Workflow:

```
npx create-expo-app@latest . --template # blank (typescript)
npm install @azesmway/react-native-unity
npm install --save-dev @react-native-community/cli
npx expo prebuild # Get android and ios build
```

Create directories in the root of the project:

- unity/builds/
    - android
    - ios

Inside the android directory of the project, place the unity build for android.

Edit `android/settings.gradle` inside the React Native Project with:

```
include ':unityLibrary'
project(':unityLibrary').projectDir=new File('../unity/builds/android/unityLibrary')
```

Inside `android/build.gradle`, add:

```
allprojects {
    repositories {
        flatDir {
            dirs "${project(':unityLibrary').projectDir}/libs"
        }
    }
}
```

Modify `android/gradle.properties`

```
unityStreamingAssets=.unity3d
```

Modify `android/app/src/main/res/values/strings.xml`

```
<string name="game_view_content_description">Game view</string>
```

Update Unity's Android Manifest

Go to `unity/builds/android/unityLibrary/src/main/AndroidManifest.xml` and **remove** the `<intent-filter>` tags to prevent Unity from launching as a separate app.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools">
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-feature android:glEsVersion="0x00030000" />
  <uses-feature android:name="android.hardware.vulkan.version" android:required="false" />
  <uses-feature android:name="android.hardware.touchscreen" android:required="false" />
  <uses-feature android:name="android.hardware.touchscreen.multitouch" android:required="false" />
  <uses-feature android:name="android.hardware.touchscreen.multitouch.distinct" android:required="false" />
  <application android:enableOnBackInvokedCallback="false" android:extractNativeLibs="true" android:appCategory="game">
    <meta-data android:name="unity.splash-mode" android:value="0" />
    <meta-data android:name="unity.splash-enable" android:value="True" />
    <meta-data android:name="unity.launch-fullscreen" android:value="True" />
    <meta-data android:name="unity.render-outside-safearea" android:value="True" />
    <meta-data android:name="notch.config" android:value="portrait|landscape" />
    <meta-data android:name="unity.auto-report-fully-drawn" android:value="true" />
    <meta-data android:name="unity.auto-set-game-state" android:value="true" />
    <meta-data android:name="unity.strip-engine-code" android:value="true" />
    <meta-data android:name="android.game_mode_config" android:resource="@xml/game_mode_config" />
    <activity android:configChanges="mcc|mnc|locale|touchscreen|keyboard|keyboardHidden|navigation|orientation|screenLayout|uiMode|s
      <meta-data android:name="unityplayer.UnityActivity" android:value="true" />
      <meta-data android:name="android.app.lib_name" android:value="game" />
      <meta-data android:name="WindowManagerPreference:FreeformWindowSize" android:value="@string/FreeformWindowSize_maximize" />
      <meta-data android:name="WindowManagerPreference:FreeformWindowOrientation" android:value="@string/FreeformWindowOrientation_l
      <meta-data android:name="notch_support" android:value="true" />
    </activity>
  </application>
</manifest>
```

Create `App.tsx`

```
import React, { useRef } from 'react';
import { View } from 'react-native';
import UnityView from '@azesmway/react-native-unity';

const App = () => {
  const unityRef = useRef<typeof UnityView|null>(null);

  return (
    <View style={{ flex: 1 }}>
      <UnityView
        ref={unityRef}
        style={{ width: '100%', height: '100%' }}
      />
    </View>
  );
};

export default App;
```

Create `react-native-unity.d.ts`

```
declare module '@azesmway/react-native-unity'
```

Error:

```
> Project with path ':unityLibrary:mobilenotifications.androidlib' could not be found in project ':unityLibrary'. * Try:
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
> Run with --scan to get full insights.
> Get more help at https://help.gradle.org.
============================================================================== 2: Task failed with an exception.
```

Remove below from `/unity/builds/android/settings.gradle` and `/unity/builds/android/unityLibrary/build.gradle`:

```
include 'unityLibrary:mobilenotifications.androidlib'
```

Issues with NDK version

```
 error Failed to install the app. Command failed with exit code 1: ./gradlew app:installDebug -PreactNativeDevServerPort=8081 FAILURE
A problem occurred configuring project ':unityLibrary'.
> com.android.builder.errors.EvalIssueException: [CXX1101] Location specified by android.ndkPath (/Applications/Unity/Hub/Editor/600
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
> Run with --scan to get full insights.
> Get more help at https://help.gradle.org. BUILD FAILED in 5s.
info Run CLI with --verbose flag for more details.
```

Changing NDK version can be done below but there is an issue with the UPlayer.java file in the package itself (`@azesmway/react-native-unity`). Replace it with below code having type fixes and removed unnecessary types:

`./node_modules/@azesmway/react-native-unity/android/src/main/java/com/azesmwayreactnativeunity/UPlayer.java`:

```java
package com.azesmwayreactnativeunity;

import android.app.Activity;
import android.content.res.Configuration;
import android.view.View;
import android.widget.FrameLayout;

import com.unity3d.player.IUnityPlayerLifecycleEvents;
import com.unity3d.player.UnityPlayer;

import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

public class UPlayer {
    private static UnityPlayer unityPlayer;

    public UPlayer(final Activity activity, final ReactNativeUnity.UnityPlayerCallback callback)
            throws ClassNotFoundException, InvocationTargetException, IllegalAccessException, InstantiationException {
        Class<?> _player = null;
        try {
            _player = Class.forName("com.unity3d.player.UnityPlayerForActivityOrService");
        } catch (ClassNotFoundException e) {
            _player = Class.forName("com.unity3d.player.UnityPlayer");
        }

        // Note: using the second constructor based on the original code.
        Constructor<?> constructor = _player.getConstructors()[1];
        unityPlayer = (UnityPlayer) constructor.newInstance(activity, new IUnityPlayerLifecycleEvents() {
            @Override
            public void onUnityPlayerUnloaded() {
                callback.onUnload();
            }

            @Override
            public void onUnityPlayerQuitted() {
                callback.onQuit();
            }
        });
    }

    public static void UnitySendMessage(String gameObject, String methodName, String message) {
        UnityPlayer.UnitySendMessage(gameObject, methodName, message);
    }

    public void pause() {
        unityPlayer.pause();
    }

    public void windowFocusChanged(boolean hasFocus) {
        unityPlayer.windowFocusChanged(hasFocus);
    }

    public void resume() {
        unityPlayer.resume();
    }

    public void unload() {
        unityPlayer.unload();
    }

    public Object getParentPlayer() throws NoSuchMethodException, InvocationTargetException, IllegalAccessException {
        try {
            Method getFrameLayout = unityPlayer.getClass().getMethod("getFrameLayout");
            FrameLayout frame = (FrameLayout) this.requestFrame();
            return frame.getParent();
        } catch (NoSuchMethodException e) {
            Method getParent = unityPlayer.getClass().getMethod("getParent");
            return getParent.invoke(unityPlayer);
        }
    }

    public void configurationChanged(Configuration newConfig) {
        unityPlayer.configurationChanged(newConfig);
    }

    public void destroy() {
        unityPlayer.destroy();
    }

    public void requestFocusPlayer() throws NoSuchMethodException, InvocationTargetException, IllegalAccessException {
        try {
            // Try to get the frame and request focus
            FrameLayout frame = (FrameLayout) this.requestFrame();
            frame.requestFocus();
        } catch (NoSuchMethodException e) {
            // Fallback: call requestFocus directly on unityPlayer
            Method requestFocus = unityPlayer.getClass().getMethod("requestFocus");
            requestFocus.invoke(unityPlayer);
        }
    }

    /**
     * Attempts to retrieve a FrameLayout from UnityPlayer by calling getFrameLayout.
     * If not available, falls back to calling a hypothetical getView() method and wraps the result in a FrameLayout.
     */
    public FrameLayout requestFrame() throws NoSuchMethodException {
        try {
            Method getFrameLayout = unityPlayer.getClass().getMethod("getFrameLayout");
            return (FrameLayout) getFrameLayout.invoke(unityPlayer);
        } catch (NoSuchMethodException | IllegalAccessException | InvocationTargetException e) {
            // Fallback: try to use getView() if available
            FrameLayout frameLayout = new FrameLayout(unityPlayer.getContext());
            try {
                Method getViewMethod = unityPlayer.getClass().getMethod("getView");
                Object viewObj = getViewMethod.invoke(unityPlayer);
                if (viewObj instanceof View) {
                    frameLayout.addView((View) viewObj);
                    return frameLayout;
                } else {
                    throw new RuntimeException("UnityPlayer.getView() did not return a View instance.");
                }
            } catch (NoSuchMethodException | IllegalAccessException | InvocationTargetException ex) {
                throw new RuntimeException("Neither getFrameLayout nor getView are available in UnityPlayer.", ex);
            }
        }
    }

    public void setZ(float z) throws NoSuchMethodException, InvocationTargetException, IllegalAccessException {
        try {
            Method setZ = unityPlayer.getClass().getMethod("setZ");
            setZ.invoke(unityPlayer, z);
        } catch (NoSuchMethodException e) {
            // Do nothing if setZ is not available
        }
    }

    public Object getContextPlayer() {
        return unityPlayer.getContext();
    }
}
```

Delete other NDK versions from Android studio in SDK manager and install NDK 21.3.6528147

> ndkPath: `/Users/aravinthan/Library/Android/sdk/ndk/<ndkversion>`

Change it across the codebase along with ndkVersion: `<ndkversion>`