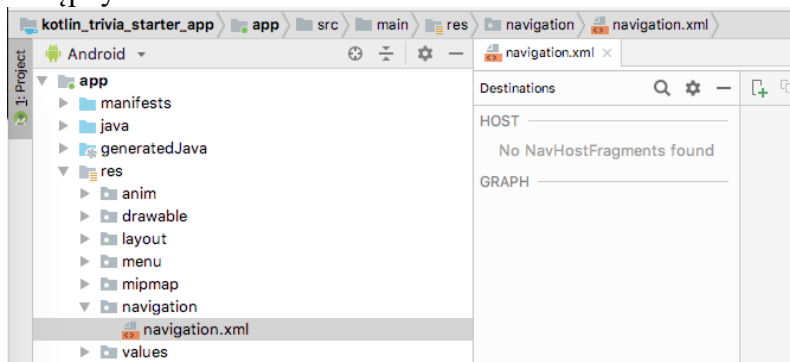
## Krok 2: Dodaj navigation graph do projektu

1. W okienku Project: widok Android, kliknij prawym przyciskiem myszy folder **res** i wybierz **New > Android Resource File**.
2. W oknie dialogowym **New Resource File** wybierz **Navigation** jako **Resource type**.
3. W polu **File name** wpisz nazwę **navigation**.

4. Upewnij się, że pole **Chosen qualifiers** jest puste, i kliknij **OK**. Nowy plik, **navigation.xml**, pojawi się w folderze **res > navigation**.



5. Otwórz plik **res > navigation > navigation.xml** i kliknij kartę **Design** aby otworzyć Edytor nawigacji [Navigation Editor](#). Zwróć uwagę na komunikat **No NavHostFragments found** w edytorze układu. Rozwiążemy ten problem w następnym zadaniu.



## 4. Zadanie: Utwórz NavHostFragment

A *navigation host fragment* działa jak host dla fragmentów na grafie nawigacji. Fragment hosta nawigacji zwykle nosi nazwę [NavHostFragment](#).

Gdy użytkownik przemieszcza się między miejscami docelowymi zdefiniowanymi na grafie nawigacyjnym, *navigation host fragment* zamienia fragmenty w razie potrzeby. Fragment tworzy również odpowiedni stos *fragment back stack* i zarządza nim.

W tym zadaniu modyfikujesz kod, aby zastąpić `TitleFragment` przez `NavHostFragment`.

1. Otwórz **res > layout > activity\_main.xml** i kliknij kartę **Text**.
2. W pliku `activity_main.xml` zmień nazwę istniejącego fragmentu (`title fragment`) na `androidx.navigation.fragment.NavHostFragment`.
3. Zmień identyfikator ID na `myNavHostFragment`.

4. The *navigation host fragment* musi wiedzieć, którego zasobu grafu nawigacji należy użyć. Dodaj atrybut `app:navGraph` i ustaw go na zasób grafu nawigacyjnego, którym jest `@navigation/navigation`.
5. Dodaj atrybut `app:defaultNavHost` i ustaw na `"true"`. Teraz ten host nawigacyjny jest domyślnym hostem i przechwytyje przycisk Wstecz systemu. *system Back button*.

Wewnątrz pliku layout `activity_main.xml` layout file, twój fragment wygląda teraz następująco:

```
<!-- The NavHostFragment within the activity_main layout -->
<fragment
    android:id="@+id/myNavHostFragment"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:navGraph="@navigation/navigation"
    app:defaultNavHost="true" />
```

## 5. Zadanie: dodaj fragmenty do grafu nawigacyjnego navigation graph

W tym zadaniu dodajesz fragment tytułu i fragment gry do wykresu nawigacyjnego aplikacji. Łączycie ze sobą fragmenty. Następnie dodajesz moduł obsługi kliknięć do przycisku **Play**, aby użytkownik mógł przechodzić z ekranu tytułowego do ekranu gry.

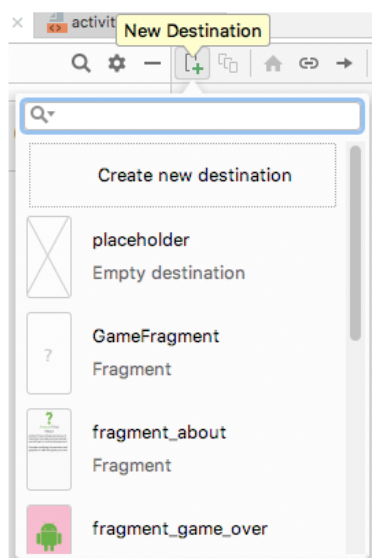
### Krok 1: Dodaj dwa fragmenty do wykresu nawigacyjnego i połącz je akcją

1. Otwórz **navigation.xml** z folderu zasobów **navigation** W Edytorze nawigacji kliknij

przycisk **New Destination**

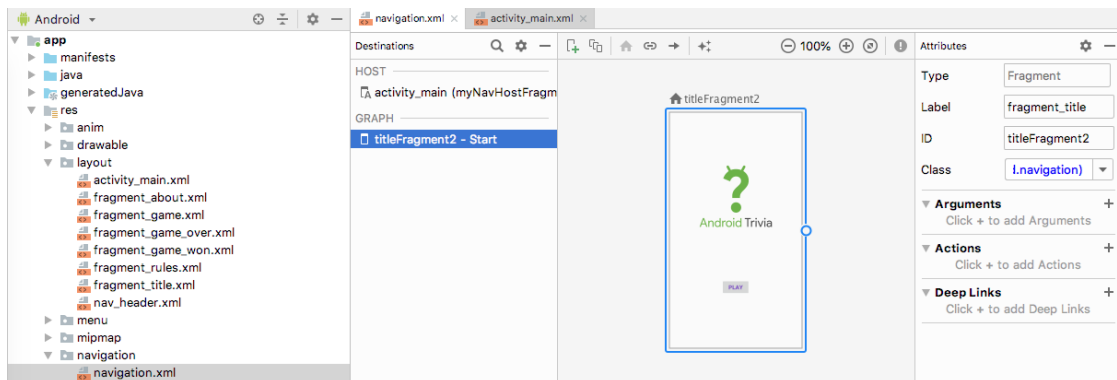
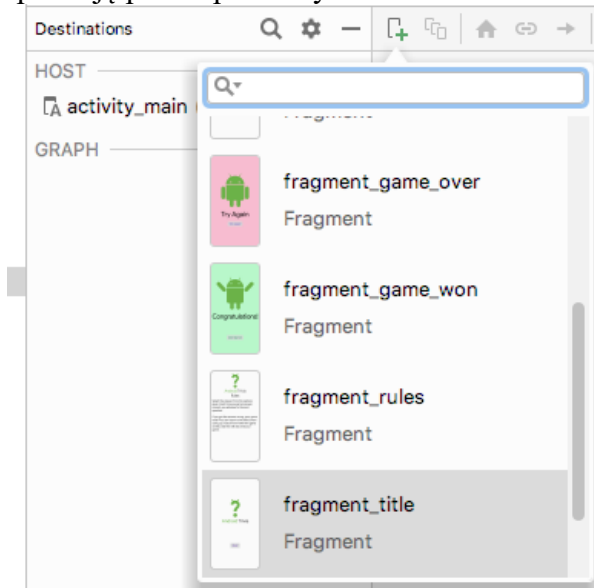


. Pojawi się lista fragmentów i activities.



2. Wybierz **fragment\_title**. Najpierw dodajesz **fragment\_title** ponieważ `TitleFragment` to miejsce, w którym użytkownicy aplikacji zaczynają, gdy otwierają

aplikację po raz pierwszy..

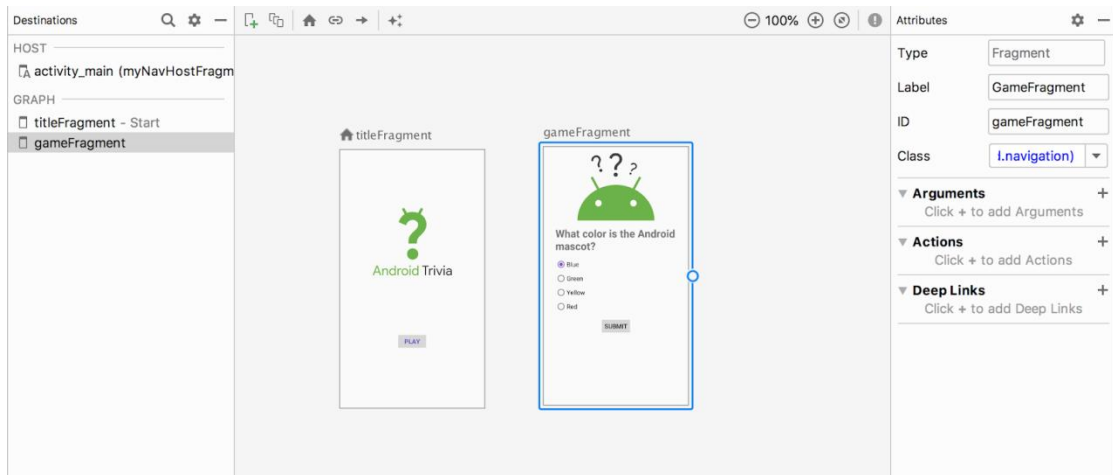


3. Użyj przycisku **New Destination** aby dodać **GameFragment**.

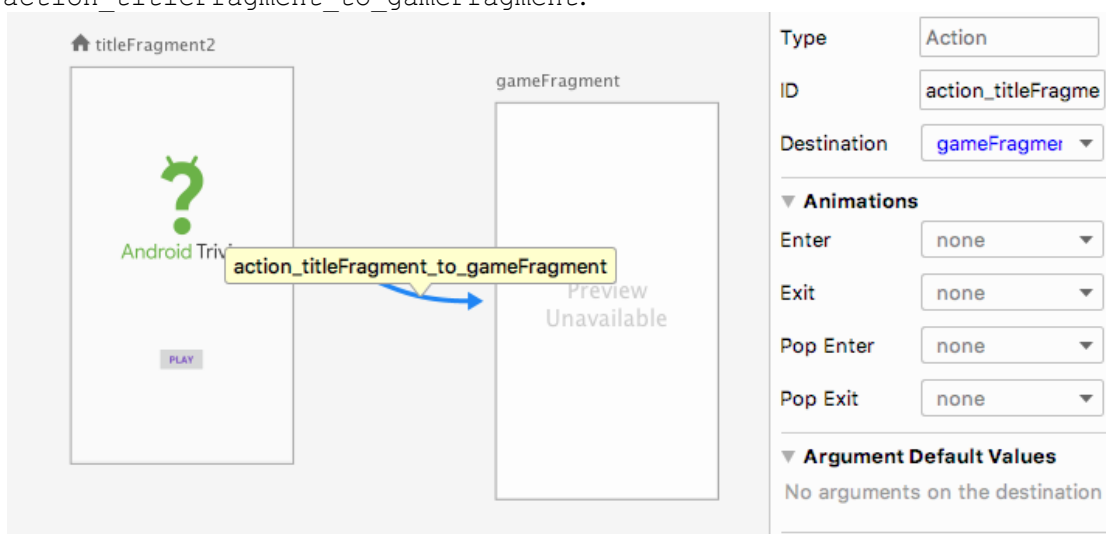
Jeśli podgląd wyświetla komunikat "Preview Unavailable" kliknij kartę **Text** aby otworzyć nawigacyjny plik XML. Upewnij się, że element `fragment` dla `gameFragment` zawiera `tools:layout="@layout/fragment_game"`, jak pokazano poniżej.

```
<!-- The game fragment within the navigation XML, complete with
tools:layout. -->
<fragment
    android:id="@+id/gameFragment"
    android:name="com.example.android.navigation.GameFragment"
    android:label="GameFragment"
    tools:layout="@layout/fragment_game" />
```

4. W edytorze układu przeciągnij fragment gry w prawo, aby nie nachodził na fragment tytułowy.



5. W podglądzie przytrzymaj wskaźnik nad title fragment. Okrągły punkt połączenia pojawia się po prawej stronie widoku fragmentu. Kliknij punkt połączenia i przeciągnij go do podglądu game fragment. Tworzona jest akcja (*action*), która łączy dwa fragmenty.
6. Aby zobaczyć atrybuty akcji, kliknij linię łączącą dwa fragmenty. W panelu **Attributes** sprawdź, czy identyfikator akcji jest ustawiony na `action_titleFragment_to_gameFragment`.



## Krok 2: Dodaj moduł obsługi kliknięć do przycisku Play

The Fragment tytułowy jest połączony z fragmentem gry przez akcję. Teraz chcesz, aby przycisk **Play** na ekranie tytułowym nawigował użytkownika do ekranu gry.

1. W Android Studio otwórz plik `TitleFragment.kt`. Wewnątrz metody `onCreateView()` dodaj następujący kod przed instrukcją `return`:

```
binding.playButton.setOnClickListener{
```

2. Wewnątrz `setOnClickListener`, dodaj kod, aby uzyskać dostęp do przycisku **Play** przez klasę wiążącą (binding class) oraz nawigację do fragmentu gry:

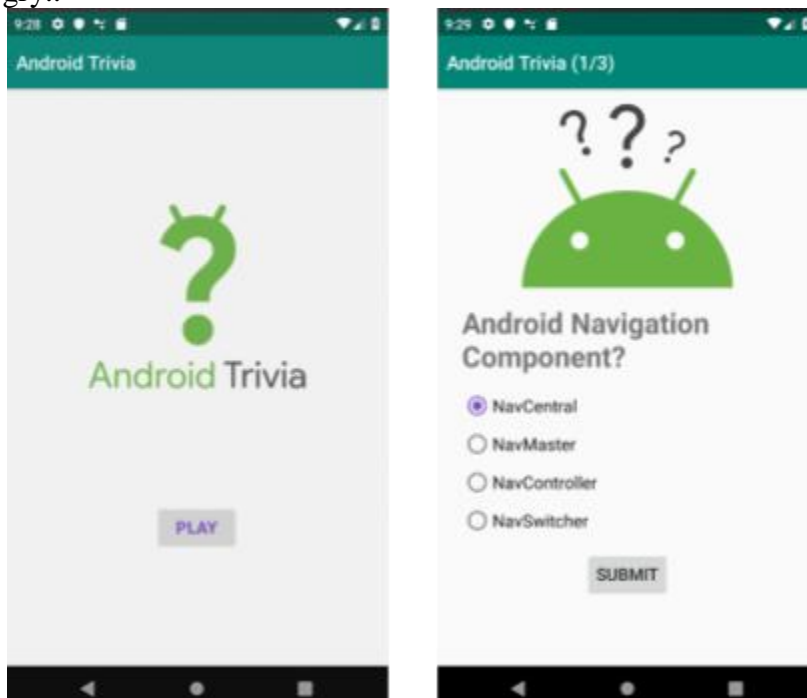
```
//The complete onClickListener with Navigation
binding.playButton.setOnClickListener { view : View ->

view.findNavController().navigate(R.id.action_titleFragment_to_gameFragment
)
}
}
```

3. Zbuduj aplikację i upewnij się, że ma wszystkie potrzebne importy. Na przykład może być konieczne dodanie następującego wiersza do pliku `TitleFragment.kt`:

```
import androidx.navigation.findNavController
```

4. Uruchom aplikację i dotknij przycisku **Play** na ekranie tytułowym. Otwiera się ekran gry..



## 6. Zadanie: dodaj nawigację warunkową (conditional navigation)

W tym kroku dodajesz nawigację warunkową, która jest nawigacją dostępną tylko dla użytkownika w określonych kontekstach. Typowym przypadkiem użycia dla nawigacji warunkowej jest sytuacja, gdy aplikacja ma inny przepływ, w zależności od tego, czy użytkownik jest zalogowany.

Twoja aplikacja ma inny przypadek: Twoja aplikacja przejdzie do innego fragmentu na podstawie tego, czy użytkownik poprawnie odpowie na wszystkie pytania.

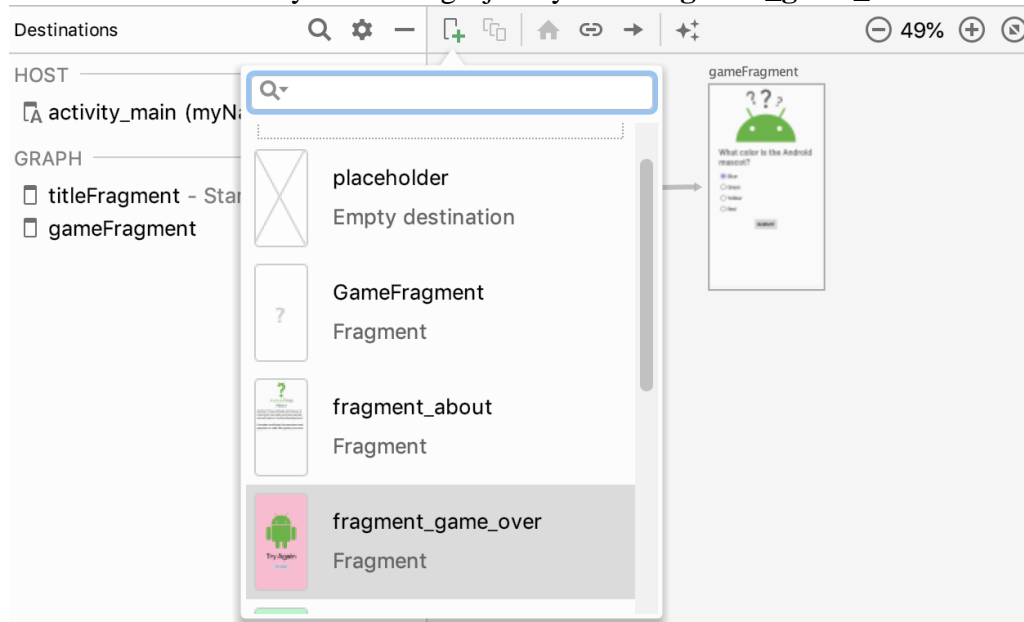
Kod startowy zawiera dwa fragmenty do użycia w nawigacji warunkowej:

- `GameWonFragment` przenosi użytkownika do ekranu z komunikatem "Congratulations!".
- `GameOverFragment` przenosi użytkownika do ekranu z komunikatem "Try Again".


## Krok 1: Dodaj GameWonFragment i GameOverFragment do navigation graph

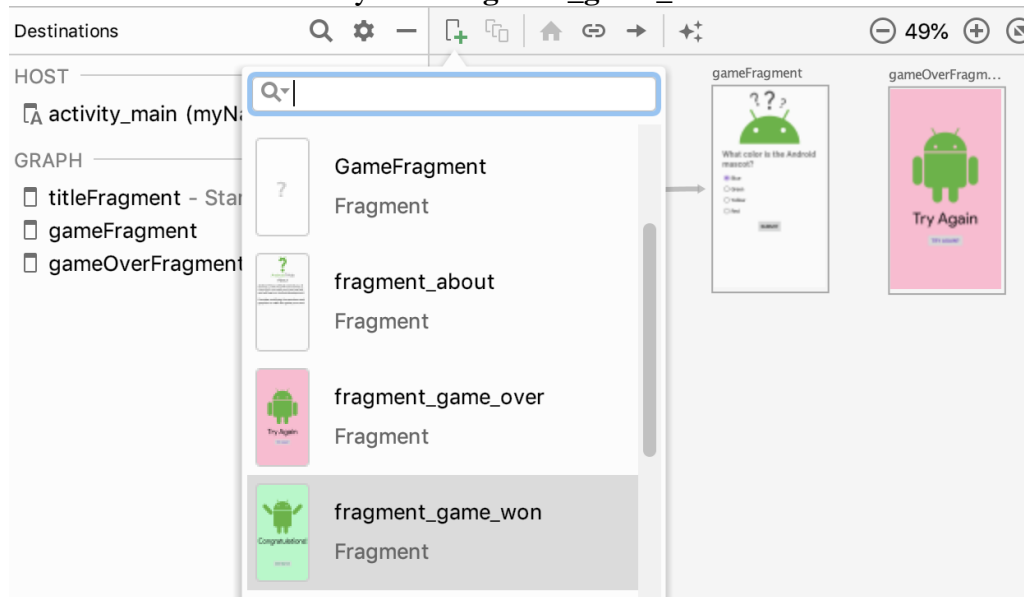
1. Otwórz plik `navigation.xml` który znajduje się w folderze nawigacji. `Navigation`.
2. Aby dodać fragment game-over do wykresu nawigacji, kliknij przycisk **New**

**Destination**  w Edytorze nawigacji i wybierz **fragment\_game\_over**.



3. W obszarze podglądu edytora przeciągnij fragment nad grą na prawo od fragmentu gry, aby nie nakładały się na siebie. Pamiętaj o zmianie atrybutu ID fragmentu Game-Over na `gameOverFragment`.
4. Aby dodać game-won fragment do wykresu nawigacyjnego, kliknij przycisk **New**

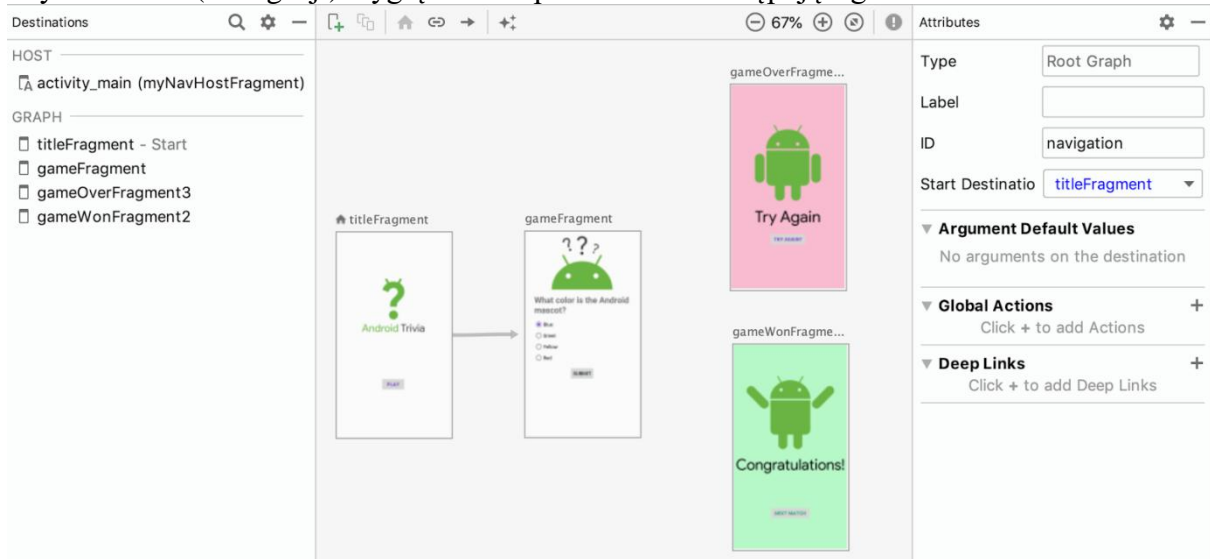
**Destination**  i wybierz **fragment\_game\_won**.



5. Przeciągnij fragment game-won poniżej fragmentu game-over aby nie nakładały się na siebie. Pamiętaj, aby nazwać atrybut **ID** jako `gameWonFragment`.



Edytor układu (nawigacji) wygląda teraz podobnie do następującego zrzutu ekranu:

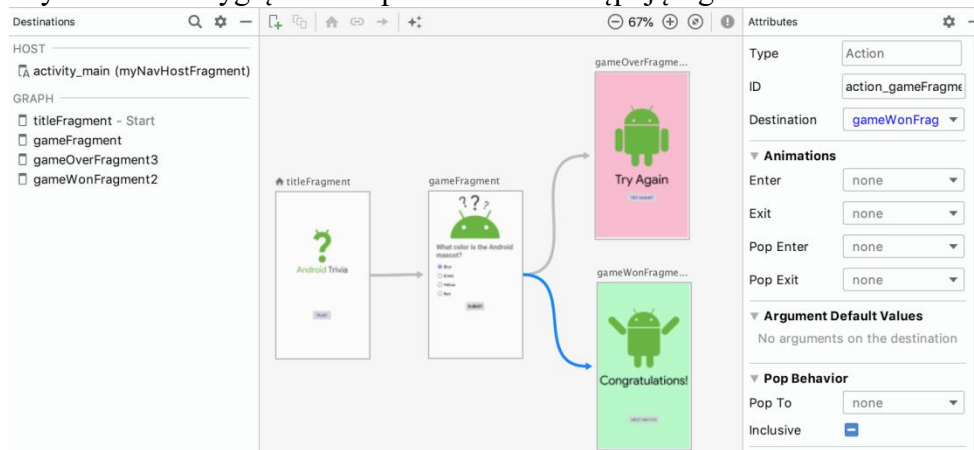


## Krok 2: Połącz fragment gry z fragmentem wyniku gry game-result fragment

W tym kroku łączysz fragment gry jednocześnie z fragmentem wygranej gry i przegranej gry.

1. W obszarze podglądu edytora układu przytrzymaj wskaźnik nad fragmentem gry, aż pojawi się okrągły punkt połączenia.
2. Kliknij punkt połączenia i przeciągnij go do fragmentu gry. Pojawi się niebieska linia połączenia reprezentująca akcję, która teraz łączy fragment gry z fragmentem rozgrywki.
3. W ten sam sposób utwórz akcję łączącą fragment gry z fragmentem wygranej gry.

Edytor układu wygląda teraz podobnie do następującego zrzutu ekranu:



4. W podglądzie przytrzymaj wskaźnik nad linią łączącą fragment gry z fragmentem wygranej gry. Zauważ, że identyfikator akcji został przypisany automatycznie.

## Krok 3: Dodaj kod, aby przechodzić od jednego fragmentu do następnego

`GameFragment` to fragment class która zawiera pytania i odpowiedzi dotyczące gry. Klasa obejmuje również logikę, która określa, czy użytkownik wygrywa, czy przegrywa. Trzeba dodać nawigację warunkową w klasie `GameFragment` w zależności od tego, czy gracz wygra, czy przegra.

1. Otwórz plik `GameFragment.kt` Metoda `onCreateView()` określa warunek `if/else` który określa, czy gracz wygrał, czy przegrał:

```
binding.submitButton.setOnClickListener
@Suppress("UNUSED_ANONYMOUS_PARAMETER")
{
    ...
    // answer matches, we have the correct answer.
    if (answers[answerIndex] == currentQuestion.answers[0]) {
        questionIndex++
        // Advance to the next question
        if (questionIndex < numQuestions) {
            currentQuestion = questions[questionIndex]
            setQuestion()
            binding.invalidateAll()
        } else {
            // We've won! Navigate to the gameWonFragment.
        }
    } else {
        // Game over! A wrong answer sends us to the
gameOverFragment.
    }
}
```

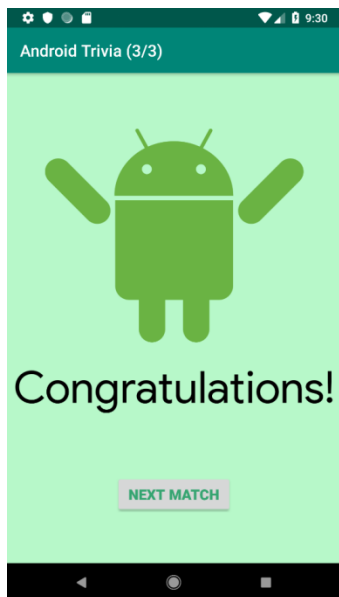
2. W przypadku wygranej dodaj następujący kod, który przechodzi do `gameWonFragment`. Upewnij się, że nazwa akcji (`action_gameFragment_to_gameWonFragment` in this example) dokładnie odpowiada ustawieniom w pliku `navigation.xml`.

```
// We've won! Navigate to the gameWonFragment.
view.findNavController()
    .navigate(R.id.action_gameFragment_to_gameWonFragment)
```

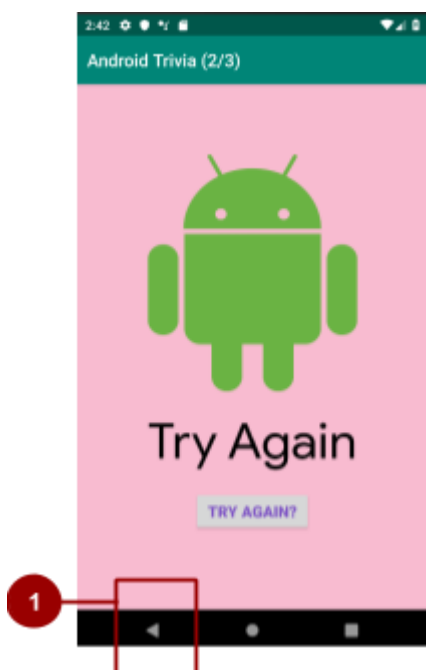
3. W przypadku przegranej przechodzimy do `gameOverFragment`:

```
// Game over! A wrong answer sends us to the gameOverFragment.
view.findNavController().
    navigate(R.id.action_gameFragment_to_gameOverFragment)
```

4. Uruchom aplikację i zagraj w grę, odpowiadając na pytania. Jeśli odpowiesz poprawnie na wszystkie trzy pytania, aplikacja przejdzie do `GameWonFragment`.



Jeśli jakieś pytanie się nie powiedzie, aplikacja natychmiast przejdzie do `GameOverFragment`.

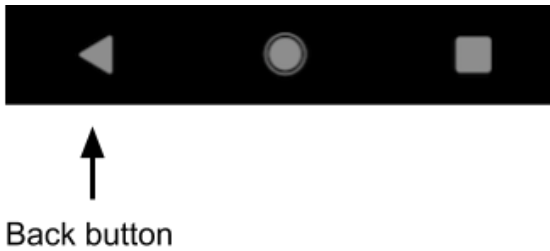


Przycisk Wstecz systemu Android jest pokazany jako 1 na zrzucie ekranu powyżej. Jeśli użytkownik naciśnie przycisk Wstecz w fragmencie wygranej w grze lub fragmencie po zakończeniu gry, aplikacja przejdzie do ekranu pytań. Lepiej by było gdyby przycisk Wstecz prowadził do ekranu tytułowego aplikacji. Zmienimy to zaraz.

## 7. Zadanie: Zmień miejsce docelowe przycisku Wstecz

System Android śledzi, dokąd użytkownicy poruszają się na urządzeniu z Androidem. Za każdym razem, gdy użytkownik przechodzi do nowego miejsca docelowego na urządzeniu, Android dodaje to miejsce docelowe do stosu (back stack). Gdy użytkownik naciśnie przycisk Wstecz, aplikacja przechodzi do miejsca docelowego znajdującego się na górze stosu.

Domyślnie górna część stosu to ekran ostatnio oglądany przez użytkownika. Przycisk Wstecz jest zwykle lewym przyciskiem u dołu ekranu, jak pokazano poniżej. (Dokładny wygląd przycisku Wstecz jest różny na różnych urządzeniach).



Do tej pory pozwalałeś kontrolerowi nawigacyjnemu obsługiwać stos za Ciebie. Gdy użytkownik nawiguje do miejsca docelowego w aplikacji, Android dodaje to miejsce docelowe do stosu (back stack).

W aplikacji `AndroidTrivia`, gdy użytkownik naciśnie przycisk Wstecz na ekranie `GameOverFragment` lub `GameWonFragment` ponownie pojawia się `GameFragment`. Ale nie chcesz wysłać użytkownika do `GameFragment`, ponieważ gra się skończyła. Użytkownik może ponownie uruchomić grę, ale lepszym doświadczeniem byłoby powrót do ekranu tytułowego.

Akcja nawigacji może modyfikować back stack. W tym zadaniu zmieniasz akcję, która nawiguje z `GameFragment` taka by usunąć the `GameFragment` ze stosu. Gdy użytkownik wygrywa lub przegrywa grę, po naciśnięciu przycisku Wstecz aplikacja pomija `GameFragment` i wraca do `TitleFragment`.

## Krok 1: Ustaw zachowanie pop dla akcji nawigacyjnych

W tym kroku zarządzasz tylnym stosem, tak że gdy użytkownik znajduje się na ekranie `GameWon` or `GameOver` naciśnięcie przycisku Wstecz powoduje powrót do ekranu tytułowego.

Zarządzamy tylnym stosem, ustawiając zachowanie „pop” dla akcji łączących fragmenty:

- Atrybut `popUpTo` akcji "wykasuje" stos do określonego miejsca docelowego przed nawigacją. (Miejsca docelowe są usuwane z tylnego stosu.)
- Jeśli atrybut `popUpToInclusive` ma wartość `false` lub nie jest ustawiony, `popUpTo` usuwa miejsca docelowe do określonego miejsca docelowego, ale pozostawia określone miejsce docelowe na stosie.
- Jeśli parametr `popUpToInclusive` ma wartość `true`, atrybut `popUpTo` usuwa wszystkie miejsca docelowe do podanego miejsca docelowego włącznie.
- Jeśli `popUpToInclusive` jest `true` a `popUpTo` jest ustawione na lokalizację początkową aplikacji, akcja usuwa wszystkie miejsca docelowe aplikacji z tylnego stosu. Przycisk Wstecz usuwa użytkownika z aplikacji.

W tym kroku ustawiasz atrybut `popUpTo` dla dwóch akcji utworzonych w poprzednim zadaniu. Robi się to za pomocą pola **Pop To** w panelu **Attributes** edytora układu.

1. Otwórz plik `navigation.xml` w folderze **res > navigation**. Jeśli wykres nawigacyjny nie pojawia się w edytorze układu, kliknij kartę **Design**.
2. Wybierz akcję do nawigacji z `gameFragment` do `gameOverFragment`. (W obszarze podglądu akcja jest reprezentowana przez niebieską linię, która łączy dwa fragmenty.)
3. W panelu **Attributes** ustaw **Pop To** na `gameFragment`. Zaznacz pole wyboru **Inclusive**.



To ustawia atrybuty `popUpTo` i `popUpToInclusive` w pliku XML. Atrybuty nakazują komponentowi nawigacyjnemu usuwanie fragmentów z tylnego stosu aż do `GameFragment` włącznie. (Ma to taki sam efekt, jak ustawienie pola **Pop To** na `titleFragment` i wyczyszczenie checkboxa **Inclusive**)

4. Wybierz akcję do nawigacji z `gameFragment` do `gameWonFragment`. I zrób to samo co poprzednio
5. Uruchom aplikację i zagraj w grę, a następnie naciśnij przycisk Wstecz. Bez względu na to, czy wygrasz, czy przegrasz, przycisk Wstecz przeniesie Cię z powrotem do `TitleFragment`.

## Krok 2: Dodaj więcej akcji nawigacyjnych i dodaj moduły obsługi onClick

Twoja aplikacja ma obecnie następujący przepływ *user flow*:

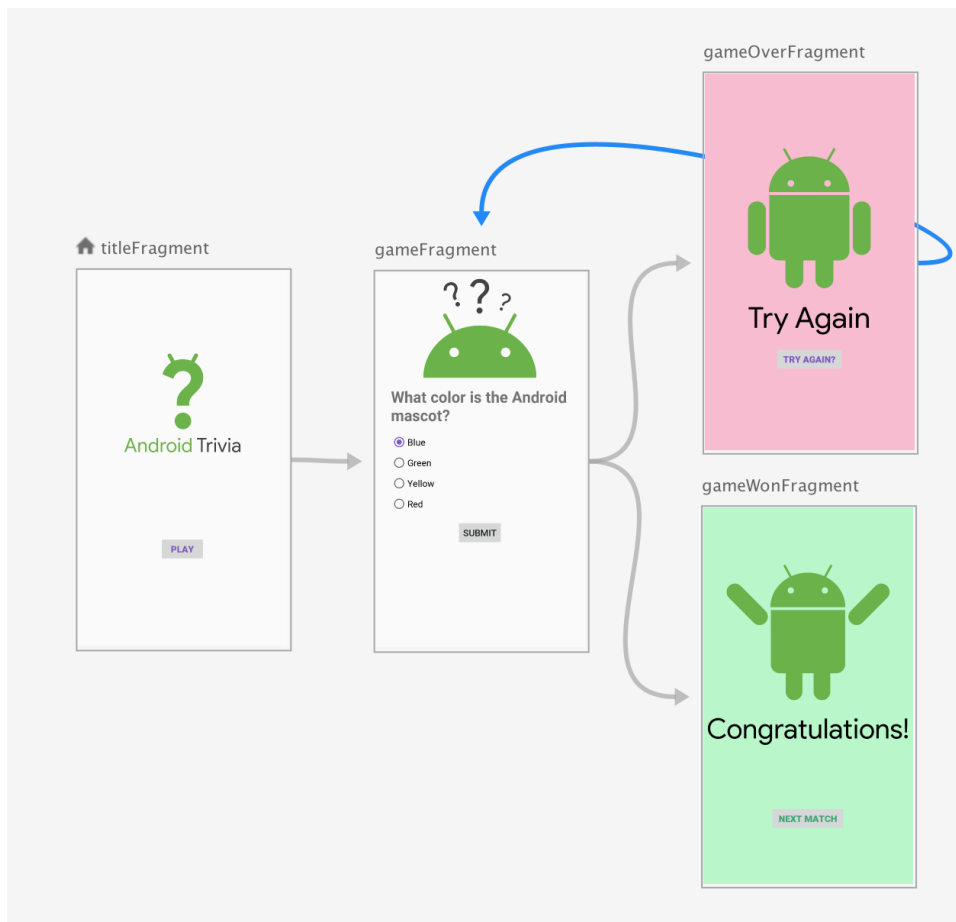
- Użytkownik gra i wygrywa lub przegrywa, a aplikacja przechodzi do ekranu `GameWon` lub `GameOver`.
- Jeśli w tym momencie użytkownik naciśnie przycisk Wstecz systemu, aplikacja przejdzie do `TitleFragment`.

W tym kroku wdrażasz dwa kolejne kroki przepływu:

- Jeśli użytkownik stuknie przycisk **Next Match** lub **Try Again** aplikacja przejdzie do ekranu `GameFragment`.
- Jeśli w tym momencie użytkownik naciśnie przycisk Wstecz systemu, aplikacja przejdzie do ekranu `TitleFragment` screen (zamiast z powrotem do ekranu `GameWon` lub `GameOver`).

Aby utworzyć ten przepływ *user flow*, użyj atrybutu `PopUpTo` do zarządzania stosem:

1. W pliku `navigation.xml` dodaj akcję nawigacyjną łączącą `gameOverFragment` z `gameFragment`. Upewnij się, że nazwy fragmentów w identyfikatorze akcji są zgodne z nazwami fragmentów zawartymi w pliku XML. Na przykład identyfikatorem akcji może być `action_gameOverFragment_to_gameFragment`.



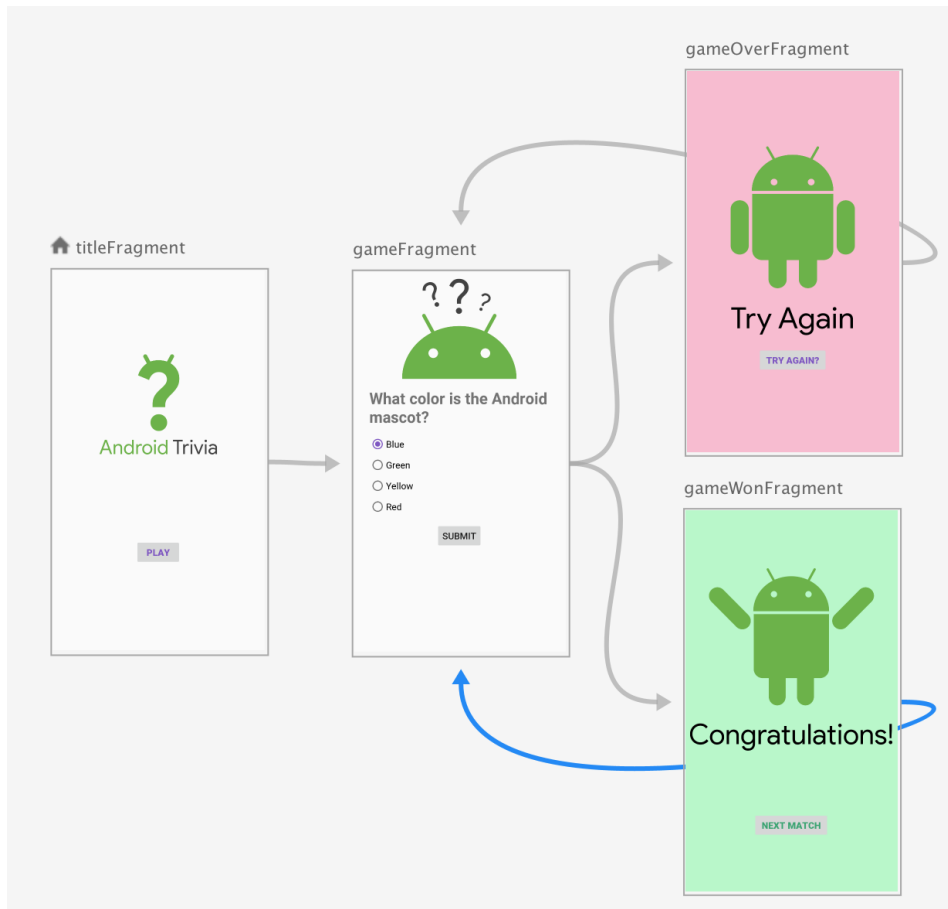
2. W panelu **Attributes** ustaw atrybut **Pop To** akcji na titleFragment.
3. Wyczyść checkbox **Inclusive**, ponieważ nie chcesz, aby titleFragment był uwzględniany w miejscach docelowych, które zostały usunięte z tylnego stosu. Zamiast tego chcesz usunąć wszystko aż do TitleFragment (ale nie włączając go) ze stosu.

▼ **Pop Behavior**

Pop To titleFragment ▼

Inclusive ☐

4. W pliku navigation.xml dodaj akcję nawigacyjną łączącą gameWonFragment z gameFragment.



5. Dla akcji, którą właśnie utworzyłeś, ustaw odpowiedni atrybut **Pop To** do titleFragment.



Teraz dodaj funkcjonalność do przycisków **Try Again** i **Next Match** Gdy użytkownik stuknie którykolwiek z przycisków, aplikacja ma przejść do ekranu GameFragment aby użytkownik mógł ponownie spróbować uruchomić grę.

1. Otwórz plik GameOverFragment.kt Na końcu metody onCreateView() przed instrukcją return dodaj następujący kod. Kod dodaje click listener do przycisku **Try Again** Gdy użytkownik naciśnie przycisk, aplikacja przechodzi do fragmentu gry.

```
// Add OnClick Handler for Try Again button
binding.tryAgainButton.setOnClickListener{view: View->
    view.findNavController()
        .navigate(R.id.action_gameOverFragment_to_gameFragment) }
```

2. Otwórz plik GameWonFragment.kt Na końcu metody onCreateView() dodaj następujący kod:

```
// Add OnClick Handler for Next Match button
binding.nextMatchButton.setOnClickListener{view: View->
    view.findNavController().navigate(R.id.action_gameWonFragment_to_gameFragment)}
```

3. Uruchom aplikację, zagraj w grę i przetestuj przyciski Następny mecz i Spróbuj ponownie. Oba przyciski powinny zabrać Cię z powrotem do ekranu gry, abyś mógł ponownie wypróbować grę.
4. Po wygraniu lub przegranej, stuknij w Następny mecz lub Spróbuj ponownie. Teraz naciśnij przycisk Wstecz systemu. Aplikacja powinna przejść do ekranu tytułowego, zamiast wracać do ekranu, z którego właśnie przyszedłeś.

## 8. Zadanie: Dodaj przycisk Up na pasku aplikacji

### Pasek aplikacji

*AppBar*, Pasek aplikacji zapewnia użytkownikowi dostęp do znanych funkcji nawigacji, takich jak menu opcji. Aby uzyskać dostęp do menu opcji z paska aplikacji, użytkownik naciska



ikonę trzema pionowymi kropkami.

Pasek aplikacji wyświetla title string, który można zmieniać na każdym ekranie. Na ekranie tytułowym aplikacji AndroidTrivia pasek aplikacji wyświetla „Android Trivia”. Na ekranie pytania ciąg tytułowy pokazuje również, na które pytanie użytkownik odpowiada („1/3”, „2/3” lub „3/3”).

### Up button

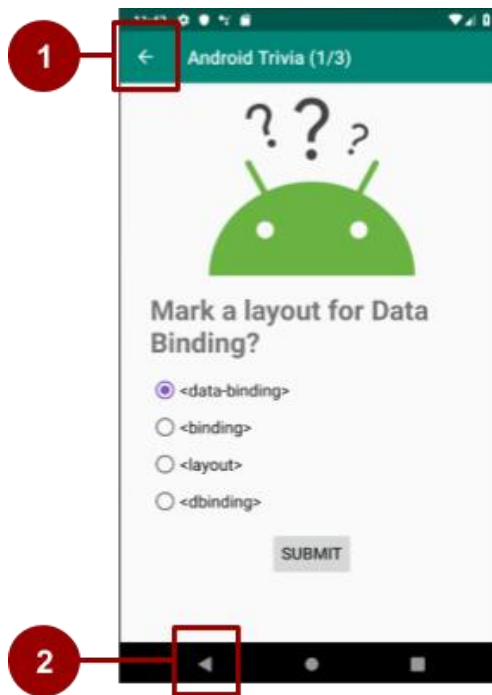
Obecnie w aplikacji użytkownik używa systemowego przycisku Wstecz, aby przejść do poprzednich ekranów. Jednak aplikacje na Androida mogą mieć także przycisk ekranowy w górę, który pojawia się w lewym górnym rogu paska aplikacji.

W aplikacji AndroidTrivia chcesz, aby przycisk W górę pojawiał się na każdym ekranie oprócz ekranu tytułowego. Przycisk W górę powinien zniknąć, gdy użytkownik dotrze do ekranu tytułowego, ponieważ ekran tytułowy znajduje się na szczycie hierarchii ekranów aplikacji.

#### Przycisk W górę (Up) kontra przycisk Wstecz:

- Przycisk W górę, pokazany jako 1 na zrzucie ekranu poniżej, pojawia się na pasku aplikacji.
- Przycisk W górę nawiguje po aplikacji w oparciu o hierarchiczne relacje między ekranami. Przycisk W górę nigdy nawiguje użytkownika z aplikacji.
- Przycisk Wstecz porusza się wstecz po ekranach, z którymi użytkownik ostatnio pracował (back stack).





## Dodaj obsługę przycisku W górę (Up button)

Komponenty nawigacyjne obejmują bibliotekę interfejsu użytkownika o nazwie [NavigationUI](#). Kontroler nawigacyjny integruje się z paskiem aplikacji, aby zaimplementować zachowanie przycisku W górę, dzięki czemu nie musisz tego robić sam.

W poniższych krokach za pomocą kontrolera nawigacyjnego dodaj przycisk W górę do swojej aplikacji:

1. Otwórz plik `MainActivity.kt` Wewnątrz metody `onCreate()` dodaj kod, aby znaleźć obiekt kontrolera nawigacji:

```
val navController = this.findNavController(R.id.myNavHostFragment)
```

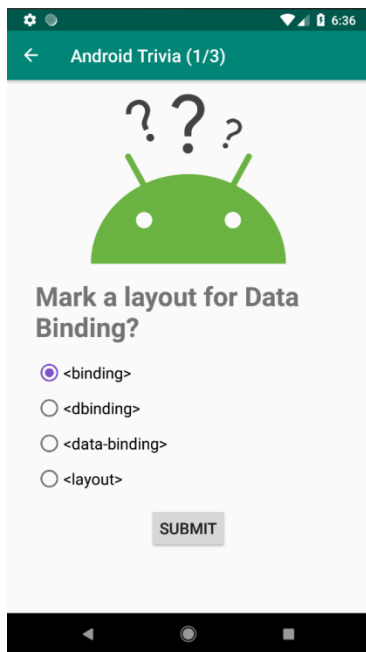
2. Również wewnątrz metody `onCreate()` dodaj kod, aby połączyć kontroler nawigacji z paskiem aplikacji:

```
NavigationUI.setupActionBarWithNavController(this, navController)
```

3. Po metodzie `onCreate()` zastąp metodę (override) `onSupportNavigateUp()` aby wywołać `navigateUp()` w kontrolerze nawigacyjnym:


```
override fun onSupportNavigateUp(): Boolean {
    val navController = this.findNavController(R.id.myNavHostFragment)
    return navController.navigateUp()
}
```

4. Uruchom aplikację. Przycisk W górę pojawia się na pasku aplikacji na każdym ekranie oprócz ekranu tytułowego. Bez względu na to, gdzie jesteś w aplikacji, naciśnięcie przycisku W górę spowoduje przejście do ekranu tytułowego.



## . Zadanie: Dodaj menu opcji (options menu)

Android ma różne rodzaje menu, w tym menu opcji. Na nowoczesnych urządzeniach z systemem Android użytkownik uzyskuje dostęp do menu opcji, dotykając trzech pionowych

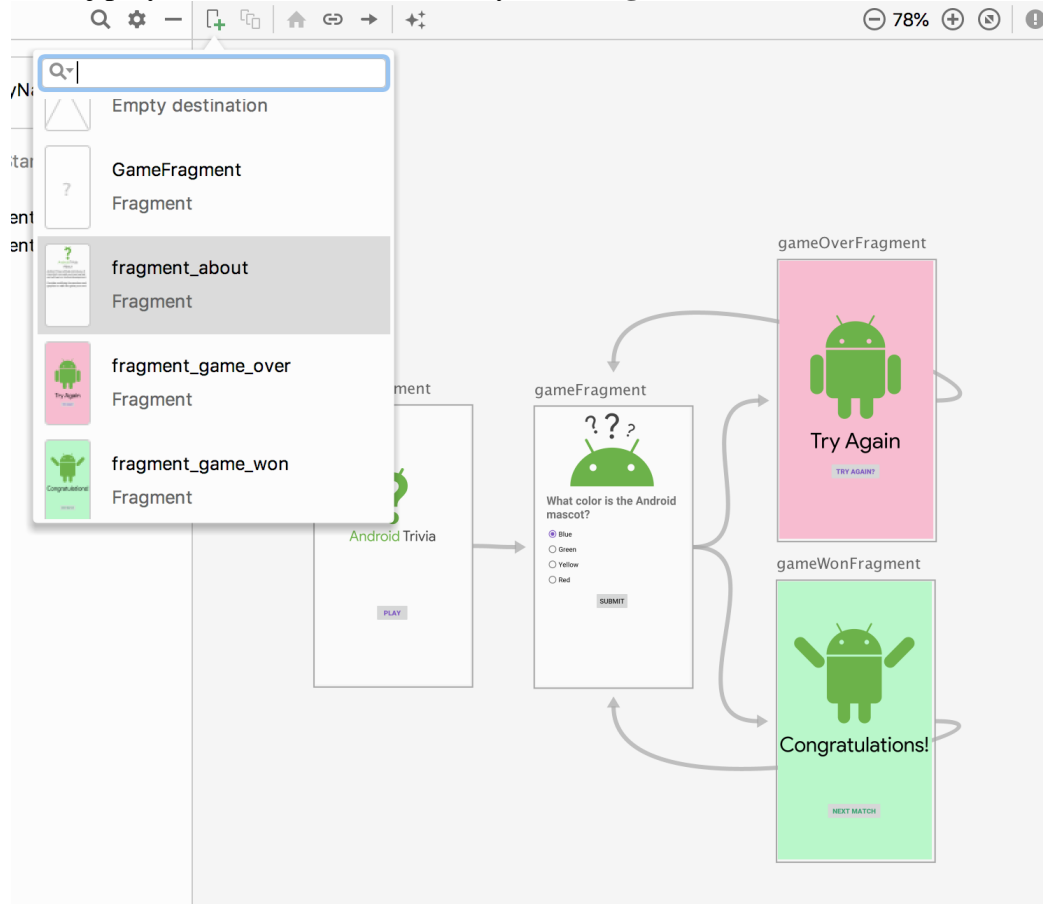
kropek  wyświetlanych na pasku aplikacji.

W tym zadaniu dodajesz pozycję menu Informacje do menu opcji. Gdy użytkownik stuknie element menu Informacje, aplikacja przejdzie do części AboutFragment, a użytkownik zobaczy informacje o tym, jak korzystać z aplikacji.

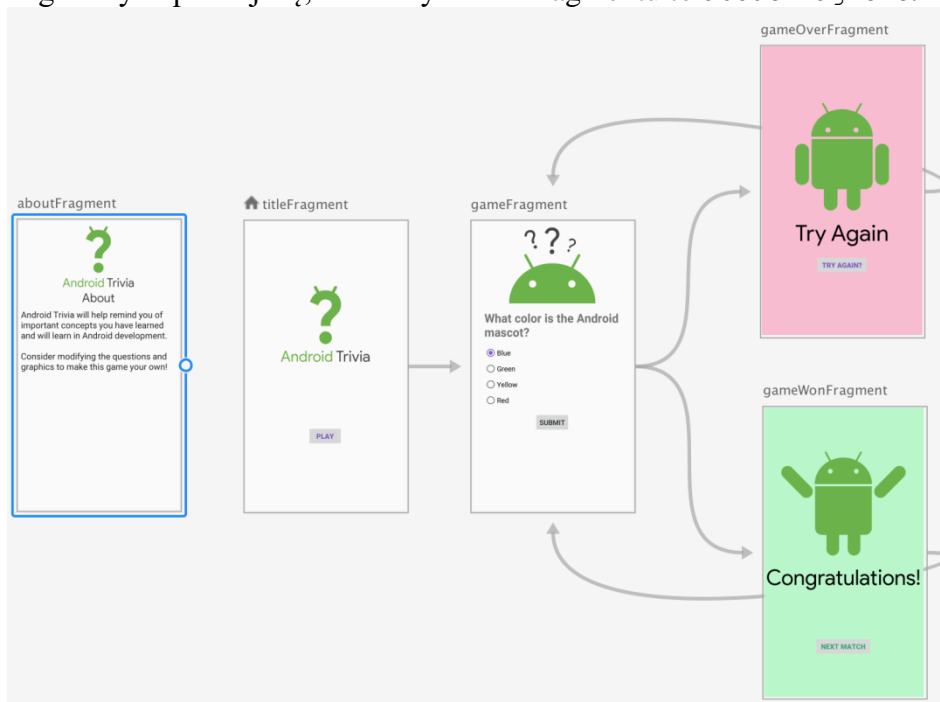
## Krok 1: Dodaj AboutFragment do navigation graph

1. Otwórz plik `navigation.xml` i kliknij kartę **Design** aby wyświetlić navigation graph.

2. Kliknij przycisk **New Destination** i wybierz **fragment\_about**.

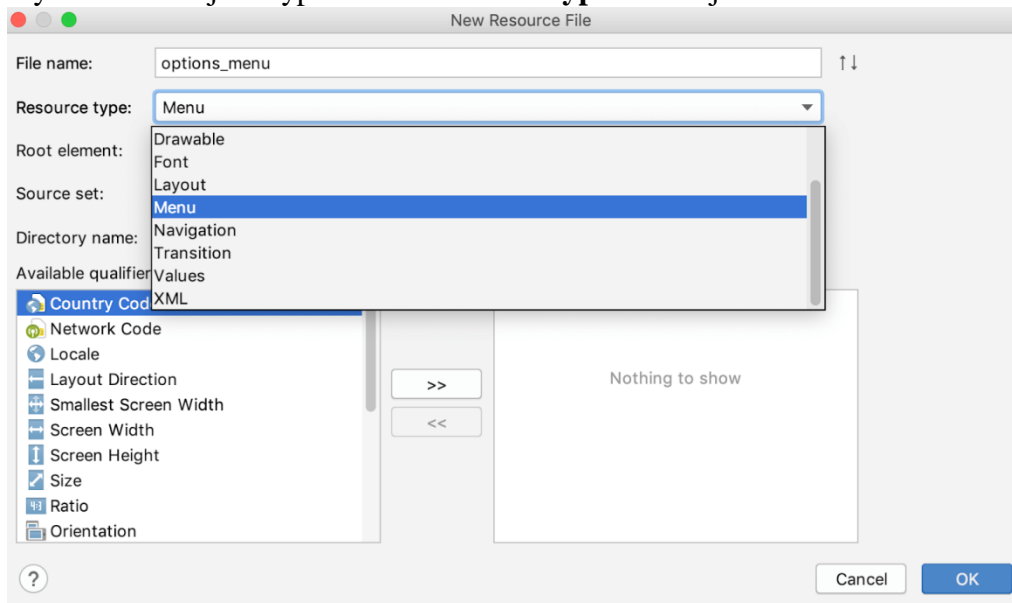


3. W edytorze układu przeciągnij fragment „około” w lewo, aby nie zachodził na inne fragmenty. Upewnij się, że identyfikator fragmentu to `aboutFragment`.

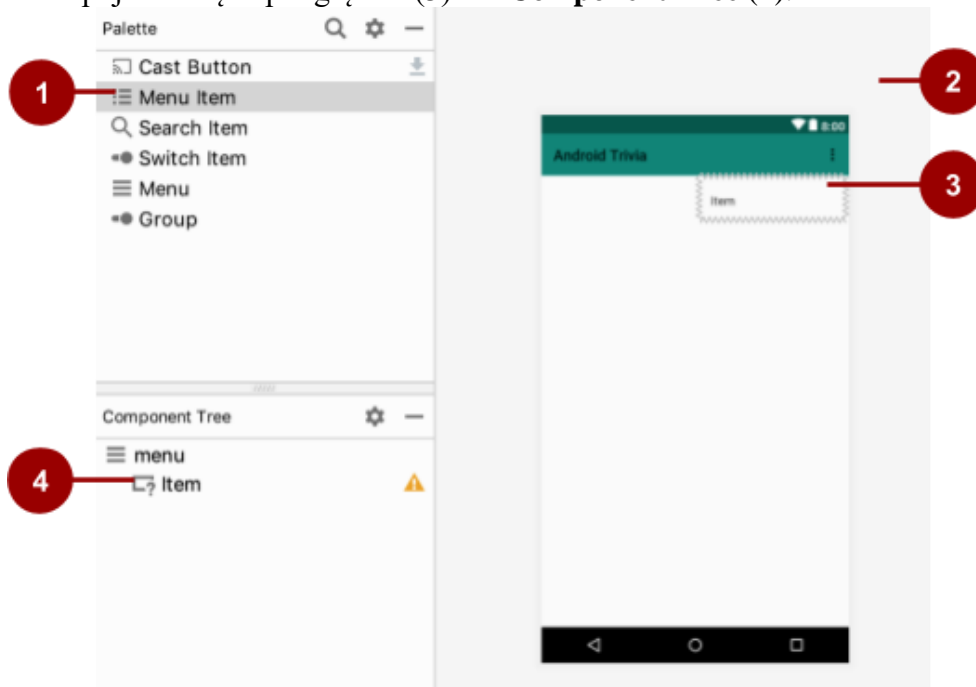


## Krok 2: Dodaj zasób menu opcji (options-menu resource)

1. W okienku Projekt Android Studio kliknij prawym przyciskiem myszy folder **res** i wybierz **New > Android Resource File**.
2. W oknie dialogowym **New Resource File** nazwij plik **options\_menu**.
3. Wybierz **Menu** jako typ zasobu **Resource type** i kliknij **OK**.

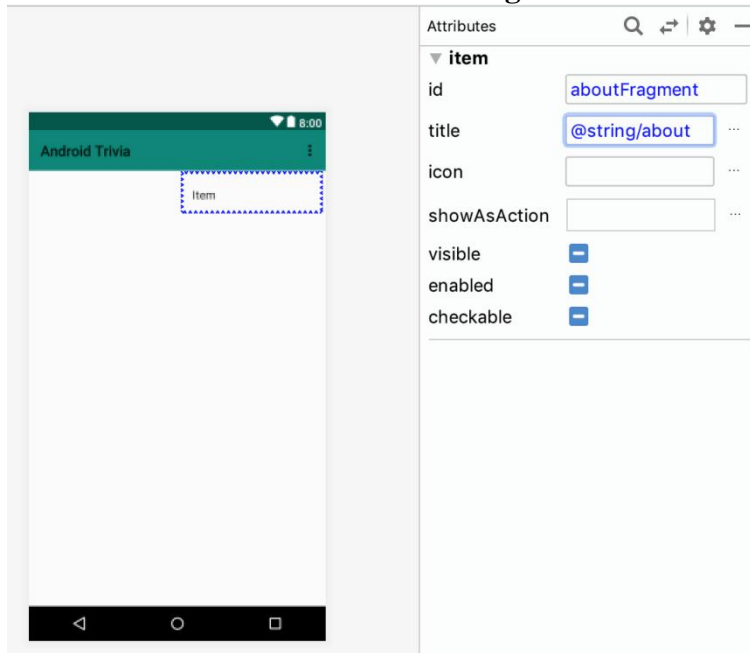


3. Otwórz plik `options_menu.xml` z folderu **res > menu** i kliknij kartę **Design** aby wyświetlić Layout Editor.
4. Z panelu **Palette** przeciągnij element **Menu Item** (pokazany jako 1 na zrzucie ekranu poniżej) i upuść go w dowolnym miejscu w panelu edytora projektu (2). Element menu pojawia się w podglądzie (3) i w **Component Tree** (4).



5. W podglądzie lub w **Component Tree**, kliknij element menu, aby wyświetlić jego atrybuty w panelu **Attributes**.

6. Ustaw ID elementu menu na **aboutFragment**. Ustaw title na **@string/about**.



**Tip:** upewnij się, że identyfikator dodanego elementu menu jest dokładnie taki sam, jak identyfikator AboutFragment dodanego na wykresie nawigacyjnym. To znacznie uprości kod dla procedury obsługi `onClick`.

### Krok 3: Dodaj moduł obsługi `onClick` (`onClick handler`)

W tym kroku dodajesz kod, aby zaimplementować zachowanie, gdy użytkownik stuknie element menu Informacje.

1. Otwórz plik `TitleFragment.kt` Wewnątrz metody `onCreateView()` przed `return`, wywołaj metodę `setHasOptionsMenu()` i przekaż `true`.

```
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?): View? {
    ...
    setHasOptionsMenu(true)
    return binding.root
}
```

2. Po metodzie `onCreateView()` zastąp metodę (override) `onCreateOptionsMenu()` W metodzie dodaj menu opcji i napełnij plik zasobów menu.

```
override fun onCreateOptionsMenu(menu: Menu?, inflater: MenuInflater?) {
    super.onCreateOptionsMenu(menu, inflater)
    inflater?.inflate(R.menu.options_menu, menu)
}
```

3. Zastąp metodę `onOptionsItemSelected()` aby podjąć odpowiednie działanie po dotknięciu elementu menu. W takim przypadku czynnością jest przejście do fragmentu, który ma taki sam identyfikator, jak wybrany element menu.

```
override fun onOptionsItemSelected(item: MenuItem?): Boolean {
```

```

return NavigationUI.onNavDestinationSelected(item!!,
    view!!.findNavController())
    || super.onOptionsItemSelected(item)
}

```

4. Jeśli aplikacja nie zostanie zbudowana, sprawdź, czy musisz zaimportować pakiety, aby naprawić nierozwiązane odwołania w kodzie. Na przykład możesz dodać `import android.view.*`. Aby rozwiązać kilka odniesień (i zastąpić bardziej szczegółowe importy, takie jak `import import android.view.ViewGroup`).
5. Uruchom aplikację i przetestuj element menu About znajdujący się w menu opcji. Po dotknięciu elementu menu aplikacja powinna przejść do ekranu "about".

