AndroidStudio

Nachfolgend soll die Installation und Konfiguration von Android Studio unter DEBIAN Linux dokumentiert werden. Es wird kein Anspruch auf Vollständigkeit erhoben.

Voraussetzungen

Es wird ein DEBIAN-Linux-System vorausgesetzt, sowie eine Verbindung zum Internet, um Pakete und das Android Studio selbst herunterzuladen. Sollte ein Minimalinstallation als Voraussetzung genutzt worden sein, ist die Installation einer grafischen Benutzeroberfläche zwingend notwendig, da Android Studio ein grafisches Programm ist.

JAVA

Sollte das distributionseigene JDK genutzt werden, muss folgendes Paket installiert werden: „`openjdk-8-jdk`“.

Es gibt aber auch die Möglichkeit Oracle JAVA zu installieren. Hierfür muss das JDK-Archiv (nachfolgend das Archiv: „`jdk-8u162-linux-x64.tar.gz`“) heruntergeladen und mit Hilfe von „`make-jpkg`“ in ein DEBIAN-Paket umgewandelt werden.

Installation des DEBIAN-Paketes, welches „`make-jpkg`“ enthält:

```
~# apt install java-package
```

Umwandeln des heruntergeladenen JAVA-Archivs (als Benutzer):

```
~$ make-jpkg jdk-8u162-linux-x64.tar.gz
Creating temporary directory: /tmp/make-jpkg.dhA9AOK2g4
Loading plugins: /usr/share/java-package/common.sh /usr/share/java-
package/javase.sh /usr/share/java-package/jdk-doc.sh \
/usr/share/java-package/oracle-jdk-doc.sh /usr/share/java-package/oracle-
jdk.sh /usr/share/java-package/oracle-jre.sh \
/usr/share/java-package/oracle-server-jre.sh

Detected Debian build architecture: amd64
Detected Debian GNU type: x86_64-linux-gnu
```

```
Detected product:
    Java(TM) Development Kit (JDK)
    Standard Edition, Version 8 Update 162
    Oracle(TM)
Is this correct [Y/n]: Y

Checking free diskspace: done.

In the next step, the binary file will be extracted. Probably a
license agreement will be displayed. Please read this agreement
carefully. If you do not agree to the displayed license terms, the
package will not be built.

Press [Return] to continue:



Testing extracted archive... okay.

Create debian package:
dpkg-buildpackage: Information: Quellpaket oracle-java8-jdk
dpkg-buildpackage: Information: Quellversion 8u162

...

dpkg-deb: Paket »oracle-java8-jdk-dbgsym« wird in »../oracle-java8-jdk-
dbgsym_8u162_amd64.deb« gebaut.
dpkg-deb: Paket »oracle-java8-jdk« wird in »../oracle-java8-
jdk_8u162_amd64.deb« gebaut.
 dpkg-genbuildinfo --build=binary
 dpkg-genchanges --build=binary >../oracle-java8-jdk_8u162_amd64.changes
dpkg-genchanges: Information: Binärpaket(e) hochzuladen (kein Quellcode
enthalten)
 dpkg-source --after-build package
dpkg-buildpackage: Information: Binärpaket(e) hochzuladen (keine Quelle
enthalten)
    copy oracle-java8-jdk_8u162_amd64.deb into directory
/home/service/Downloads/

The Debian package has been created in the current directory.
You can install the package as root with:

    dpkg -i oracle-java8-jdk_8u162_amd64.deb


Removing temporary directory: done
```

Bevor das Paket installiert werden kann, muss die Abhängigkeit „`java-common`“ noch installiert werden:

```
~# apt install java-common
```

Das erstellte Paket („`oracle-java8-jdk_8u162_amd64.deb`“) kann, wie bereits oben vorgeschlagen, dann manuell installiert werden:

```
~# dpkg -i oracle-java8-jdk_8u162_amd64.deb
```

Download

Das Programm „`Android Studio`“ kann in einer aktuellen Version von der Seite android.com heruntergeladen. Das ZIP-Archiv ist über 700 MB groß, der Download dauert also je nach Anbindung eine Weile. Das Archiv kann an beliebiger Stelle im Dateisystem entpackt werden, nachfolgend ist es das Verzeichnis „`/data/AndroidStudio/AndroidStudio3.0`“ (Das Verzeichnis „`/data/`“ ist eine eigene Partition).

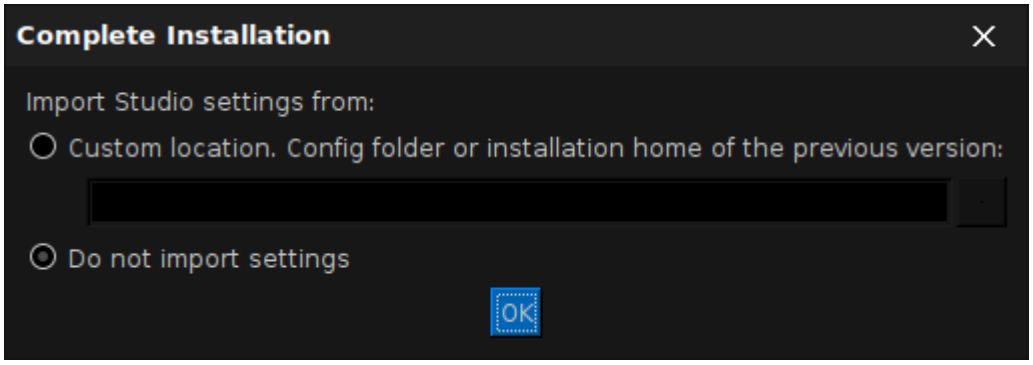
Konfiguration

Leider bringt das Programm keinen eigenen „Launcher“ mit (eine Datei mit der Endung „`*.desktop`“). Dieser kann aber relativ einfach erstellt werden. Dazu einfach eine Datei (zum Beispiel: „`AndroidStudio3.0.desktop`“) mit folgendem Inhalt erstellen:

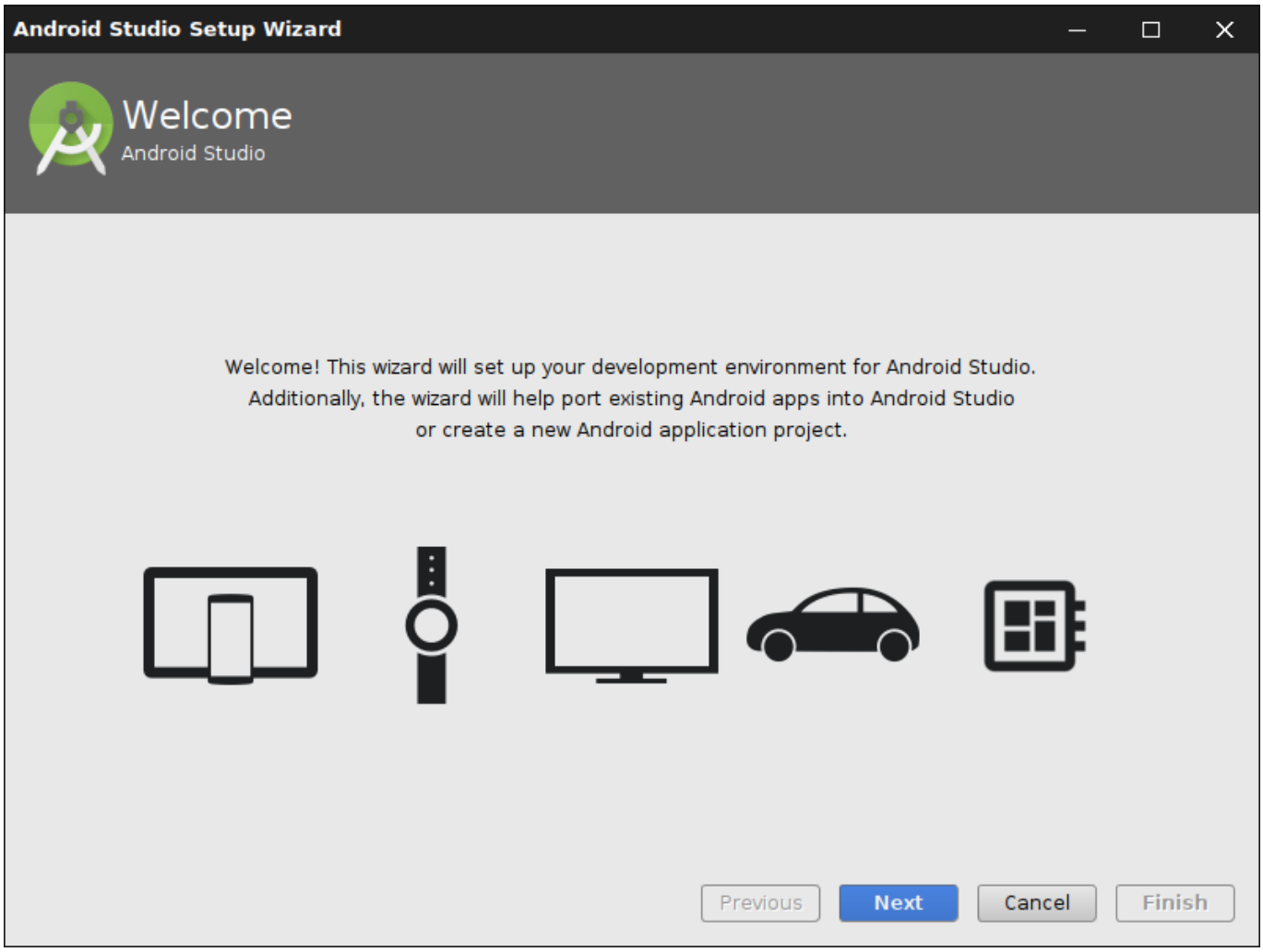
```
[Desktop Entry]
Version=1.0
Type=Application
Name=Android Studio 3.0
Name[de_DE]=Android Studio 3.0
Exec="/data/AndroidStudio/AndroidStudio3.0/bin/studio.sh" %f
Icon=/data/AndroidStudio/AndroidStudio3.0/bin/studio.png
Categories=Development;IDE;
Terminal=false
StartupNotify=true
```

Der Pfad muss hier natürlich entsprechend den eigenen Gegebenheiten angepasst werden.

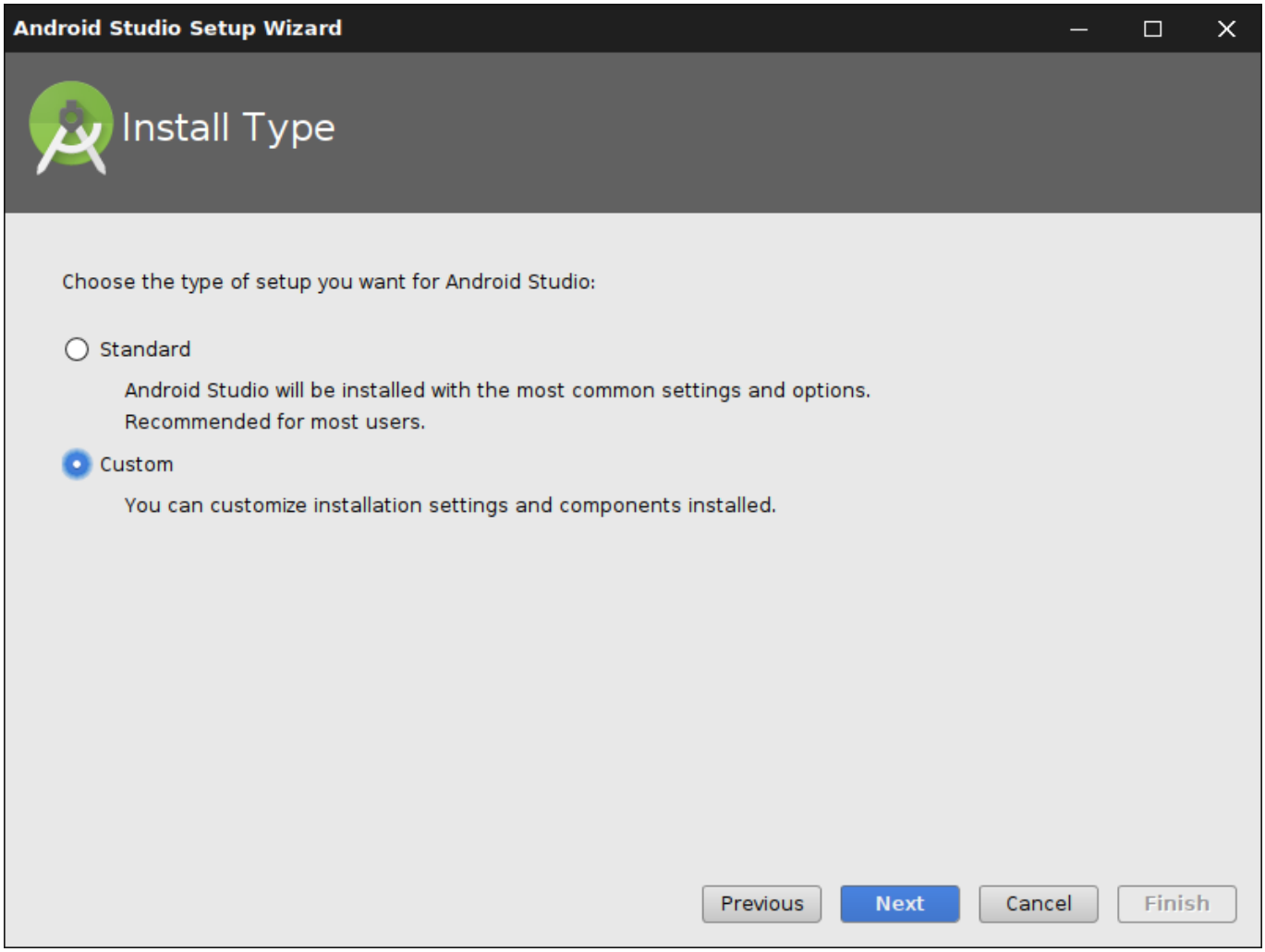
Wird das Programm das erste Mal gestartet, können Einstellungen aus alten Versionen bei Bedarf importiert werden:



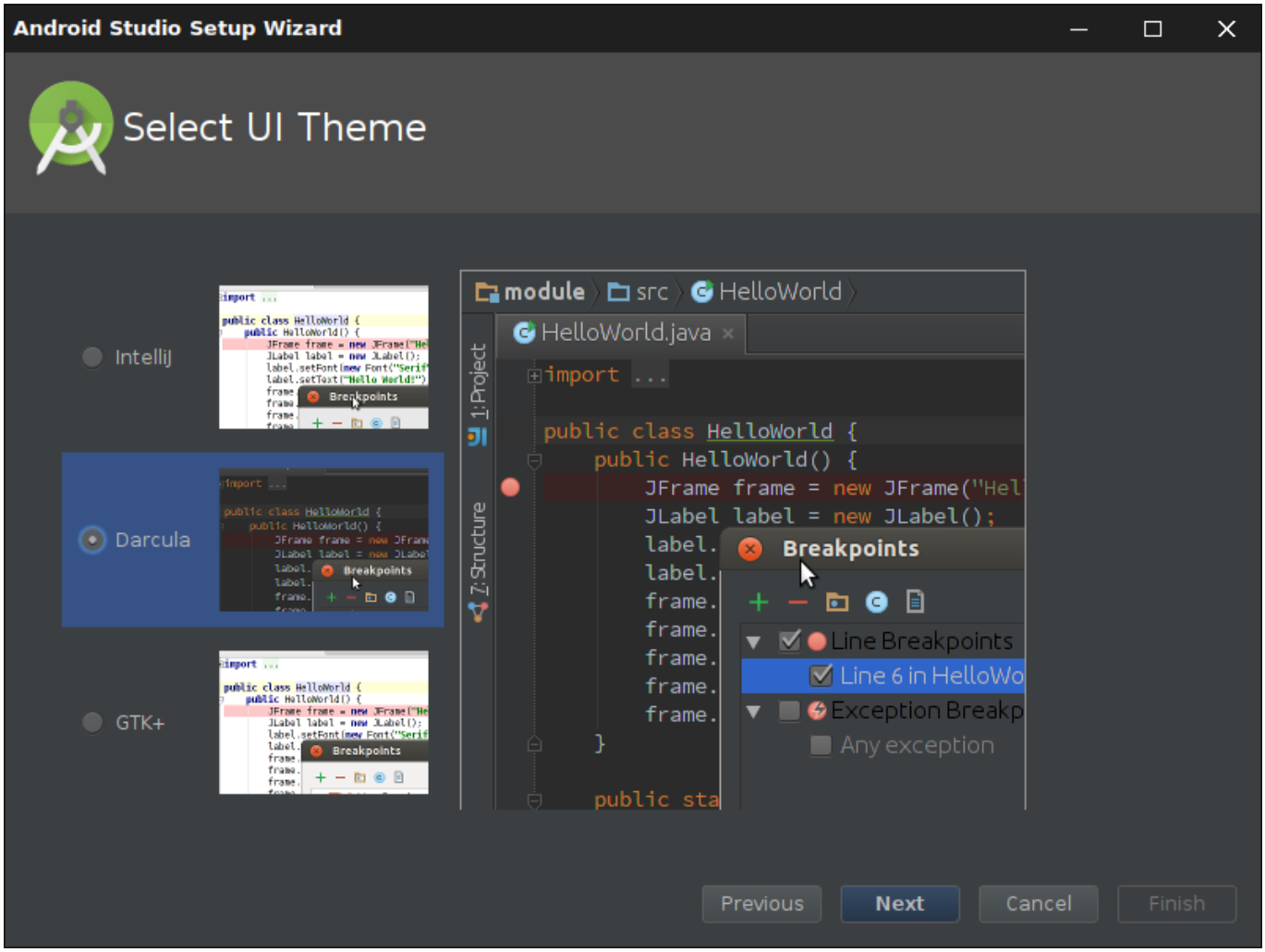
Der Willkommensbildschirm leitet durch die erste Konfiguration:



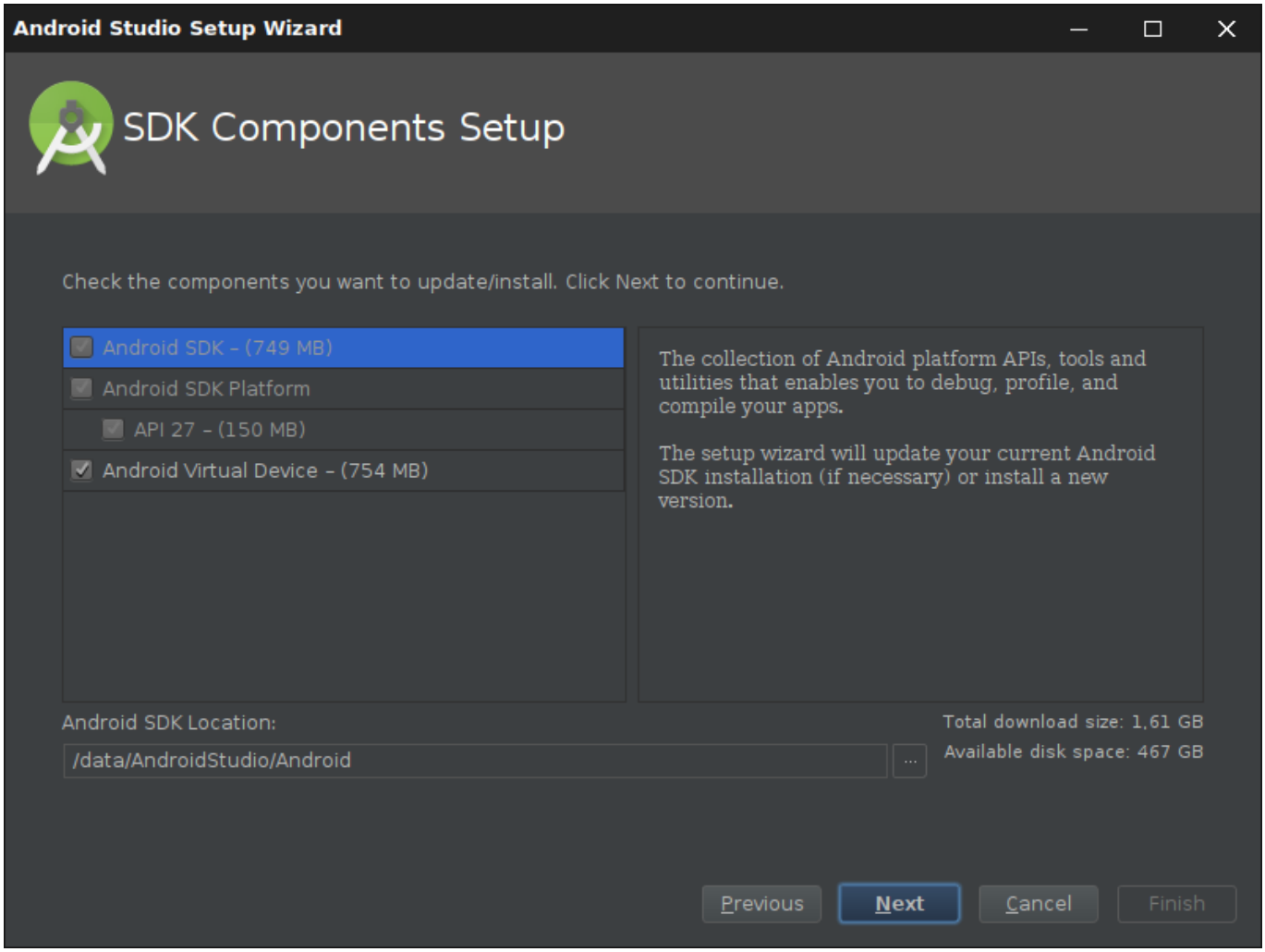
Der Benutzer kann entscheiden, ob Android Studio alle Einstellungen selbst vornehmen soll oder man die einzelnen Komponenten selbst auswählt:



Soll der Benutzer selbst entscheiden, kann als erstes ein gewünschtes Thema ausgewählt werden:

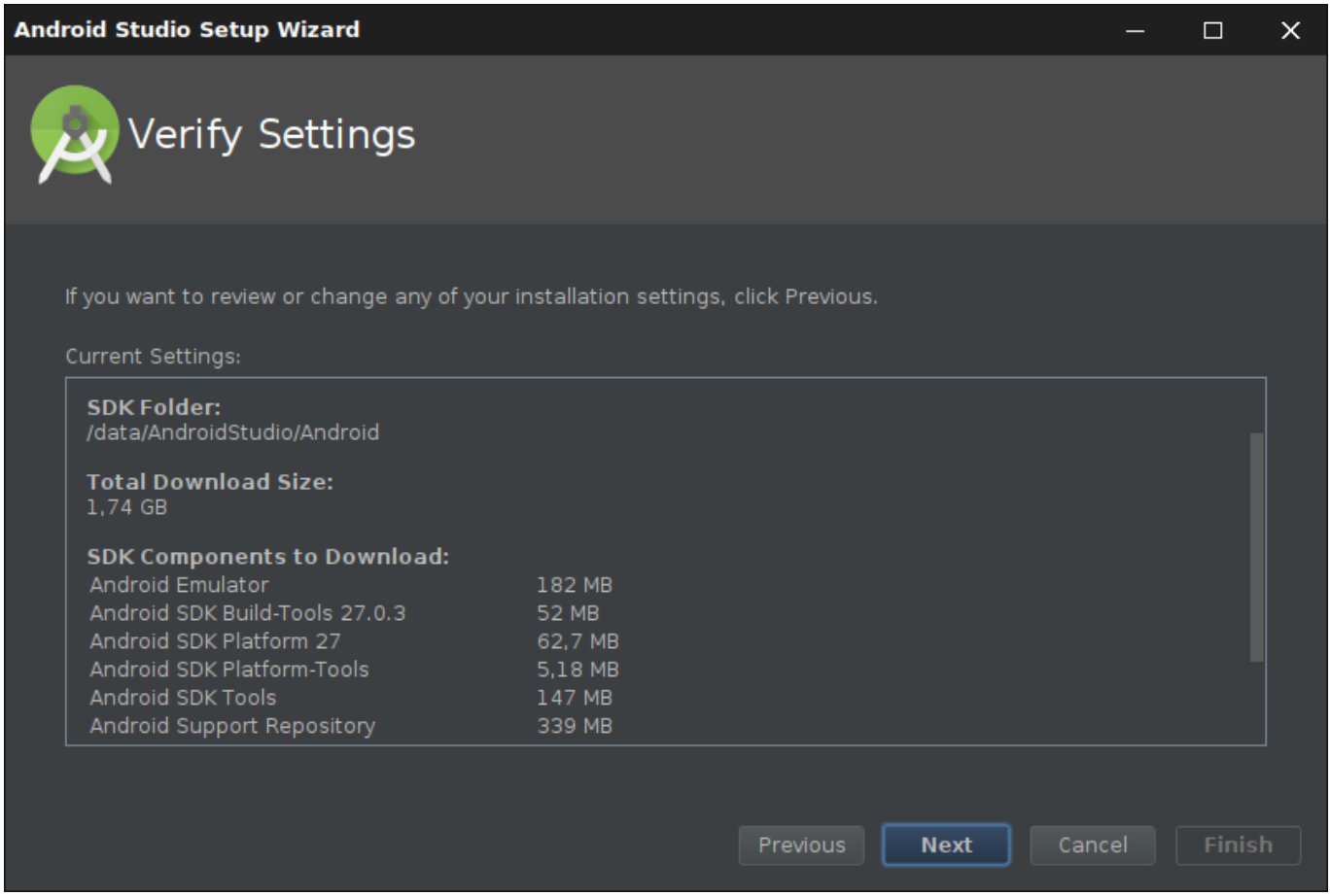


Im nächsten Schritt wird entschieden, welche zusätzlichen Komponenten mit installiert werden sollen:

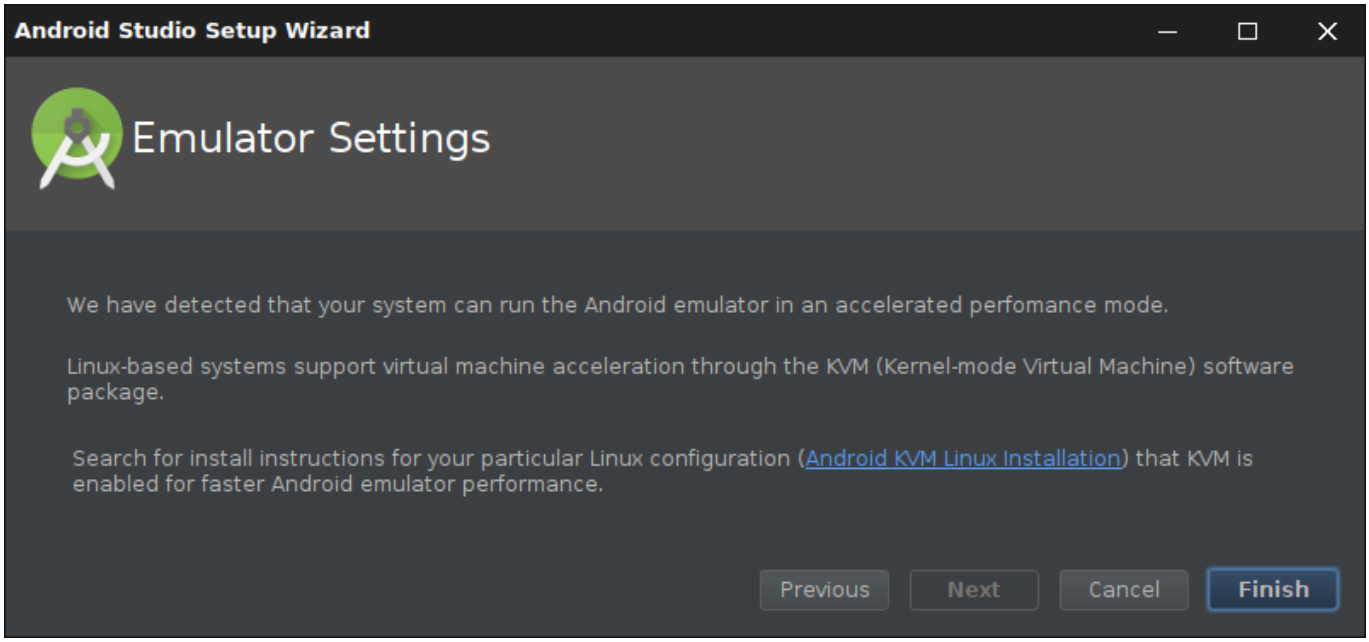


Hier wurde das Verzeichnis vom SDK in das Verzeichnis „`/data/`“ verschoben, damit nicht alles im Benutzerverzeichnis landet.

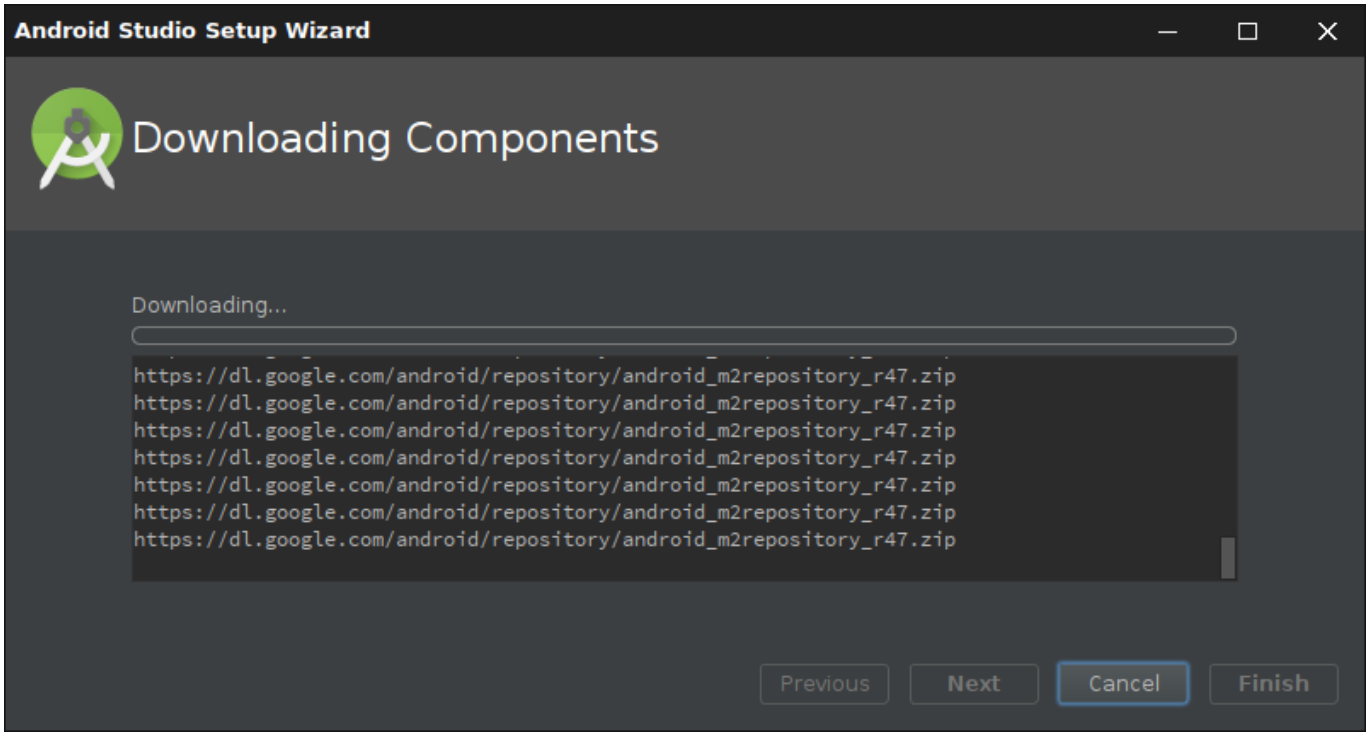
Das Programm zeigt an, wie viel Daten heruntergeladen werden müssen:



Es kann vorkommen, dass Android Studio einen Hinweis auf die Nutzung eines beschleunigten Emulatormodus hinweist:

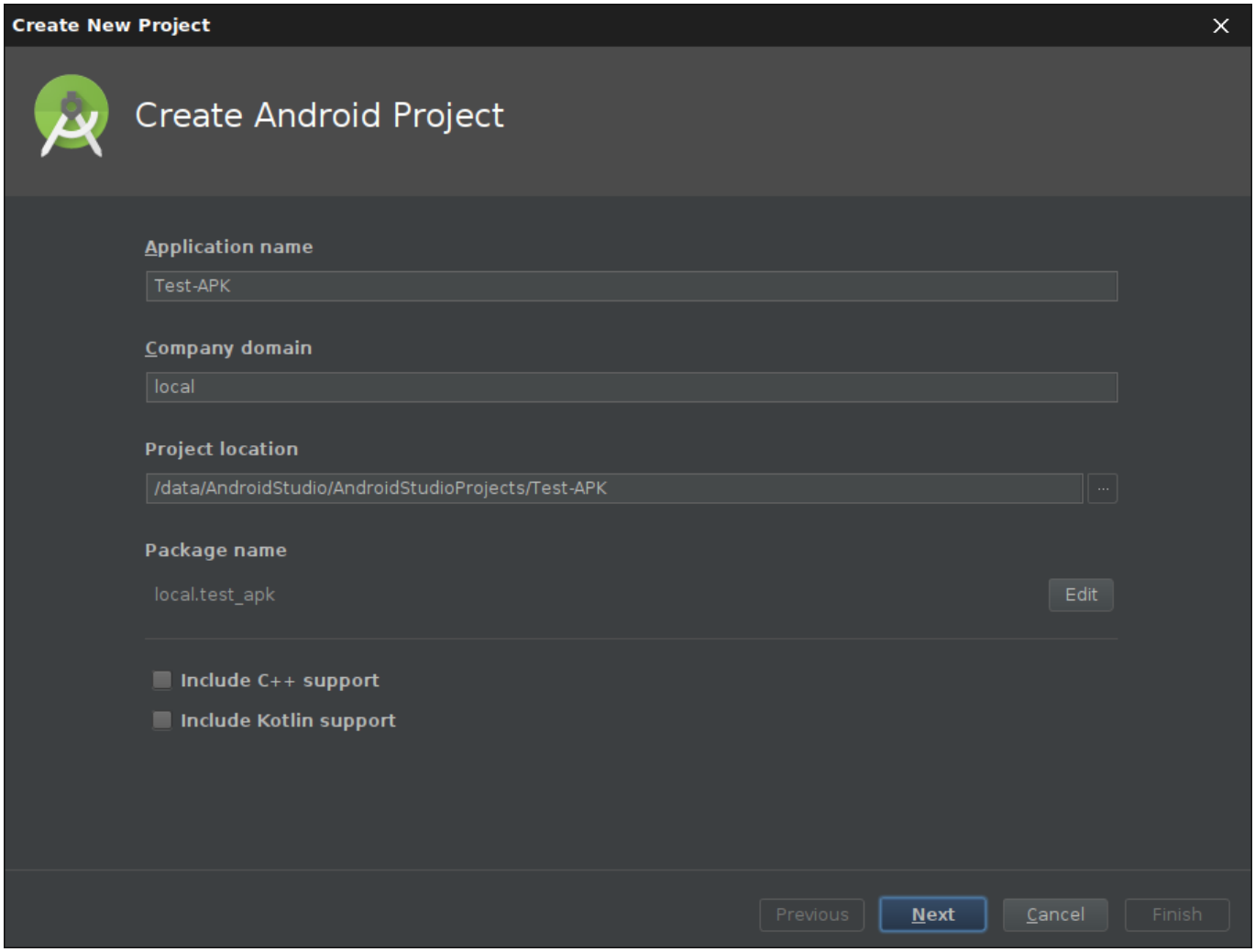


Nach einem Klick auf „`Finish`“ beginnt der Download, was, je nach Anbindung, einige Zeit dauern kann:



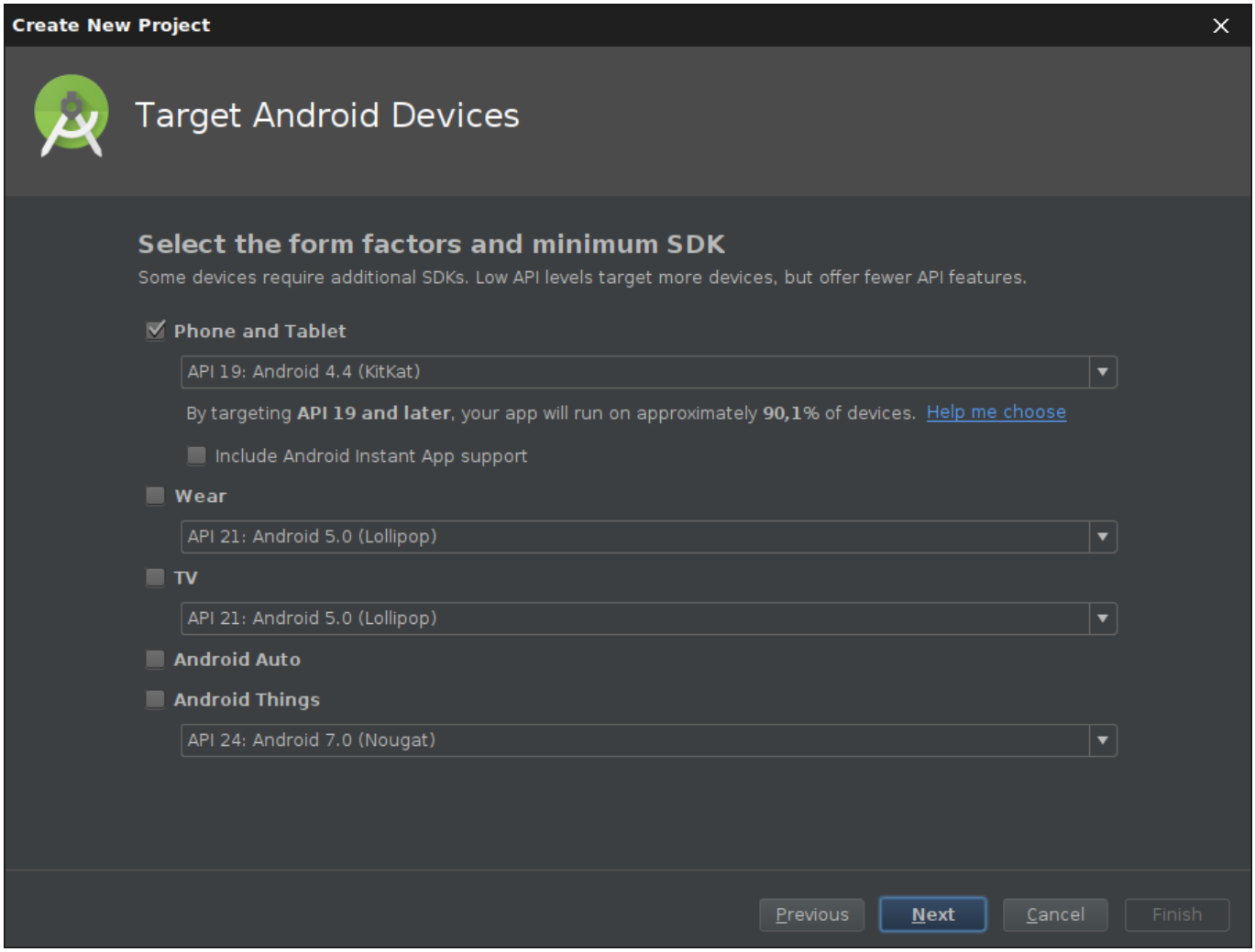
Nach dem Download kann dann ein vorhandenes Projekt geöffnet oder ein neues gestartet werden:

Für ein neues Projekt müssen einige Angaben gemacht werden:

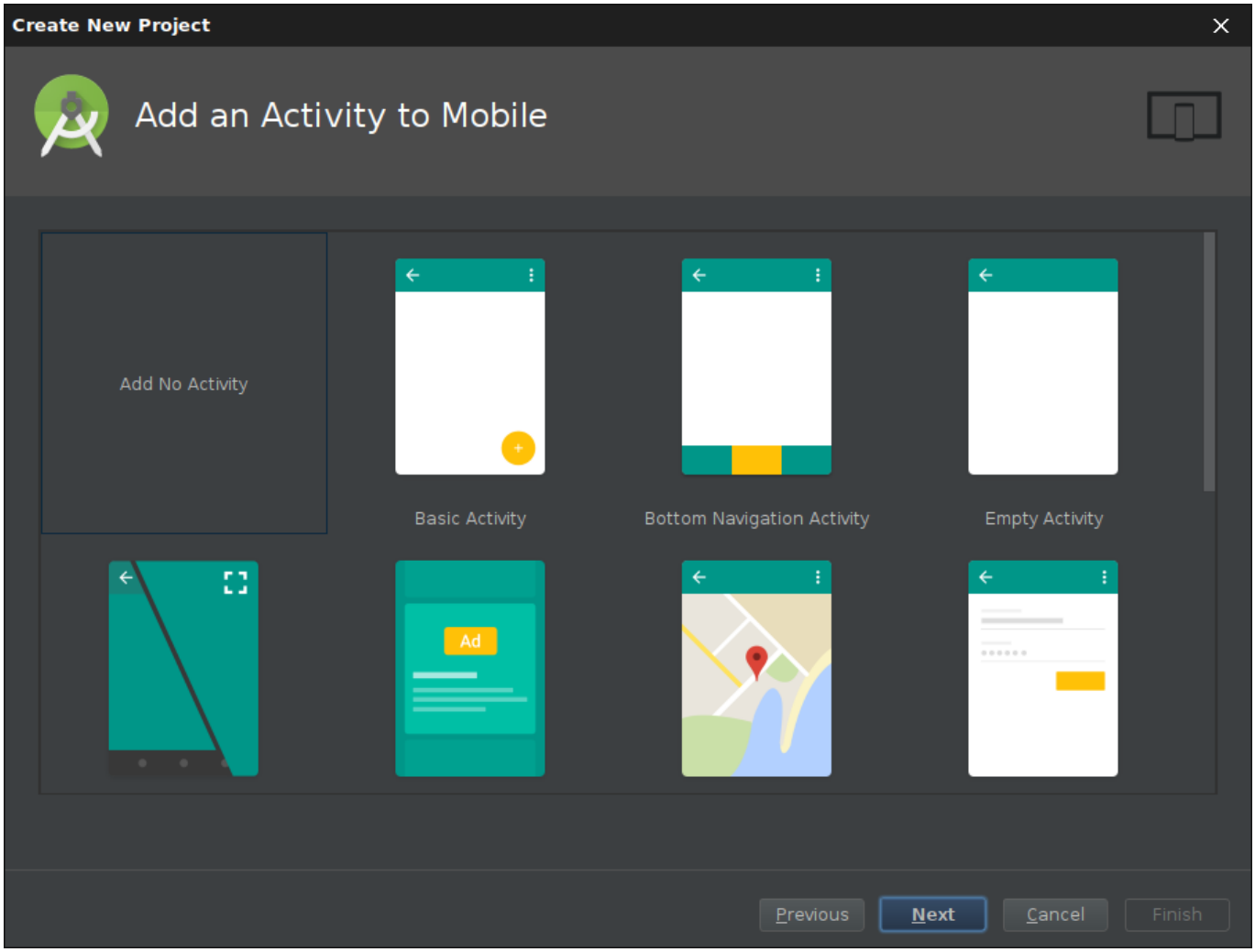


Auch hier wurde der Pfad für das Projekt nach „`/data`“ verschoben.

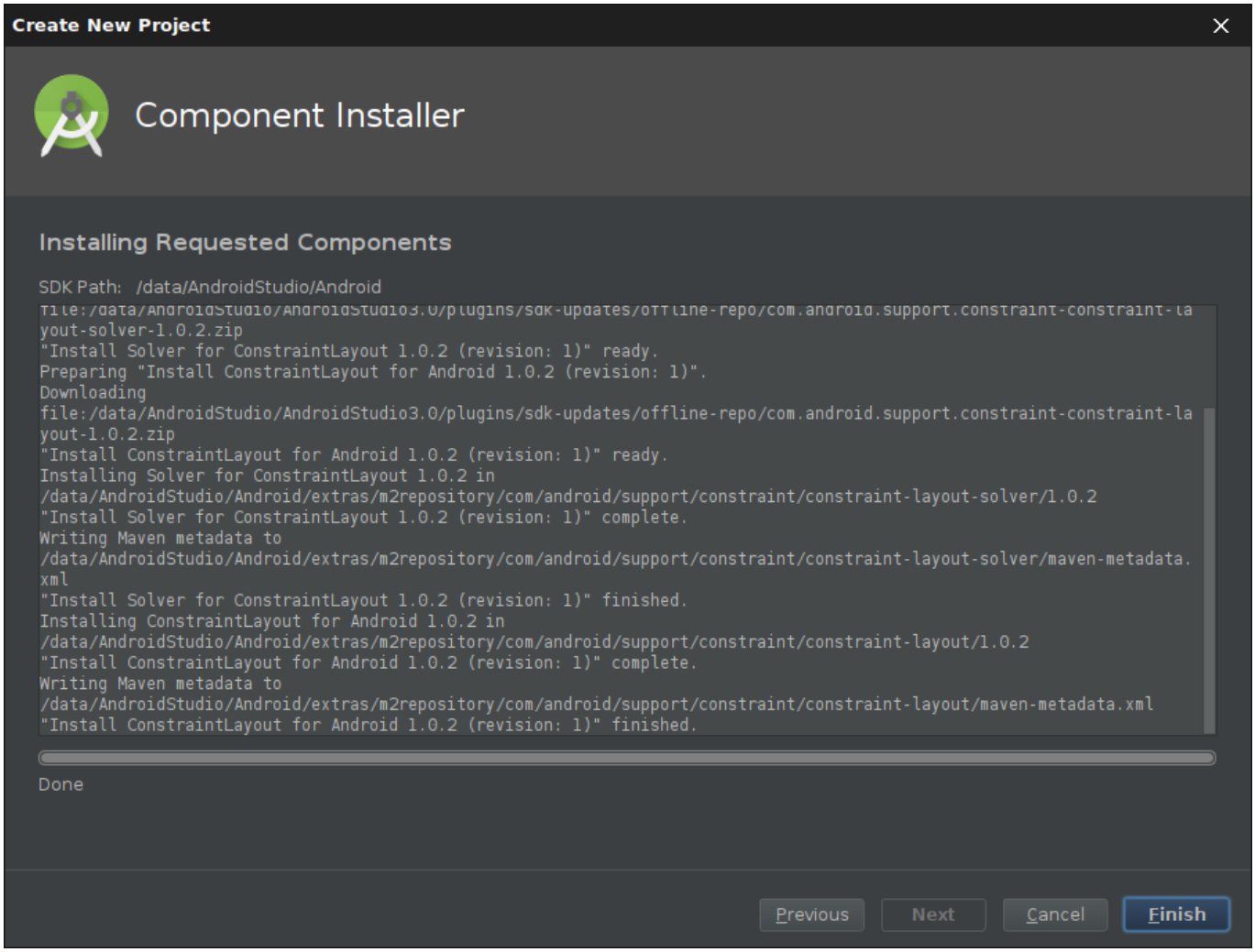
Im nächsten Schritt kann die Zielplattform ausgewählt werden:



Dann kann die Aktivität gewählt werden:



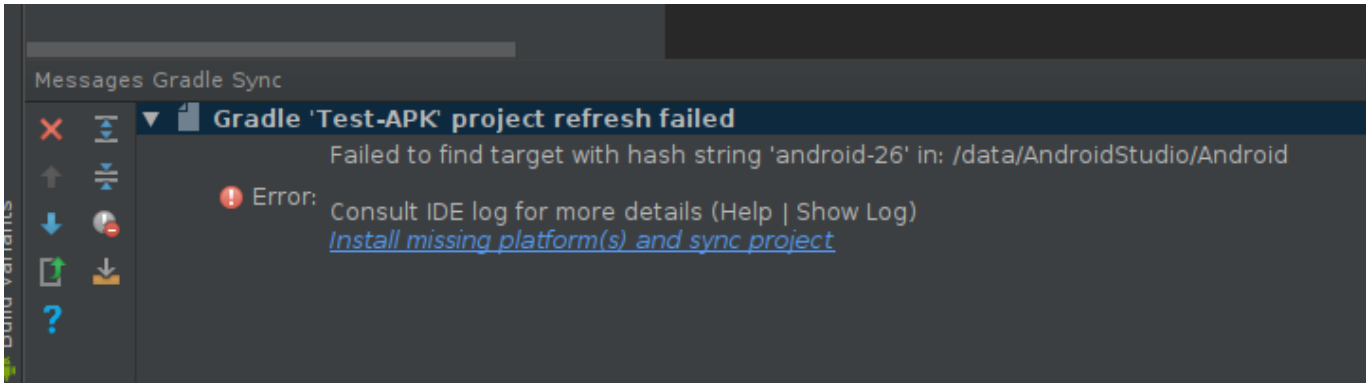
Die gewählte Zielplattform und die Aktivitäten werden installiert:



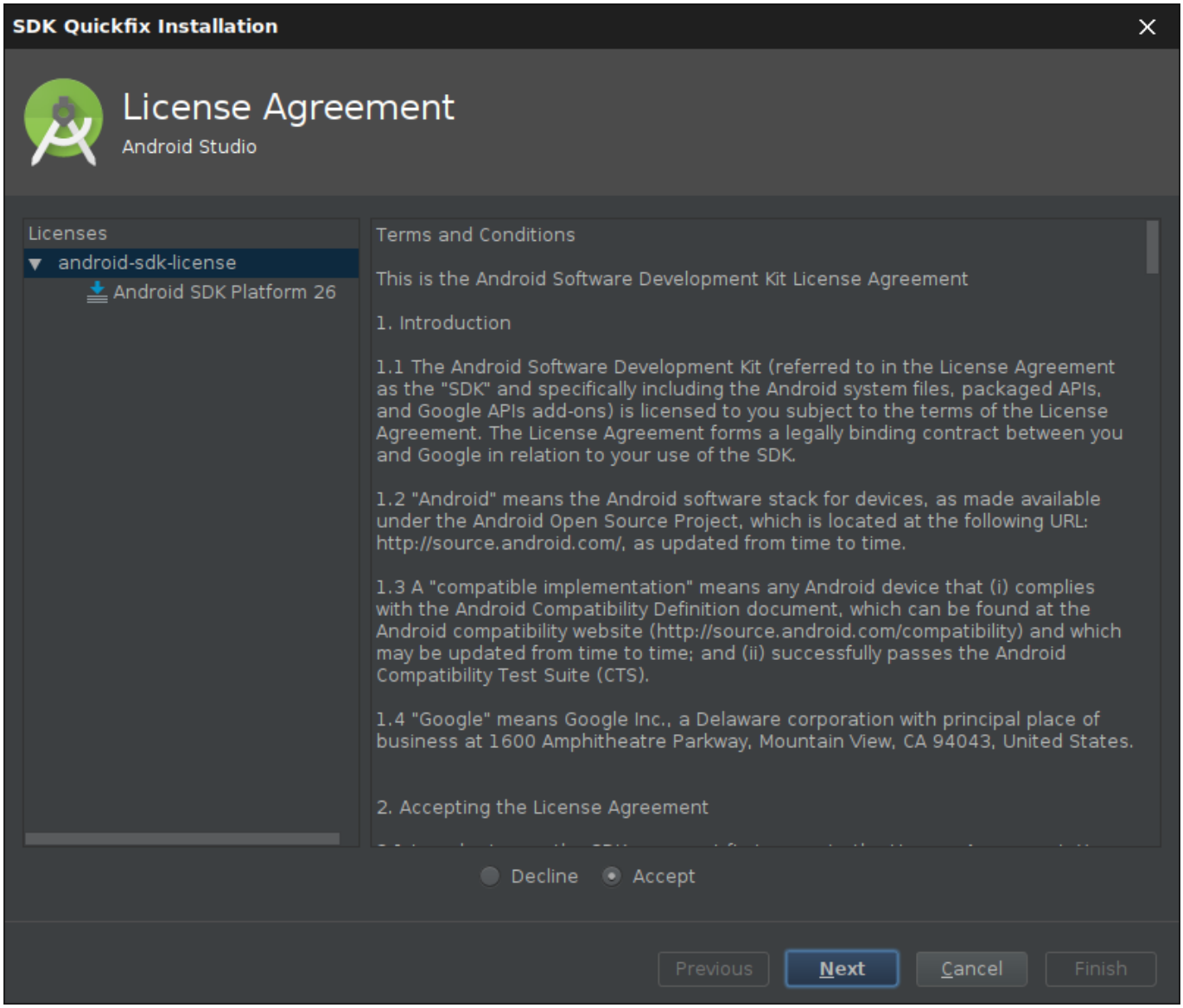
Dann beginnt automatisch das Bauen von Gradle:

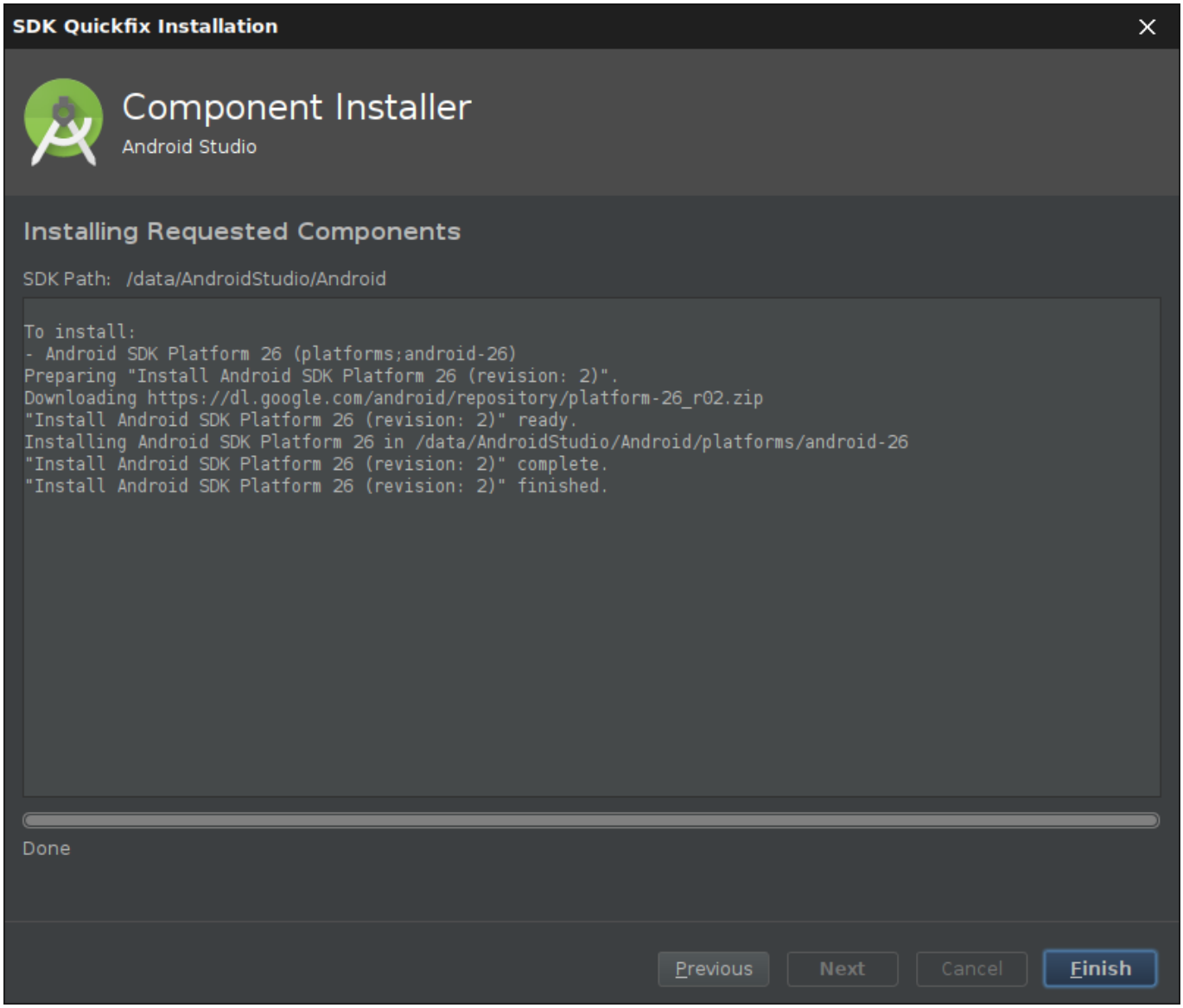
Im Hintergrund wird dafür das eingestellte Gradle-Paket aus dem Internet heruntergeladen, was einige Zeit in Anspruch nehmen kann.

Android Studio zeigt an, wenn weiter Komponenten nachinstalliert werden müssen:

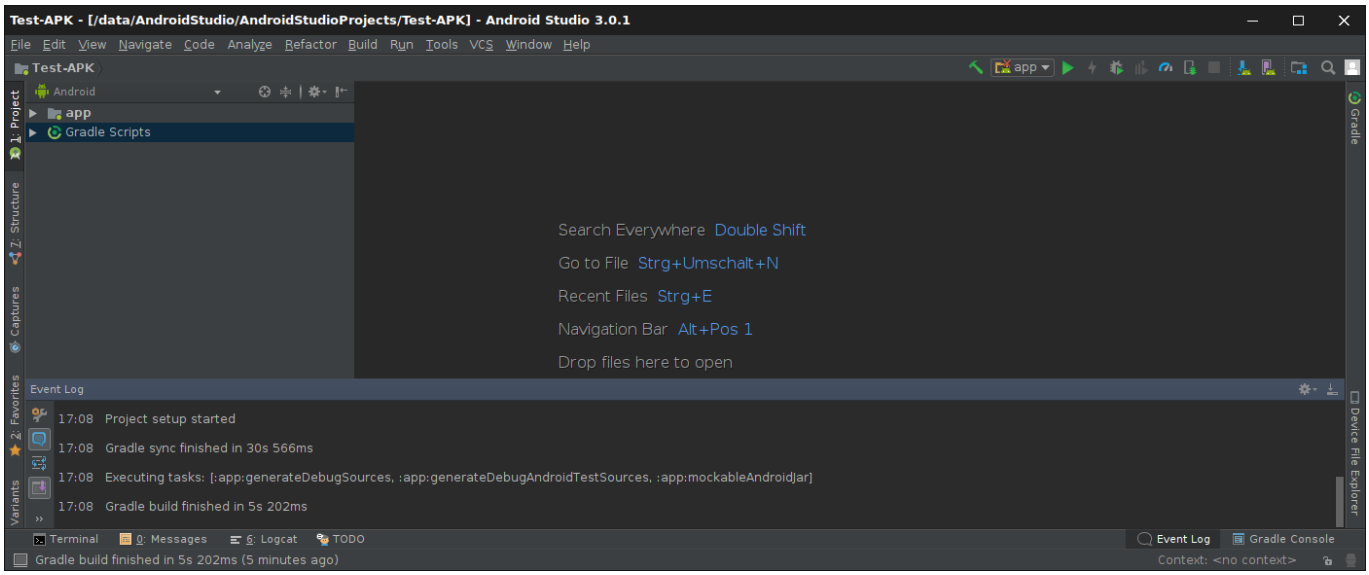


Klickt der Benutzer auf den blau hinterlegten Link, erfolgt automatisch ein Download. Bei Bedarf müssen die Lizenzbedingungen akzeptiert werden:


SDK Quickfix Installation
License Agreement
Android Studio
Licenses
android-sdk-license
Android SDK Platform 26
Terms and Conditions
This is the Android Software Development Kit License Agreement
1. Introduction
1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.
1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).
1.4 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.
2. Accepting the License Agreement
Decline
Accept
Previous
Next
Cancel
Finish



Nach dem Download aller fehlenden Komponenten, erscheint das Bearbeitungsfenster:



Verzeichnisse verschieben

Das Android Studio und seine zugehörigen Komponenten (zum Beispiel „`Gradle`“) legen die Daten im Verzeichnis des Benutzers ab. Da diese unter Umständen einigen Speicherplatz beanspruchen, erfolgt im nachfolgenden Fall eine Verschiebung in das Verzeichnis „`/data/AndroidStudio/`“.

Verschiebung von „`.android`“:

```
~$ cd
~$ mv .android /data/AndroidStudio/ && ln -s /data/AndroidStudio/.android
```

Verschiebung von „`.AndroidStudio3.0`“:

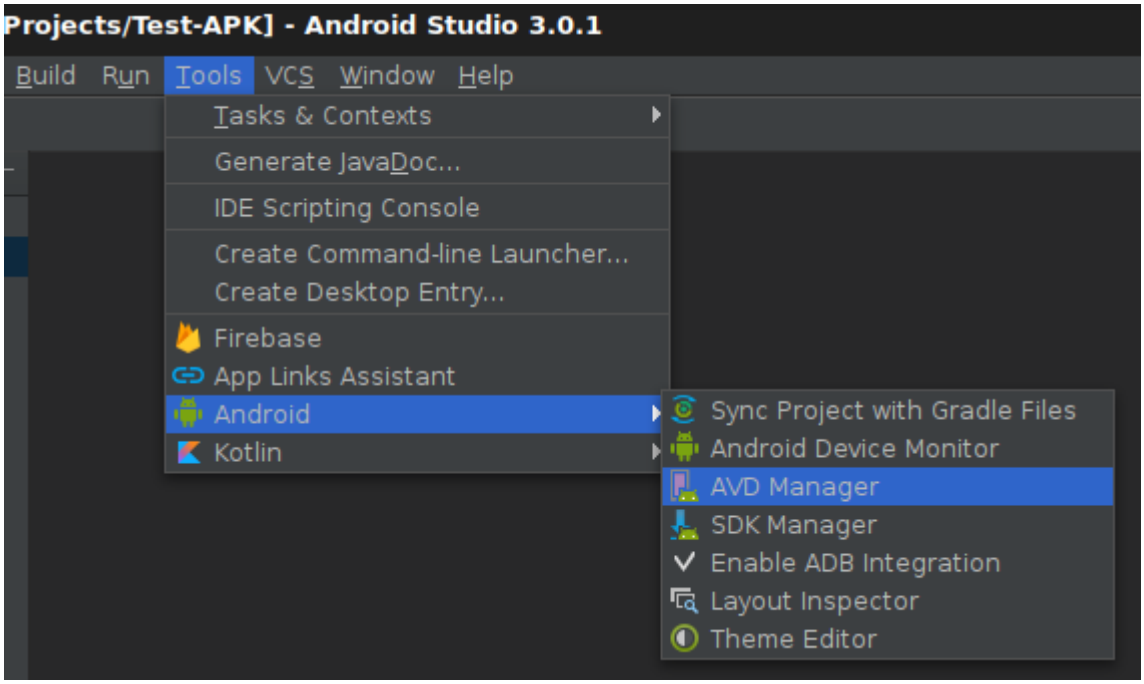
```
~$ cd
~$ mv .AndroidStudio3.0 /data/AndroidStudio/ && ln -s
/data/AndroidStudio/.AndroidStudio3.0
```

Verschiebung von „`.gradle`“:

```
~$ cd
~$ mv .gradle /data/AndroidStudio/ && ln -s /data/AndroidStudio/.gradle
```

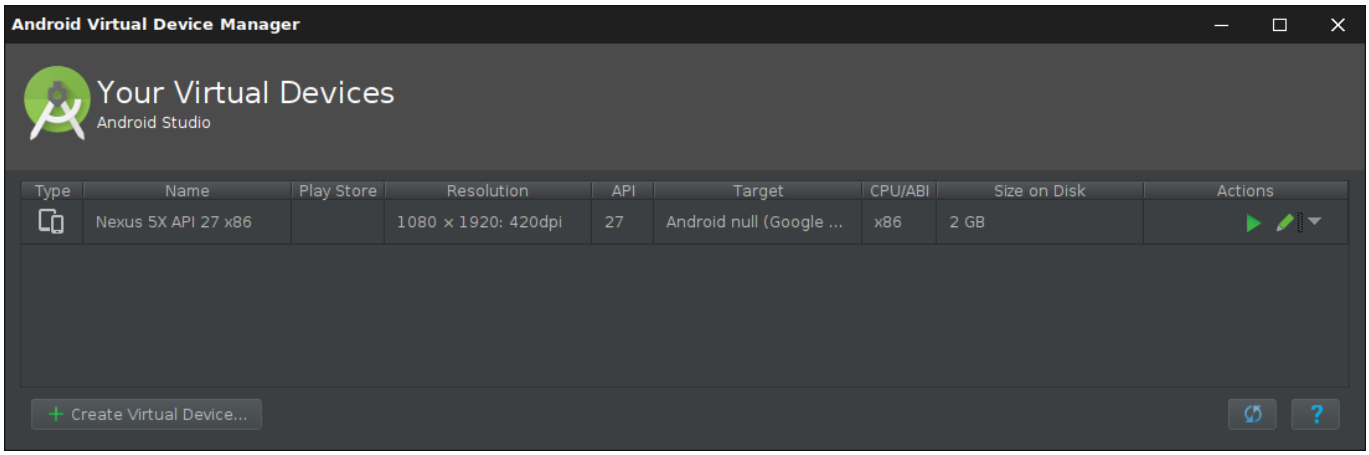
Emulator

Der Emulator, welcher ein Android-Gerät simuliert, kann wie folgt gestartet werden:



In der Menüleiste auf „`Tools`“ → „`Android`“ → „`AVD Manager`“ klicken.

Hier ist bereits ein Telefon vorinstalliert, welches durch Drücken auf den Abspielknopf (rechts unter „`Actions`“) gestartet werden kann:



Probleme

Im vorliegenden Fall startete der Emulator nicht und zeigte im Android Studio an, dass er Probleme mit dem Grafiktreiber („`Intel i965`“). Nach Recherche im Internet ergab sich, dass Android Studio Probleme mit der eigenen Bibliothek „`libstdc++.so.6`“ hatte. Hier half dann ein Verknüpfen mit der systemeigenen Bibliothek:

```
~$ cd /data/AndroidStudio/Android/emulator/lib64/libstdc++
~$ mv libstdc++.so.6 libstdc++.so.6.BAK
~$ ln -sf /usr/lib/x86_64-linux-gnu/libstdc++.so.6 libstdc++.so.6
```

Konsolenbefehle

Der Emulator kann auch gut über die Konsolenbefehle benutzt werden.

Auflisten aller installierten virtuellen Geräte:

```
~$ /data/AndroidStudio/Android/tools/bin/avdmanager list avd
Parsing /data/AndroidStudio/Android/build-tools/26.0.2/package.xmlParsing
/data/AndroidStudio/Android/build-tools/27.0.3/package.xmlParsing
/data/AndroidStudio/Android/emulator/package.xmlParsing
/data/AndroidStudio/Android/extras/android/m2repository/package.xmlParsing
/data/AndroidStudio/Android/extras/google/m2repository/package.xmlParsing
/data/AndroidStudio/Android/extras/m2repository/com/android/support/constrai
nt/constraint-layout-solver/1.0.2/package.xmlParsing
/data/AndroidStudio/Android/extras/m2repository/com/android/support/constrai
nt/constraint-layout/1.0.2/package.xmlParsing
/data/AndroidStudio/Android/patcher/v4/package.xmlParsing
/data/AndroidStudio/Android/platform-tools/package.xmlParsing
/data/AndroidStudio/Android/platforms/android-26/package.xmlParsing
/data/AndroidStudio/Android/platforms/android-27/package.xmlParsing
/data/AndroidStudio/Android/sources/android-27/package.xmlParsing
/data/AndroidStudio/Android/system-
images/android-27/google_apis/x86/package.xmlParsing
/data/AndroidStudio/Android/system-
images/android-27/google_apis_playstore/x86/package.xmlParsing
/data/AndroidStudio/Android/tools/package.xmlAvailable Android Virtual
Devices:
    Name: Nexus_5X_API_27_x86
  Device: Nexus 5X (Google)
    Path: /home/service/.android/avd/Nexus_5X_API_27_x86.avd
  Target: Google APIs (Google Inc.)
          Based on: Android API 27 Tag/ABI: google_apis/x86
    Skin: 1920x1080
  Sdcard: 800 MiB
```

Löschen der Datenpartition und anschließendes Starten des Gerätes:

```
~$ /data/AndroidStudio/Android/tools/emulator -avd <Gerätename> -wipe-data
```

Der Gerätename entspricht dabei dem „Namen“ bei den aufgelisteten Geräten.

Administrator-Zugriff

Es ist auch möglich, dass Gerät für Testzwecke zu „rooten“, damit Apps installiert werden können, die Administrator-Zugriffe benötigen. Es wird dabei nach der Anleitung von github.com vorgegangen.

Vorbereitung

Folgende Vorbereitungen sollten getroffen werden:

- wenn kein „adb“ vorhanden ist, die Android-SDK-werkzeuge „sdk-tools-linux-<Versionsnummer>.zip“ (zum Beispiel von der Hersteller-Seite)
- Herunterladen der APK-Datei „SuperSU“ in der aktuellen Version (zum Beispiel von der Hersteller-Seite)
- Herunterladen des installierbaren Recovery-Archives „Recovery Flashable.zip“ (Dateiname: „SuperSU-v<Versionsnummer>.zip“) in der gleichen Version (zum Beispiel ebenfalls von der Hersteller-Seite)

Die Android-SDK-Werkzeuge müssen entpackt und es muss der Zugriff auf das Programm „adb“ gewährleistet werden (zum Beispiel über Eintragen in die Pfadumgebungsvariable).

Das Recovery-Archiv „Recovery Flashable.zip“ muss in einem Unterverzeichnis entpackt werden:

```
~$ mkdir SuperSU
~$ unzip SuperSU-v<Versionsnummer>.zip -d SuperSU/
```

Emulator

Jetzt kann das emulierte Gerät eingerichtet werden, falls noch nicht geschehen. Wichtig ist, sich die verwendete Architektur (zum Beispiel „x86“ oder „arm64“) zu merken. Das Gerät wird dann über die Konsole mit beschreibbaren „system.img“ und einem ausgeschalteten „selinux“ gestartet:

```
~$ emulator -avd <Name-des-Gerätes> -writable-system -selinux permissive -
qemu -enable-kvm
emulator: WARNING: System image is writable
```

Nach dem Start des Gerätes kann die APK-Datei „SuperSU“ installiert werden:

```
~$ adb install supersu.apk
Success
```

Jetzt muss „adb“ als Administrator gestaretet werden:

```
~$ adb root && adb remount
restarting adbd as root
remount succeeded
```

Jetzt wird die originale „su“-Datei mit der aus dem heruntergeladen Archiv überschrieben:

```
~$ adb push SuperSU/<Architektur>/su.pie /system/xbin/su
SuperSU/x86/su.pie: 1 file pushed. 23.6 MB/s (104012 bytes in 0.004s)
```

Anpassen der Rechte:

```
~$ adb shell chmod 0755 /system/xbin/su
```

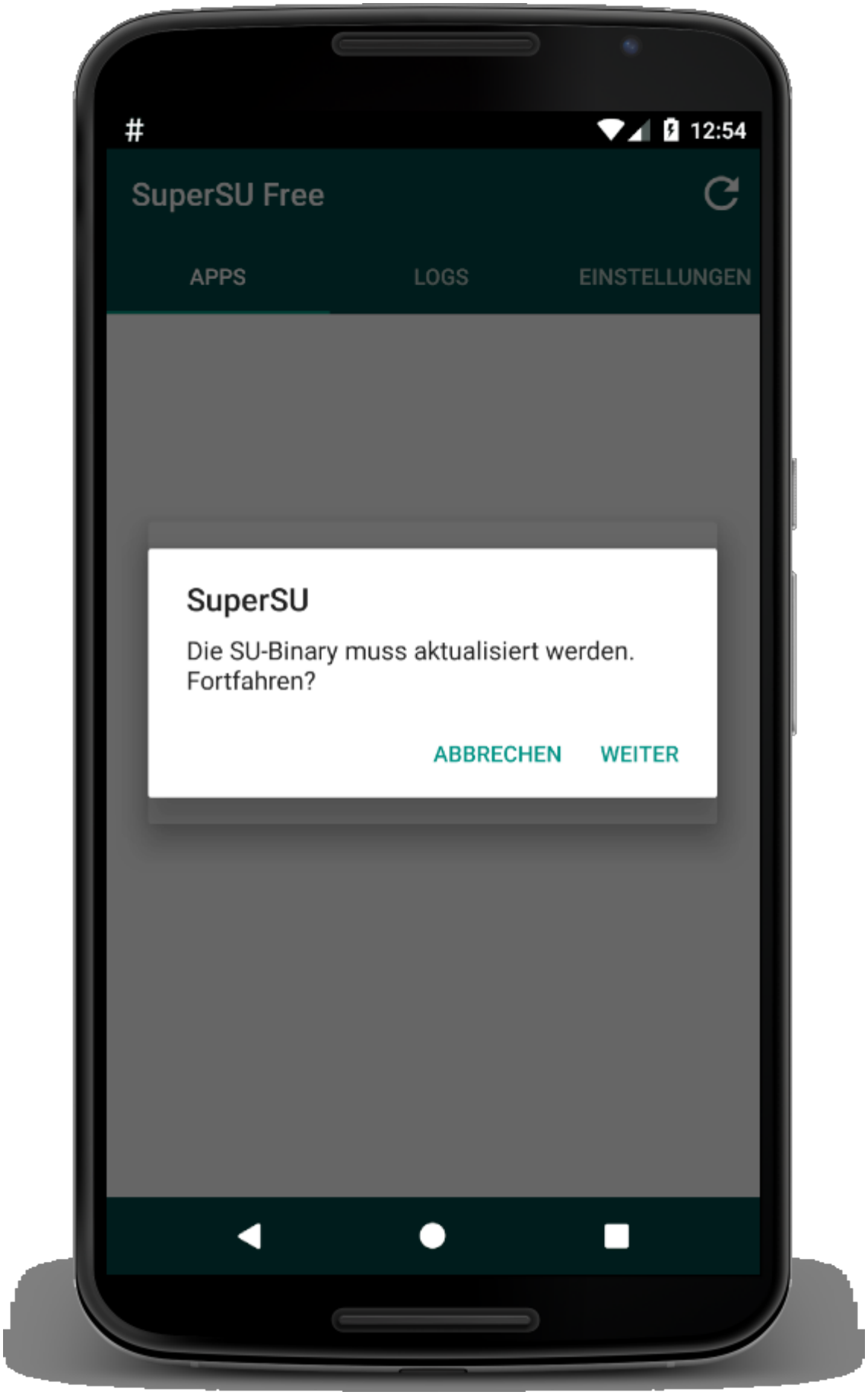
Deaktivieren von SELinux:

```
~$ adb shell setenforce 0
```

Das Program „su“ im System installieren und als Daemon starten:

```
~$ adb shell su --install && adb shell su --daemon&
[1] 23737
```

Wird jetzt die App „SuperSU“ auf dem Gerät gestartet, sollte es so aussehen:



Das Aktualisieren der App, was angeboten wird, hat bei mir bisher nicht geklappt.

Neustart

Leider überlebt diese Einstellung einen Neustart nicht bzw. mir fehlen noch die Informationen, wie ich das Ganze persistent machen kann. Bisher hilft es nur, dass Gerät wieder im beschreibbaren Modus und den Damon erneut zu starten:

```
~$ emulator -avd <Name-des-Gerätes> -writable-system -selinux permissive
emulator: WARNING: System image is writable
```

Nach dem Gerätestart dann folgendes über „adb“ ausführen:

```
~$ adb root && adb remount && adb shell su --daemon&
restarting adbd as root
remount succeeded
[2] 24385
```

— *Steffen Bornemann 2018/02/06*

DEBIAN, JAVA, AndroidStudio, Emulator, Konsole, SDK