BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG
-------o0o-------

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

Hải Phòng 2016

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG
−−−−−−−○0○−−−−−−−

XÂY DỰNG ỨNG DỤNG GAME ANDROID ĐOÁN LÁ BÀI ĐÃ CHỌN

ĐỒ ÁN TỐT NGHIỆP HỆ ĐẠI HỌC CHÍNH QUY

Ngành: Công nghệ Thông tin

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG
−−−−−−−o0o−−−−−−−

XÂY DỰNG ỨNG DỤNG GAME ANDROID ĐOÁN LÁ BÀI ĐÃ CHỌN

ĐỒ ÁN TỐT NGHIỆP HỆ ĐẠI HỌC CHÍNH QUY

Ngành: Công nghệ Thông tin

Sinh viên thực hiện: Đỗ Xuân Cường
Giáo viên hướng dẫn: Ths. Phùng Anh Tuấn
Mã số sinh viên: 120673

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc
-------o0o-------

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

Sinh viên: Đỗ Xuân Cường Mã số: 120673

Lớp: CT1201 Ngành: Công nghệ Thông tin

Tên đề tài: Xây dựng ứng dụng game Android đoán lá bài đã chọn

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

a. Nội dung:

- Tìm hiểu hệ điều hành Android.
- Tìm hiểu môi trường lập trình Android Studio.
- ứng dụng Android Studio để xây dựng ứng dụng game.

b. Các yêu cầu cần giải quyết:

- Nắm được một số khái niệm cơ bản của hệ điều hành Android.
- Tải và cài đặt môi trường lập trình ứng dụng cho thiết bị di động Android Studio.
- Tìm hiểu một số kỹ thuật lập trình game với các lá bài tú lơ khơ.
- Sử dụng công cụ lập trình Android Studio, các kỹ thuật lập trình xây dựng chương trình thực nghiệm game đoán lá bài bằng suy nghĩ của người chơi.
- Đóng gói ứng dụng thành bộ cài đặt cho phép cài đặt trực tiếp trên các thiết bị di động Android.

2. Các số liệu cần thiết để thiết kế, tính toán

- Số liệu giả lập

3. Địa điểm thực tập

Trường Đại học Dân lập Hải Phòng.

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Người hướng dẫn thứ nhất:

Họ và tên: Phùng Anh Tuấn

Học hàm, học vị: Thạc sỹ

Cơ quan công tác: Trường Đại học Dân lập Hải Phòng

Nội dung hướng dẫn:

- Tìm hiểu hệ điều hành Android.

- Tìm hiểu môi trường lập trình Android Studio.

- Tìm hiểu một số kỹ thuật lập trình game với các lá bài tú lơ khơ.

- ứng dụng Android Studio để xây dựng chương trình ứng dụng game đoán lá bài đã chọn bằng suy nghĩ của người chơi trên thiết bị động Android.

- Đóng gói chương trình ứng dụng game đoán lá bài cho phép tải về từ internet cài đặt trực tiếp trên thiết bị di động.

Người hướng dẫn thứ hai:

Họ và tên: …….....

Học hàm, học vị…….....

Cơ quan công tác: ……..

Nội dung hướng dẫn: …….....

……

Đề tài tốt nghiệp được giao ngày 03 tháng 10 năm 2016

Yêu cầu phải hoàn thành trước ngày 30 tháng 12 năm 2016

Đã nhận nhiệm vụ: Đ.T.T.N Sinh viên	Đã nhận nhiệm vụ: Đ.T.T.N Cán bộ hướng dẫn Đ.T.T.N

Hải Phòng, ngày .....tháng.....năm 2016

Hiệu trưởng

GS.TS.NGƯT Trần Hữu Nghị

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp:

- Có khả năng làm việc độc lập
- Cố gắng tìm hiểu tài liệu phục vụ cho nội dung của đề tài tốt nghiệp.
- Thực hiện đúng hạn các yêu cầu của cán bộ hướng dẫn.

2. Đánh giá chất lượng của đề tài tốt nghiệp (so với nội dung yêu cầu đã đề ra trong nhiệm vụ đề tài tốt nghiệp)

- *Về lý thuyết:*

+ Tổng hợp và nắm được một số khái niệm cơ bản của hệ điều hành Android.

+ Nắm được các bước xây dựng và cấu trúc chung của một ứng dụng android bằng công cụ Android Studio.

+ Nắm được một số kỹ thuật lập trình game với các lá bài tú lơ khơ cho thiết bị di động Android.

- *Về thực nghiệm:*

+ Cài đặt thành công môi trường lập trình Android Studio

+ Xây dựng thành công chương trình ứng dụng game đoán lá bài bằng suy nghĩ chạy trên thiết bị di động Android.

+ Đóng gói thành công chương trình ứng game cho phép tải về từ internet và cài đặt trực tiếp trên thiết bị di động Android.

3. Cho điểm của cán bộ hướng dẫn:

(Điểm ghi bằng số và chữ): 9.5 (chín điểm rưỡi)

Ngày.....tháng.....năm 2016
Cán bộ hướng dẫn chính
(Ký, ghi rõ họ tên)

PHẦN NHẬN XÉT ĐÁNH GIÁ CỦA CÁN BỘ CHẤM PHẢN BIỆN ĐỀ TÀI TỐT NGHIỆP

1. **Đánh giá chất lượng đề tài tốt nghiệp (về các mặt như cơ sở lý luận, thuyết minh chương trình, giá trị thực tế, ...)**

2. Cho điểm của cán bộ phản biện

(*Điểm ghi bằng số và chữ*)

.....

.....

Ngày.....tháng.....năm 2016

Cán bộ chấm phản biện

(Ký, ghi rõ họ tên)

Mục lục

LỜI CẢM ƠN

Em xin chân thành cảm ơn các thầy cô khoa Công Nghệ Thông Tin trường Đại học Dân Lập Hải Phòng đã quan tâm, dạy dỗ, chỉ bảo tận tình trong suốt thời gian học tập tài trường. Cho đến nay em đã có thể hoàn thành luận văn, đề tài: "Xây dựng ứng dụng game Android đoán lá bài đã chọn".

Em xin gửi lời cảm ơn tới giảng viên, Thạc sĩ Phùng Anh Tuấn – người trực tiếp hướng dẫn và chỉ bảo cho em hoàn thành đề tài tốt nghiệp này.

Em xin chân thành cảm ơn!

Hải Phòng, ngày 24 tháng 10 năm 2016

Sinh viên

Đỗ Xuân Cường

Chương 1 HỆ ĐIỀU HÀNH ANDROID

Android là một hệ điều hành dựa trên nền tảng Linux được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Ban đầu, Android được phát triển bởi Tổng công ty Android, với sự hỗ trợ tài chính từ Google và sau này được chính Google mua lại vào năm 2005. Android ra mắt vào năm 2007 cùng với tuyên bố thành lập Liên minh thiết bị cầm tay mở: một hiệp hội gồm các công ty phần cứng, phần mềm, và viễn thông với mục tiêu đẩy mạnh các tiêu chuẩn mở cho các thiết bị di động. Chiếc điện thoại đầu tiên chạy Android được bán vào tháng 10 năm 2008.

Android có mã nguồn mở và Google phát hành mã nguồn theo Giấy phép Apache. Chính mã nguồn mở cùng với một giấy phép không có nhiều ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên nhiệt huyết được điều chỉnh và phân phối Android một cách tự do. Ngoài ra, Android còn có một cộng đồng lập trình viên đông đảo chuyên viết các ứng dụng để mở rộng chức năng của thiết bị, bằng một loại ngôn ngữ lập trình Java có sửa đổi. Vào tháng 10 năm 2012, có khoảng 700.000 ứng dụng trên Android, và số lượt tải ứng dụng từ Google Play, cửa hàng ứng dụng chính của Android, ước tính khoảng 25 tỷ lượt.

Những yếu tố này đã giúp Android trở thành nền tảng điện thoại thông minh phổ biến nhất thế giới, vượt qua Symbian vào quý 4 năm 2010, và được các công ty công nghệ lựa chọn khi họ cần một hệ điều hành không nặng nề, có khả năng tinh chỉnh, và giá rẻ chạy trên các thiết bị công nghệ cao thay vì tạo dựng từ đầu. Kết quả là mặc dù được thiết kế để chạy trên điện thoại và máy tính bảng, Android đã xuất hiện trên TV, máy chơi game và các thiết bị điện tử khác. Bản chất mở của Android cũng khích lệ một đội ngũ đông đảo lập trình viên và những người đam mê sử dụng mã nguồn mở để tạo ra những dự án do cộng đồng quản lý. Những dự án này bổ sung các tính năng cao cấp cho những người dùng thích tìm tòi hoặc đưa Android vào các thiết bị ban đầu chạy hệ điều hành khác.

Android chiếm 75% thị phần điện thoại thông minh trên toàn thế giới vào thời điểm quý 3 năm 2012, với tổng cộng 500 triệu thiết bị đã được kích hoạt và 1,3 triệu lượt kích hoạt mỗi ngày. Sự thành công của hệ điều hành cũng khiến nó trở

thành mục tiêu trong các vụ kiện liên quan đến bằng phát minh, góp mặt trong cái gọi là "cuộc chiến điện thoại thông minh" giữa các công ty công nghệ.

1.1Tổng quan

> Android - nền tảng di động
> phổ biến nhất trên thế giới

Android được cài đặt trên hàng trăm triệu các thiết bị di động tại hơn 190 quốc gia trên toàn thế giới. Đó là hệ thống cơ sở được cài đặt nhiều nhất trên nhiều nền tảng điện thoại di động nào và phát triển với tốc độ rất nhanh - mỗi ngày có một triệu người khởi động thiết bị Android của họ lần đầu tiên và bắt đầu tìm kiếm các ứng dụng, trò chơi và nội dung kỹ thuật số khác.

Android cung cấp cho bạn một nền tảng đẳng cấp thế giới để tạo ra các ứng dụng và trò chơi dành cho người dùng Android ở khắp mọi nơi, cũng như một thị trường mở để phân phối cho họ ngay lập tức.

1.1.1Quan hệ đối tác toàn cầu và cơ sở cài đặt lớn

Xây dựng trên sự đóng góp của cộng đồng mã nguồn mở Linux và hơn 300 đối tác về phần cứng, phần mềm và các hãng cung cấp dịch vụ di động, Android đã nhanh chóng trở thành hệ điều hành điện thoại di động phát triển nhanh nhất.

> Mỗi ngày có hơn một triệu thiết bị Android
> được kích hoạt mới trên toàn thế giới

Sự cởi mở của Android đã làm cho nó trở thành sự yêu thích cho người tiêu dùng và các nhà phát triển, thúc đẩy tăng trưởng mạnh trong tiêu dùng ứng dụng. Người dùng Android tải về hàng tỷ các ứng dụng và trò chơi từ Google Play mỗi tháng.

Với các đối tác của mình, Android liên tục đẩy ranh giới của phần cứng và phần mềm về phía trước để mang lại khả năng mới cho người dùng và các nhà phát triển. Đối với các nhà phát triển, sự đổi mới của Android cho phép bạn xây dựng các ứng dụng mạnh mẽ, khác biệt với việc sử dụng các công nghệ di động mới nhất.

1.1.2Đổi mới nhanh chóng

Android đang tiếp tục đẩy ranh giới của phần cứng và phần mềm về phía trước, để mang lại khả năng mới cho người dùng và các nhà phát triển. Đối với các nhà phát triển, sự phát triển nhanh chóng của công nghệ Android cho phép bạn luôn ở phía trước với các ứng dụng mạnh mẽ và khác biệt.

Android cung cấp cho bạn truy cập vào các công nghệ mới nhất và sáng tạo qua vô số các hình thức của thiết bị, kiến trúc chipset, và mức giá. Từ xử lý đa lõi và đồ họa hiệu suất cao đến các công nghệ cảm biến tiên tiến, màn hình cảm ứng rực rỡ, và các công nghệ di động mới nổi.

1.1.3Framework phát triển mạnh mẽ

Android cung cấp cho bạn mọi thứ bạn cần để xây dựng các ứng dụng hàng đầu. Nó cung cấp cho bạn một mô hình ứng dụng duy nhất cho phép bạn triển khai các ứng dụng của bạn một cách rộng rãi cho hàng trăm triệu người dùng trên một loạt các thiết bị, từ điện thoại tới máy tính bảng và xa hơn nữa.

Android cũng cung cấp cho bạn các công cụ để tạo ra các ứng dụng tuyệt vời và tận dụng lợi thế của các tính năng phần cứng có sẵn trên mỗi thiết bị. Nó tự động điều chỉnh giao diện người dùng của bạn để nhìn đẹp nhất trên mỗi thiết bị, trong khi đem lại cho bạn nhiều quyền kiểm soát nhất mà bạn muốn qua giao diện người dùng của bạn trên các loại thiết bị khác nhau.

Ví dụ, bạn có thể tạo ra một ứng dụng duy nhất được tối ưu hóa cho cả điện thoại và tablet. Bạn khai báo giao diện người dùng của bạn trong bộ tài nguyên XML, một phần thiết đặt cho giao diện người dùng (UI) chung cho tất cả các loại máy và một phần khác thiết đặt tối ưu riêng cho điện thoại hoặc máy tính

bảng. Khi chạy, Android áp dụng các bộ nguồn chính xác dựa trên kích thước màn hình của nó, mật độ điểm ảnh và loại máy.

Để giúp bạn phát triển một cách hiệu quả, các công cụ phát triển Android cung cấp đầy đủ Java IDE với các tính năng tiên tiến để phát triển, gỡ lỗi, và đóng gói các ứng dụng Android. Sử dụng các IDE, bạn có thể phát triển trên bất kỳ thiết bị Android có sẵn hoặc tạo ra các thiết bị ảo - có thể mô phỏng bất kỳ cấu hình phần cứng nào.

1.1.4Thị trường mở để phân phối các ứng dụng của bạn

Google Play là thị trường hàng đầu cho việc bán và phân phối các ứng dụng Android. Khi bạn xuất bản một ứng dụng trên Google Play, bạn có cơ hội tiếp cận đến một lượng lớn thiết bị Android.

Là một thị trường mở, Google Play trao cho bạn cách bạn bán sản phẩm của mình. Bạn có thể xuất bản bất cứ khi nào bạn muốn, với giá tiền mà bạn muốn, và đến đối tượng khách hàng bạn muốn. Bạn có thể phân phối rộng rãi cho tất cả các thị trường và các thiết bị hoặc tập trung vào phân khúc cụ thể, các thiết bị, hoặc phạm vi phần cứng.

Bạn có thể kiếm tiền theo cách phù hợp nhất cho công ty của bạn - trả phí hoặc miễn phí, bán sản phẩm trong ứng dụng hoặc đăng ký - để được tỷ lệ tương tác và doanh thu cao nhất. Bạn cũng có thể kiểm soát hoàn toàn việc định giá cho các ứng dụng của bạn cũng như sản phẩm bán trong ứng dụng, và có thể thiết lập hoặc thay đổi giá bán thuộc bất kỳ loại tiền được hỗ trợ, vào bất cứ lúc nào.

Ngoài phát triển cơ sở khách hàng của bạn, Google Play sẽ giúp bạn xây dựng tầm nhìn và cam kết trên các ứng dụng và thương hiệu của bạn. Ví dụ như các ứng dụng của bạn tăng mức độ phổ biến, Google Play cung cấp cho nó vị trí cao hơn trong bảng các bảng xếp hạng.

Được cài đặt sẵn trên hàng tỷ thiết bị Android trên toàn thế giới, Google Play có thể là một động cơ tăng trưởng cho doanh nghiệp của bạn.

1.2Lịch sử

Tổng công ty Android (Android, Inc.) được thành lập tại Palo Alto, California vào tháng 10 năm 2003 bởi Andy Rubin (đồng sáng lập công ty Danger), Rich Miner (đồng sáng lập Tổng công ty Viễn thông Wildfire), Nick Sears (từng là Phó giám đốc T-Mobile), và Chris White (trưởng thiết kế và giao diện tại WebTV) để phát triển, theo lời của Rubin, "các thiết bị di động thông minh hơn có thể biết được vị trí và sở thích của người dùng". Dù những người thành lập và nhân viên đều là những người có tiếng tăm, Tổng công ty Android hoạt động một cách âm thầm, chỉ tiết lộ rằng họ đang làm phần mềm dành cho điện thoại di động. Trong năm đó, Rubin hết kinh phí. Steve Perlman, một người bạn thân của Rubin, mang cho ông 10.000 USD tiền mặt nhưng từ chối tham gia vào công ty.

Google mua lại Tổng công ty Android vào ngày 17 tháng 8 năm 2005, biến nó thành một bộ phận trực thuộc Google. Những nhân viên của chủ chốt của Tổng công ty Android, gồm Rubin, Miner và White, vẫn tiếp tục ở lại công ty làm việc sau thương vụ này. Vào thời điểm đó không có nhiều thông tin về Tổng công ty, nhưng nhiều người đồn đoán rằng Google dự tính tham gia thị trường điện thoại di động sau bước đi này. Tại Google, nhóm do Rubin đứng đầu đã phát triển một nền tảng thiết bị di động phát triển trên nền nhân Linux. Google quảng bá nền tảng này cho các nhà sản xuất điện thoại và các nhà mạng với lời hứa sẽ cung cấp một hệ thống uyển chuyển và có khả năng nâng cấp. Google đã liên hệ với hàng loạt hãng phần cứng cũng như đối tác phần mềm, bắn tin cho các nhà mạng rằng họ sẵn sàng hợp tác với các cấp độ khác nhau.

Ngày càng nhiều suy đoán rằng Google sẽ tham gia thị trường điện thoại di động xuất hiện trong tháng 12 năm 2006. Tin tức của BBC và Nhật báo phố Wall chú thích rằng Google muốn đưa công nghệ tìm kiếm và các ứng dụng của họ vào điện thoại di động và họ đang nỗ lực làm việc để thực hiện điều này. Các phương tiện truyền thông truyền thống lẫn online cũng viết về tin đồn rằng Google đang phát triển một thiết bị cầm tay mang thương hiệu Google. Một vài tờ báo còn nói rằng trong khi Google vẫn đang thực hiện những bản mô tả kỹ thuật chi tiết, họ đã trình diễn sản phẩm mẫu cho các nhà sản xuất điện thoại di động và nhà mạng. Tháng 9 năm 2007, InformationWeek đăng tải một nghiên cứu của Evalueserve cho biết Google đã nộp một số đơn xin cấp bằng sáng chế trong lĩnh vực điện thoại di động.

Ngày 5 tháng 11 năm 2007, Liên minh thiết bị cầm tay mở (Open Handset Alliance), một hiệp hội bao gồm nhiều công ty trong đó có Texas Instruments, Tập đoàn Broadcom, Google, HTC, Intel, LG, Tập đoàn Marvell Technology, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel và T-Mobile được thành lập với mục đích phát triển các tiêu chuẩn mở cho thiết bị di động. Cùng ngày, Android cũng được ra mắt với vai trò là sản phẩm đầu tiên của Liên minh, một nền tảng thiết bị di động được xây dựng trên nhân Linux phiên bản 2.6. Chiếc điện thoại chạy Android đầu tiên được bán ra là HTC Dream, phát hành ngày 22 tháng 10 năm 2008. Biểu trưng của hệ điều hành Android mới là một con rôbốt màu xanh lá cây do hãng thiết kế Irina Blok tại California vẽ.

Từ năm 2008, Android đã trải qua nhiều lần cập nhật để dần dần cải tiến hệ điều hành, bổ sung các tính năng mới và sửa các lỗi trong những lần phát hành trước. Mỗi bản nâng cấp được đặt tên lần lượt theo thứ tự bảng chữ cái, theo tên của một món ăn tráng miệng; ví dụ như phiên bản 1.5 Cupcake (bánh bông lan nhỏ có kem) tiếp nối bằng phiên bản 1.6 Donut (bánh vòng). Phiên bản mới nhất hiện nay là 5.0 Lollipop. Vào năm 2010, Google ra mắt loạt thiết bị Nexus - một dòng sản phẩm bao gồm điện thoại thông minh và máy tính bảng chạy hệ điều hành Android, do các đối tác phần cứng sản xuất. HTC đã hợp tác với Google trong chiếc điện thoại thông minh Nexus đầu tiên, Nexus One. Kể từ đó nhiều thiết bị mới hơn đã gia nhập vào dòng sản phẩm này, như điện thoại Nexus 4 và máy tính bảng Nexus 10, lần lượt do LG và Samsung sản xuất. Google xem điện thoại và máy tính bảng Nexus là những thiết bị Android chủ lực của mình, với những tính năng phần cứng và phần mềm mới nhất của Android.

1.3Các tính năng

1.3.1Giao diện

Giao diện người dùng của Android dựa trên nguyên tắc tác động trực tiếp, sử dụng cảm ứng chạm tương tự như những động tác ngoài đời thực như vuốt, chạm, kéo giãn và thu lại để xử lý các đối tượng trên màn hình. Sự phản ứng với tác động của người dùng diễn ra gần như ngay lập tức, nhằm tạo ra giao diện cảm ứng mượt mà, thường dùng tính năng rung của thiết bị để tạo phản hồi rung cho người dùng. Những thiết bị phần cứng bên trong như gia tốc kế, con quay hồi chuyển và cảm biến khoảng cách được một số ứng dụng sử dụng để phản hồi một số hành động khác của người dùng, ví dụ như điều chỉnh màn hình từ chế độ hiển thị dọc sang chế độ hiển thị ngang tùy theo vị trí của thiết bị, hoặc cho

phép người dùng lái xe đua bằng xoay thiết bị, giống như đang điều khiển vô-lăng.

Các thiết bị Android sau khi khởi động sẽ hiển thị màn hình chính, điểm khởi đầu với các thông tin chính trên thiết bị, tương tự như khái niệm desktop (bàn làm việc) trên máy tính để bàn. Màn hình chính Android thường gồm nhiều biểu tượng (icon) và tiện ích (widget); biểu tượng ứng dụng sẽ mở ứng dụng tương ứng, còn tiện ích hiển thị những nội dung sống động, cập nhật tự động như dự báo thời tiết, hộp thư của người dùng, hoặc những mẩu tin thời sự ngay trên màn hình chính. Màn hình chính có thể gồm nhiều trang xem được bằng cách vuốt ra trước hoặc sau, mặc dù giao diện màn hình chính của Android có thể tùy chỉnh ở mức cao, cho phép người dùng tự do sắp đặt hình dáng cũng như hành vi của thiết bị theo sở thích. Những ứng dụng do các hãng thứ ba có trên Google Play và các kho ứng dụng khác còn cho phép người dùng thay đổi "chủ đề" của màn hình chính, thậm chí bắt chước hình dáng của hệ điều hành khác như Windows Phone chẳng hạn. Phần lớn những nhà sản xuất, và một số nhà mạng, thực hiện thay đổi hình dáng và hành vi của các thiết bị Android của họ để phân biệt với các hãng cạnh tranh.

Ở phía trên cùng màn hình là thanh trạng thái, hiển thị thông tin về thiết bị và tình trạng kết nối. Thanh trạng thái này có thể "kéo" xuống để xem màn hình thông báo gồm thông tin quan trọng hoặc cập nhật của các ứng dụng, như email hay tin nhắn SMS mới nhận, mà không làm gián đoạn hoặc khiến người dùng cảm thấy bất tiện. Trong các phiên bản đời đầu, người dùng có thể nhấn vào thông báo để mở ra ứng dụng tương ứng, về sau này các thông tin cập nhật được bổ sung theoe tính năng, như có khả năng lập tức gọi ngược lại khi có cuộc gọi nhỡ mà không cần phải mở ứng dụng gọi điện ra. Thông báo sẽ luôn nằm đó cho đến khi người dùng đã đọc hoặc xóa nó đi.

1.3.2Ứng dụng

Android có lượng ứng dụng của bên thứ ba ngày càng nhiều, được chọn lọc và đặt trên một cửa hàng ứng dụng như Google Play hay Amazon Appstore để người dùng lấy về, hoặc bằng cách tải xuống rồi cài đặt tập tin APK từ trang web khác. Các ứng dụng trên Cửa hàng Play cho phép người dùng duyệt, tải về và cập nhật các ứng dụng do Google và các nhà phát triển thứ ba phát hành. Cửa hàng Play được cài đặt sẵn trên các thiết bị thỏa mãn điều kiện tương thích của Google. Ứng dụng sẽ tự động lọc ra một danh sách các ứng dụng tương thích

với thiết bị của người dùng, và nhà phát triển có thể giới hạn ứng dụng của họ chỉ dành cho những nhà mạng cố định hoặc những quốc gia cố định vì lý do kinh doanh. Nếu người dùng mua một ứng dụng mà họ cảm thấy không thích, họ được hoàn trả tiền sau 15 phút kể từ lúc tải về, và một vài nhà mạng còn có khả năng mua giúp các ứng dụng trên Google Play, sau đó tính tiền vào trong hóa đơn sử dụng hàng tháng của người dùng. Đến tháng 9 năm 2012, có hơn 675.000 ứng dụng dành cho Android, và số lượng ứng dụng tải về từ Cửa hàng Play ước tính đạt 25 tỷ.

Các ứng dụng cho Android được phát triển bằng ngôn ngữ Java sử dụng Bộ phát triển phần mềm Android (SDK). SDK bao gồm một bộ đầy đủ các công cụ dùng để phát triển, gồm có công cụ gỡ lỗi, thư viện phần mềm, bộ giả lập điện thoại dựa trên QEMU, tài liệu hướng dẫn, mã nguồn mẫu, và hướng dẫn từng bước. Môi trường phát triển tích hợp (IDE) được hỗ trợ chính thức là Eclipse sử dụng phần bổ sung Android Development Tools (ADT). Các công cụ phát triển khác cũng có sẵn, gồm có Bộ phát triển gốc dành cho các ứng dụng hoặc phần mở rộng viết bằng C hoặc C++, Google App Inventor, một môi trường đồ họa cho những nhà lập trình mới bắt đầu, và nhiều nền tảng ứng dụng web di động đa nền tảng phong phú.

Để vượt qua những hạn chế khi tiếp cận các dịch vụ của Google do sự Kiểm duyệt Internet tại Cộng hòa Nhân dân Trung Hoa, các thiết bị Android bán tại Trung Quốc lục địa thường được điều chỉnh chỉ được sử dụng dịch vụ đã được duyệt.

1.3.3Quản lý bộ nhớ

Vì các thiết bị Android chủ yếu chạy bằng pin, nên Android được thiết kế để quản lý bộ nhớ (RAM) để giảm tối đa tiêu thụ điện năng, trái với hệ điều hành máy tính để bàn luôn cho rằng máy tính sẽ có nguồn điện không giới hạn. Khi một ứng dụng Android không còn được sử dụng, hệ thống sẽ tự động ngưng nó trong bộ nhớ - trong khi ứng dụng về mặt kỹ thuật vẫn "mở", những ứng dụng này sẽ không tiêu thụ bất cứ tài nguyên nào (như năng lượng pin hay năng lượng xử lý) và nằm đó cho đến khi nó được cần đến. Cách làm như vậy có lợi kép là vừa làm tăng khả năng phản hồi nói chung của thiết bị Android, vì ứng dụng không nhất phải đóng rồi mở lại từ đầu, vừa đảm bảo các ứng dụng nền không làm tiêu hao năng lượng một cách không cần thiết.

Android quản lý các ứng dụng trong bộ nhớ một cách tự động: khi bộ nhớ thấp, hệ thống sẽ bắt đầu diệt ứng dụng và tiến trình không hoạt động được một thời gian, sắp theo thời điểm cuối mà chúng được sử dụng (tức là cũ nhất sẽ bị tắt trước). Tiến trình này được thiết kế ẩn đi với người dùng, để người dùng không cần phải quản lý bộ nhớ hoặc tự tay tắt các ứng dụng. Tuy nhiên, sự che giấu này của hệ thống quản lý bộ nhớ Android đã dẫn đến sự thịnh hành của các ứng dụng tắt chương trình của bên thứ ba trên cửa hàng Google Play; những ứng dụng kiểu như vậy được cho là có hại nhiều hơn có lợi.

1.4Phát triển

Android được Google tự phát triển riêng cho đến khi những thay đổi và cập nhật đã hoàn thiện, khi đó mã nguồn mới được công khai. Mã nguồn này, nếu không sửa đổi, chỉ chạy trên một số thiết bị, thường là thiết bị thuộc dòng Nexus. Có nhiều thiết bị có chứa những thành phần được giữ bản quyền do nhà sản xuất đặt vào thiết bị Android của họ.

1.4.1Linux

Android có một hạt nhân dựa trên nhân Linux phiên bản 2.6, kể từ Android 4.0 Ice Cream Sandwich (bánh ngọt kẹp kem) trở về sau, là phiên bản 3.x, với middleware, thư viện và API viết bằng C, còn phần mềm ứng dụng chạy trên một nền tảng ứng dụng gồm các thư viện tương thích với Java dựa trên Apache Harmony. Android sử dụng máy ảo Dalvik với một trình biên dịch động để chạy 'mã dex' (Dalvik Executable) của Dalvik, thường được biên dịch sang Java bytecode. Nền tảng phần cứng chính của Android là kiến trúc ARM. Người ta cũng hỗ trợ x86 thông qua dự án Android x86, và Google TV cũng sử dụng một phiên bản x86 đặc biệt của Android.

Nhân Linux dùng cho Android đã được Google thực hiện nhiều thay đổi về kiến trúc so với nhân Linux gốc. Android không có sẵn X Window System cũng không hỗ trợ các thư viện GNU chuẩn, nên việc chuyển các ứng dụng hoặc thư viện Linux có sẵn sang Android rất khó khăn. Các ứng dụng C đơn giản và SDL cũng được hỗ trợ bằng cách chèn những đoạn shim Java và sử dụng tương tự JNI, như khi người ta chuyển Jagged Alliance 2 sang Android.

Một số tính năng cũng được Google đóng góp ngược vào nhân Linux, đáng chú ý là tính năng quản lý nguồn điện có tên wakelock, nhưng bị những người lập trình chính cho nhân từ chối vì họ cảm thấy Google không có định sẽ tiếp tục bảo trì đoạn mã do họ viết. Google thông báo vào tháng 4 năm 2010 rằng họ sẽ thuê hai nhận viên để làm việc với cộng đồng nhân Linux, nhưng Greg Kroah-Hartman, người bảo trì nhân Linux hiện tại của nhánh ổn định, đã nói vào tháng 12 năm 2010 rằng ông ta lo ngại rằng Google không còn muốn đưa những thay đổi của mình vào Linux dòng chính nữa. Một số lập trình viên Android của Google tỏ ý rằng "nhóm Android thấy chán với quy trình đó," vì nhóm họ không có nhiều người và có nhiều việc khẩn cấp cần làm với Android hơn.

Vào tháng 8 năm 2011, Linus Torvalds rằng "rốt cuộc thì Android và Linux cũng sẽ trở lại với một bộ nhân chung, nhưng điều đó có thể sẽ không xảy ra trong 4 hoặc 5 năm nữa". Vào tháng 12 năm 2011, Greg Kroah-Hartman thông báo kích hoạt Dự án Dòng chính Android, nhắm tới việc đưa một số driver, bản vá và tính năng của Android ngược vào nhân Linux, bắt đầu từ Linux 3.3. Linux cũng đưa tính năng autosleep (tự nghỉ hoạt động) và wakelocks vào nhân 3.5, sau nhiều nỗ lực phối trộn trước đó. Tương tác thì vẫn vậy nhưng bản hiện thực trên Linux dòng chính cho phép hai chế độ nghỉ: bộ nhớ (dạng nghỉ truyền thống mà Android sử dụng), và đĩa (là ngủ đông trên máy tính để bàn). Việc trộn sẽ hoàn tất kể từ nhân 3.8, Google đã công khai kho mã nguồn trong đó có những đoạn thử nghiệm đưa Android về lại nhân 3.8.

Bộ lưu trữ flash trên các thiết bị Android được chia thành nhiều phân vùng, như "/system" dành cho hệ điều hành và "/data" dành cho dữ liệu người dùng và cài đặt ứng dụng. Khác với các bản phân phối Linux cho máy tính để bàn, người sở hữu thiết bị Android không được trao quyền truy cập root vào hệ điều hành và các phân vùng nhạy cảm như /system được thiết lập chỉ đọc. Tuy nhiên, quyền truy cập root có thể chiếm được bằng cách tận dụng những lỗ hổng bảo mật trong Android, điều mà cộng đồng mã nguồn mở thường xuyên sử dụng để nâng cao tính năng thiết bị của họ, kể cả bị những người ác ý sử dụng để cài virus và phần mềm ác ý.

Việc Android có được xem là một bản phân phối Linux hay không vẫn còn là vấn đề gây tranh cãi, tuy được Linux Foundation và Chris DiBona, trưởng nhóm mã nguồn mở Google, ủng hộ. Một số khác, như linux-magazine.com thì không đồng ý, do Android không không hỗ trợ nhiều công cụ GNU, trong đó có glibc.

1.4.2Lịch cập nhật

Google đưa ra các bản cập nhật lớn cho Android theo chu kỳ từ 6 đến 9 tháng, mà phần lớn thiết bị đều có thể nhận được qua sóng không dây. Bản cập nhật lớn mới nhất là Android 6.0 Marshmallow.

So với các hệ điều hành cạnh tranh khác, như iOS, các bản cập nhật Android thường mất thời gian lâu hơn để đến với các thiết bị. Với những thiết bị không thuộc dòng Nexus, các bản cập nhật thường đến sau vài tháng kể từ khi phiên bản được chính thức phát hành. Nguyên nhân của việc này một phần là do sự phong phú về phần cứng của các thiết bị Android, nên người ta phải mất thời gian điều chỉnh bản cập nhật cho phù hợp, vì mã nguồn chính thức của Google chỉ chạy được trên những thiết bị Nexus chủ lực của họ. Chuyển Android sang những phần cứng cụ thể là một quy trình tốn thời gian và công sức của các nhà sản xuất thiết bị, những người luôn ưu tiên các thiết bị mới nhất và thường bỏ rơi các thiết bị cũ hơn. Do đó, những chiếc điện thoại thông minh thế hệ cũ thường không được cập nhật nếu nhà sản xuất quyết định rằng nó không đáng để bỏ thời gian, bất kể chiếc điện thoại đó có khả năng chạy bản cập nhật hay không. Vấn đề này còn trầm trọng hơn khi những nhà sản xuất điều chỉnh Android để đưa giao diện và ứng dụng của họ vào, những thứ này cũng sẽ phải làm lại cho mỗi bản cập nhật. Sự chậm trễ còn được đóng góp bởi nhà mạng, sau khi nhận được bản cập nhật từ nhà sản xuất, họ còn điều chỉnh thêm cho phù hợp với nhu cầu rồi thử nghiệm kỹ lưỡng trên hệ thống mạng của họ trước khi chuyển nó đến người dùng.

Việc thiếu các hỗ trợ hậu mãi của nhà sản xuất và nhà mạng đã bị những nhóm người dùng và các trang tin công nghệ chỉ trích rất nhiều. Một số người viết còn nói rằng giới công nghiệp do cái lợi về tài chính đã cố tình không cập nhật thiết bị, vì nếu thiết bị hiện tại không cập nhật sẽ thúc đẩy việc mua thiết bị mới, một thái độ được coi là "xúc phạm". The Guardian đã than phiền rằng phương cách phân phối bản cập nhật trở nên phức tạp chính vì những nhà sản xuất và nhà mạng đã cố tình làm nó như thế. Vào năm 2011, Google đã hợp tác cùng một số hãng công nghiệp và ra mắt "Liên minh Cập nhật Android", với lời hứa sẽ cập nhật thường xuyên cho các thiết bị trong vòng 18 tháng sau khi ra mắt. Tính đến năm 2012, người ta không còn nghe nhắc đến liên minh này nữa.

1.4.3Cộng đồng mã nguồn mở

Android có một cộng đồng các lập trình viên và những người đam mê rất năng động. Họ sử dụng mã nguồn Android để phát triển và phân phối những phiên bản chỉnh sửa của hệ điều hành. Các bản Android do cộng đồng phát triển thường đem những tính năng và cập nhật mới vào nhanh hơn các kênh chính thức của nhà sản xuất/nhà mạng, tuy không được kiểm thử kỹ lưỡng cũng như không có đảm bảo chất lượng; cung cấp sự hỗ trợ liên tục cho các thiết bị cũ không còn nhận được bản cập nhật chính thức; hoặc mang Android vào những thiết bị ban đầu chạy một hệ điều hành khác, như HP Touchpad. Các bản Android của cộng đồng thường được root sẵn và có những điều chỉnh không phù hợp với những người dùng không rành rẽ, như khả năng ép xung hoặc tăng/giảm áp bộ xử lý của thiết bị. CyanogenMod là firmware của cộng đồng được sử dụng phổ biến nhất, và hoạt động như một tổ chức của số đông khác.

Trước đây, nhà sản xuất thiết bị và nhà mạng tỏ ra thiếu thiện chí với việc phát triển firmware của bên thứ ba. Những nhà sản xuất còn thể hiện lo ngại rằng các thiết bị chạy phần mềm không chính thức sẽ hoạt động không tốt và dẫn đến tốn tiền hỗ trợ. Hơn nữa, các firmware đã thay đổi như CyanogenMod đôi khi còn cung cấp những tính năng, như truyền tải mạng (tethering), mà người dùng bình thường phải trả tiền nhà mạng mới được sử dụng. Kết quả là nhiều thiết bị bắt đầu đặt ra hàng rào kỹ thuật như khóa bootloader hay hạn chế quyền truy cập root. Tuy nhiên, khi phần mềm do cộng đồng phát triển ngày càng trở nên phổ biến, và sau một thông cáo của Thư viện Quốc hội Hoa Kỳ cho phép "jailbreak" (vượt ngục) thiết bị di động, các nhà sản xuất và nhà mạng đã tỏ ra mềm mỏng hơn với các nhà phát triển thứ ba, thậm chí một số hãng như HTC, Motorola, Samsung và Sony, còn hỗ trợ và khuyến khích phát triển. Kết quả của việc này là dần dần nhu cầu tìm ra các hạn chế phần cứng để cài đặt được firmware không chính thức đã bớt đi do ngày càng nhiều thiết bị được phát hành với bootloader đã mở khóa sẵn hoặc có thể mở khóa, tương tự như điện thoại dòng Nexus, tuy rằng thông thường họ sẽ yêu cầu người dùng từ bỏ chế độ bảo hành nếu họ làm như vậy. Tuy nhiên, tuy được sự chấp thuận của nhà sản xuất, một số nhà mạng tại Mỹ vẫn bắt buộc điện thoại phải bị khóa.

Việc mở khóa và "hack" điện thoại thông minh và máy tính bảng vẫn còn là tác nhân gây căng thẳng giữa cộng đồng và công nghiệp. Cộng đồng luôn biện hộ rằng sự hỗ trợ không chính thức ngày càng trở nên quan trọng trước việc nền công nghiệp không cung cấp các bản cập nhật thường xuyên và/hoặc ngưng hỗ trợ cho chính các thiết bị của họ.

1.5Bảo mật và tính riêng tư

Các ứng dụng Android chạy trong một "hộp cát", là một khu vực riêng rẽ với hệ thống và không được tiếp cận đến phần còn lại của tài nguyên hệ thống, trừ khi nó được người dùng trao quyền truy cập một cách công khai khi cài đặt. Trước khi cài đặt ứng dụng, Cửa hàng Play sẽ hiển thị tất cả các quyền mà ứng dụng đòi hỏi: ví dụ như một trò chơi cần phải kích hoạt bộ rung hoặc lưu dữ liệu vào thẻ nhớ SD, nhưng nó không nên cần quyền đọc tin nhắn SMS hoặc tiếp cận danh bạ điện thoại. Sau khi xem xét các quyền này, người dùng có thể chọn đồng ý hoặc từ chối chúng, ứng dụng chỉ được cài đặt khi người dùng đồng ý.

Hệ thống hộp cát và hỏi quyền làm giảm bớt ảnh hưởng của lỗi bảo mật hoặc lỗi chương trình có trong ứng dụng, nhưng sự bối rối của lập trình viên và tài liệu hướng dẫn còn hạn chế đã dẫn tới những ứng dụng hay đòi hỏi những quyền không cần thiết, do đó làm giảm đi hiệu quả của hệ thống này. Một số công ty bảo mật, như Lookout Mobile Security, AVG Technologies, và McAfee, đã phát hành những phần mềm diệt virus cho các thiết bị Android. Phần mềm này không có hiệu quả vì cơ chế hộp cát vẫn áp dụng vào các ứng dụng này, do vậy làm hạn chế khả năng quét sâu vào hệ thống để tìm nguy cơ.

Một nghiên cứu của công ty bảo mật Trend Micro đã liệt kê tình trạng lạm dụng dịch vụ trả tiền là hình thức phần mềm ác ý phổ biến nhất trên Android, trong đó tin nhắn SMS sẽ bị gửi đi từ điện thoại bị nhiễm đến một số điện thoại trả tiền mà người dùng không hề hay biết. Loại phần mềm ác ý khác hiển thị những quảng cáo không mong muốn và gây khó chịu trên thiết bị, hoặc gửi thông tin cá nhân đến bên thứ ba khi chưa được phép. Đe dọa bảo mật trên Android được cho là tăng rất nhanh theo cấp số mũ; tuy nhiên, các kỹ sư Google phản bác rằng hiểm họa từ phần mềm ác ý và virus đã bị thổi phồng bởi các công ty bảo mật nhằm mục đích thương mại, và buộc tội ngành công nghiệp bảo mật đang lợi dụng sự sợ hãi để bán phần mềm diệt virus cho người dùng. Google vẫn giữ quan điểm rằng phần mềm ác ý thật sự nguy hiểm là cực kỳ hiếm, và một cuộc điều tra do F-Secure thực hiện cho thấy chỉ có 0,5% số phần mềm ác ý Android là len vào được cửa hàng Google Play.

Google hiện đang sử dụng bộ quét phần mềm ác ý Google Bouncer để theo dõi và quét các ứng dụng trên Cửa hàng Google Play. Nó sẽ đánh dấu các phần mềm bị nghi ngờ và cảnh báo người dùng về những vấn đề có thể xảy ra trước khi họ tải nó về máy. Android phiên bản 4.2 Jelly Bean được phát hành vào năm 2012

cùng với các tính năng bảo mật được cải thiện, bao gồm một bộ quét phần mềm ác ý được cài sẵn trong hệ thống, hoạt động cùng với Google Play nhưng cũng có thể quét các ứng dụng được cài đặt từ nguồn thứ ba, và một hệ thống cảnh báo sẽ thông báo cho người dùng khi một ứng dụng cố gắng gửi một tin nhắn vào số tính tiền, chặn tin nhắn đó lại trừ khi người dùng công khai cho phép nó.

Điện thoại thông minh Android có khả năng báo cáo vị trí của điểm truy cập Wi-Fi, phát hiện ra việc di chuyển của người dùng điện thoại, để xây dựng những cơ sở dữ liệu có chứa vị trí của hàng trăm triệu điểm truy cập. Những cơ sở dữ liệu này tạo nên một bản đồ điện tử để tìm vị trí điện thoại thông minh, cho phép chúng chạy các ứng dụng như Foursquare, Google Latitude, Facebook Places, và gửi những đoạn quảng cáo dựa trên vị trí. Phần mềm theo dõi của bên thứ ba như TaintDroid, một dự án nghiên cứu trong trường đại học, đôi khi có thể biết được khi nào thông tin cá nhân bị gửi đi từ ứng dụng đến các máy chủ đặt ở xa.

Bản chất mã nguồn mở của Android cho phép những nhà thầu bảo mật lấy những thiết bị sẵn có rồi điều chỉnh để sử dụng ở mức độ bảo mật cao hơn. Ví dụ như Samsung đã cộng tác với General Dynamics sau khi họ thâu tóm Open Kernel Labs để xây dựng lại Jellybean trên nền bộ vi kiểm soát dành cho dự án "Knox".

1.6Giấy phép phát hành

Mã nguồn của Android được cấp phép theo các giấy phép phần mềm mã nguồn mở tự do. Google đưa phần lớn mã nguồn (bao gồm cả các lớp mạng và điện thoại) theo Giấy phép Apache phiên bản 2.0, và phần còn lại, các thay đổi đối với nhân Linux, theo Giấy phép Công cộng GNU phiên bản 2. Liên minh Thiết bị cầm tay mở đã thực hiện các thay đổi trên nhân Linux, với mã nguồn lúc nào cũng công khai. Phần còn lại của Android được Google phát triển một mình, và mã nguồn chỉ được công bố khi phát hành một phiên bản mới. Thông thường Google cộng tác với một nhà sản xuất phần cứng để cung cấp một thiết bị 'chủ lực' (thuộc dòng Google Nexus) với phiên bản mới nhất của Android, sau đó phát hành mã nguồn sau khi thiết bị này được bán ra.

Vào đầu năm 2011, Google quyết định tạm ngưng phát hành mã nguồn Android phiên bản 3.0 Honeycomb dành riêng cho máy tính bảng. Lý do, theo Andy Rubin trong một bài blog Android chính thức, là vì Honeycomb đã được làm

gấp gáp để phục vụ cho Motorola Xoom, và họ không muốn các bên thứ ba tạo ra một "trải nghiệm người dùng cực kỳ tồi tệ" bằng cách cố gắng đưa vào điện thoại thông minh một phiên bản dành riêng cho máy tính bảng. Mã nguồn một lần nữa được xuất bản công khai vào tháng 11 năm 2011 với sự ra mắt của Android 4.0.

Mặc dù phần mềm là mã nguồn mở, các nhà sản xuất thiết bị không thể sử dụng thương hiệu Android của Google trừ khi Google chứng nhận rằng thiết bị của họ phù hợp với Tài liệu Định nghĩa Tương thích (Compatibility Definition Document - CDD). Các thiết bị cũng phải thỏa mãn định nghĩa này thì mới được cấp phép để cài các ứng dụng mã nguồn đóng của Google, gồm cả Google Play. Vì Android không hoàn toàn được phát hành theo giấy phép tương thích GPL, ví dụ như mã nguồn của Google là theo giấy phép Apache license, và cũng vì Google Play cho phép các phần mềm có bản quyền, Richard Stallman và Quỹ phần mềm tự do luôn chỉ trích Android và khuyên người dùng sử dụng hệ điều hành khác như Replicant.

1.7Đón nhận

Android được đón nhận bằng một thái độ thờ ơ khi nó ra mắt vào năm 2007. Mặc dù những nhà phân tích rất ấn tượng với việc các công ty công nghệ có tiếng tăm hợp tác cùng Google để tạo ra Liên minh thiết bị di động mở, người ta vẫn không rõ liệu các nhà sản xuất có sẵn sàng thay thế hệ điều hành mà họ đang dùng bằng Android hay không. Ý tưởng về một nền tảng phát triển mã nguồn mở dựa trên Linux đã thu hút sự quan tâm, nhưng cũng dấy lên những lo ngại rằng Android sẽ phải đối mặt với sự cạnh tranh mạnh mẽ từ những tay chơi có hạng trong thị trường điện thoại thông minh, như Nokia và Microsoft, và các hệ điều hành di động đối thủ cũng sử dụng Linux đang trong quá trình phát triển. Những công ty hàng đầu không giấu sự hoài nghi: Nokia được trích nói rằng "chúng tôi không xem đó là một sự đe dọa," và một thành viên của nhóm Windows Mobile của Microsoft nói rằng "tôi không hiểu rồi họ sẽ có tác động ra sao."

Kể từ đó Android đã phát triển để trở thành hệ điều hành dành cho điện thoại thông minh phổ biến nhất trên thế giới và là "một trong những trải nghiệm di động nhanh nhất hiện nay." Các nhà bình luận thì nhấn mạnh vào bản chất mã nguồn mở của hệ điều hành chính là một trong những yếu tố quyết định sức mạnh, cho phép các công ty như (Kindle Fire), Barnes & Noble (Nook), Ouya,

Baidu, và những hãng khác đổi hướng phần mềm và phát hành những phần cứng chạy trên phiên bản Android đã thay đổi của riêng họ. Kết quả, nó được trang web công nghệ Ars Technica mô tả là "đương nhiên là hệ điều hành mặc định khi phát hành phần cứng mới" cho những công ty không có nền tảng di động riêng của họ. Chính sự mở và uyển chuyển này cũng hiện diện ở cấp độ người dùng cuối: Android cho phép người dùng điện thoại điều chỉnh thoải mái thiết bị của họ và ứng dụng thì có sẵn trên các cửa hàng ứng dụng và trang web không phải của Google. Những đặc điểm này được xem là đóng góp vào những thế mạnh chính của điện thoại Android so với các điện thoại khác.

Android cũng bị phê phán vì thiếu sự hỗ trợ hậu mãi từ nhà sản xuất và nhà mạng, nếu so sánh với iOS của Apple. Với những thiết bị không mang nhãn hiệu Nexus, nhà mạng luôn kiểm tra các tiêu chuẩn của họ rồi thực hiện thay đổi cho riêng từng thiết bị (bắt nguồn từ sự điều chỉnh của nhà sản xuất và sự đa dạng của thiết bị Android) được xem là tác nhân chính trì hoãn việc cập nhật. Những nhà bình luận cũng nói rằng ngành công nghiệp thiết bị di động vì lý do lợi nhuận đã cố tình không cập nhật thiết bị của họ, vì thiếu cập nhật trên thiết bị hiện tại sẽ thúc đẩy việc mua thiết bị mới.

1.8Các bảng biểu (Dashboards)

Trang này cung cấp thông tin số lượng tương đối của các thiết bị có chung một đặc điểm nhất định, chẳng hạn như phiên bản Android hoặc kích thước màn hình. Thông tin này có thể giúp bạn ưu tiên các nỗ lực để hỗ trợ các thiết bị khác nhau bằng cách làm rõ, thiết bị vào đang hoạt động trong Android và hệ sinh thái Google Play.

Dữ liệu này phản ánh các thiết bị chạy ứng dụng Google Play Store mới nhất, tương thích với Android 2.2 và cao hơn. Mỗi ảnh chụp dữ liệu đại diện cho tất cả các thiết bị đã ghé thăm Google Play Store trong vòng 7 ngày trước.

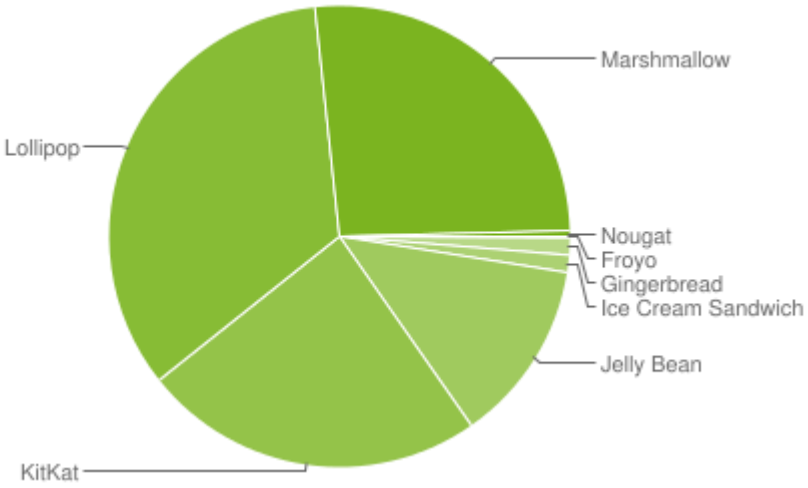
- *Các phiên bản nền tảng*

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%

2.3.3 - 2.3.7	Gingerbread	10	1.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.2%
4.1.x	Jelly Bean	16	4.5%
4.2.x		17	6.4%
4.3		18	1.9%
4.4	KitKat	19	24.0%
5.0	Lollipop	21	10.8%
5.1		22	23.2%
6.0	Marshmallow	23	26.3%
7.0	Nougat	24	0.4%

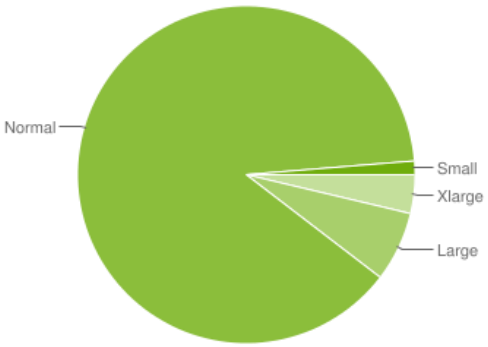
Dữ liệu thu thập được trong một khoảng thời gian 7 ngày, kết thúc vào 05 tháng 12 năm 2016.

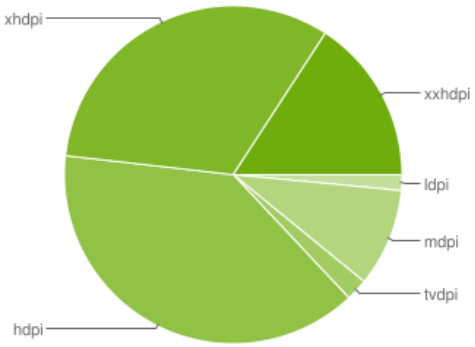
Bất kỳ phiên bản nào có số lượng phân phối ít hơn 0,1% sẽ không được hiển thị.



- *Kích thước màn hình và mật độ điểm ảnh*

	Ldpi	Mdpi	tvdpi	hdpi	xhdpi	Xxhdpi	**Total**
Small	1.3%						1.3%
Normal		2.9%	0.2%	38.0%	31.4%	15.8%	88.3%
Large	0.2%	3.8%	1.9%	0.4%	0.4%		6.7%
Xlarge		2.7%		0.4%	0.6%		3.7%
Total	1.5%	9.4%	2.1%	38.8%	32.4%	15.8%	



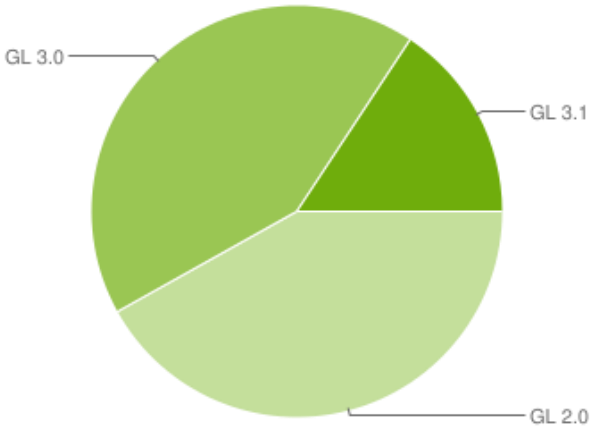


Dữ liệu thu thập được trong một khoảng thời gian 7 ngày, kết thúc vào 05 tháng 12 năm 2016.

Bất kỳ phiên bản nào có số lượng phân phối ít hơn 0,1% sẽ không được hiển thị.

- *Các phiên bản OpenGL*

OpenGL ES Version	Distribution
2.0	41.9%
3.0	42.4%
3.1	15.7%



Dữ liệu thu thập được trong một khoảng thời gian 7 ngày, kết thúc vào 05 tháng 12 năm 2016.

Chương 2 MÔI TRƯỜNG LẬP TRÌNH ANDROID STUDIO

2.1Giới thiệu

Android Studio là môi trường phát triển tích hợp chính thức (IDE) cho phát triển ứng dụng Android, dựa trên IntelliJ IDEA. Dựa trên các công cụ biên tập mã nguồn và phát triển mạnh mẽ của IntelliJ, Android Studio cung cấp nhiều hơn các tính năng nâng cao năng suất của bạn khi xây dựng các ứng dụng Android, chẳng hạn như:

- Một hệ thống dựa trên Gradle xây dựng một cách linh hoạt
- Một giả lập nhanh và nhiều tính năng
- Một môi trường thống nhất, nơi bạn có thể phát triển cho tất cả các thiết bị Android
- Chức năng Instant Run – đẩy các thay đổi trong mã nguồn lên ứng dụng ngay lập tức, để ứng dụng của bạn đang chạy mà không cần xây dựng lại một gói ứng dụng (APK) mới
- Các mẫu mã nguồn và việc tích hợp GitHub sẽ giúp bạn xây dựng các tính năng ứng dụng phổ biến và khả năng tích hợp các mã nguồn mẫu có sẵn
- Nhiều công cụ để kiểm thử và nhiều framework
- Công cụ Lint để bắt hiệu suất, khả năng sử dụng, khả năng tương thích phiên bản, và các vấn đề khác
- Hỗ trợ C ++ và NDK
- Tích hợp hỗ trợ cho Google Cloud Platform, làm cho nó dễ dàng để tích hợp Google Cloud Messaging và App Engine

2.1.1Các phiên bản

- Android Studio v0.1.x (May 2013)
- Android Studio v1.0 (December 2014) – Phiên bản đầu tiên của Android Studio.
- Android Studio v2.0.0 (April 2016)

Instant Run:

- Android Studio triển khai xây dựng nhanh hơn bao giờ hết. Ngoài ra, đẩy những thay đổi trong mã nguồn tời thiết bị giả lập hoặc một thiết bị vật lý hiện nay gần như tức thời. Xem lại các cập nhật của bạn mà không cần thực hiện lại việc xây dựng gỡ lỗi mới, trong nhiều trường hợp, không cần khởi động lại ứng dụng.

- Instant Run hỗ trợ đẩy các thay đổi tới ứng dụng đang chạy:
- Đọc tài liệu để tìm hiểu thêm về Instant Run.

- Android Studio v2.2.0 (September 2016)

New

- New **Layout Editor** với công cụ tùy chỉnh để hỗ trợ ConstraintLayout.
- New **Layout Inspector** cho phép bạn kiểm tra các bức ảnh chụp của hệ thống phân cấp layout, trong khi ứng dụng của bạn đang chạy trên giả lập hoặc thiết bị thật.
- New **Assistant** window để giúp bạn tích hợp các dịch vụ Firebase services vào ứng dụng.
- New **APK Analyzer** tool để bạn có thể kiểm tra các nội dung của ứng dụng đã được đóng gói.
- New **Espresso Test Recorder** tool (hiện tại đang ở phiên bản beta) để giúp bạn tạo ta các bài kiểm tra giao diện người dùng bằng cách ghi lại các tương tác của riêng bạn.
- New **build cache** (hiện đang thử nghiệm) để tăng tốc độ xây dựng (build).
- New **C/C++ build integration with CMake and ndk-build**. Biên dịch và xây dựng mã nguồn native vào các thư viện được đóng gói trong APK, và gỡ lỗi bằng lldb. Để tìm hiểu làm thế nào bao gồm native code trong ứng dụng Android của bạn, đọc Add C and C++ Code to Your Project. Để tìm hiểu làm thế nào gỡ lỗi native code với lldb, xem Debug Native Code.
- New **Samples Browser** bạn có thể dễ dàng tìm kiếm Google Android sample code từ ngay trong Android Studio.
- New **Merged Manifest Viewer** to help you diagnose how your manifest file merges with your app dependencies across project build variants.
- The **Run** window bây giờ chứa bản ghi các thông điệp cho ứng dụng đang chạy hiện thời. Lưu ý rằng bạn có thể cấu hình hiển thị màn hình logcat Monitor, nhưng không phải là cửa sổ **Run**.
- New **Android Emulator** features:

  - Thêm điều khiển mới **Virtual Sensors** và **Cellular** > **Signal Strength**.
  - Thêm tùy chọn **LTE** cho điều khiển **Cellular** > **Network type**.
  - Thêm mô phỏng động tác vuốt theo chiều dọc để di chuyển qua các menu dọc bằng cách cuộn bánh xe ở con chuột máy tính.

- New **Run/Debug Configuration** features:
- Cải thiện cài đặt, cấu hình các tính năng, hiệu suất và giao diện người dùng trong **GPU Debugger** (hiên tại đang ở phiên bản beta).
- Android Studio bây giờ đi kèm với **OpenJDK 8**. Các dự án hiện tại vẫn sử dụng quy định tại **File > Project Structure > SDK Location**. Bạn có

thể chuyển sang sử dụng JDK mới bằng cách nhấn vào **File > Project Structure > SDK Location** và tích chọn **Use embedded JDK**.

- Thêm menu **help và button** mới trong UI, do đó bạn có thể dễ dàng tìm thấy các tài liêu trực tuyến.

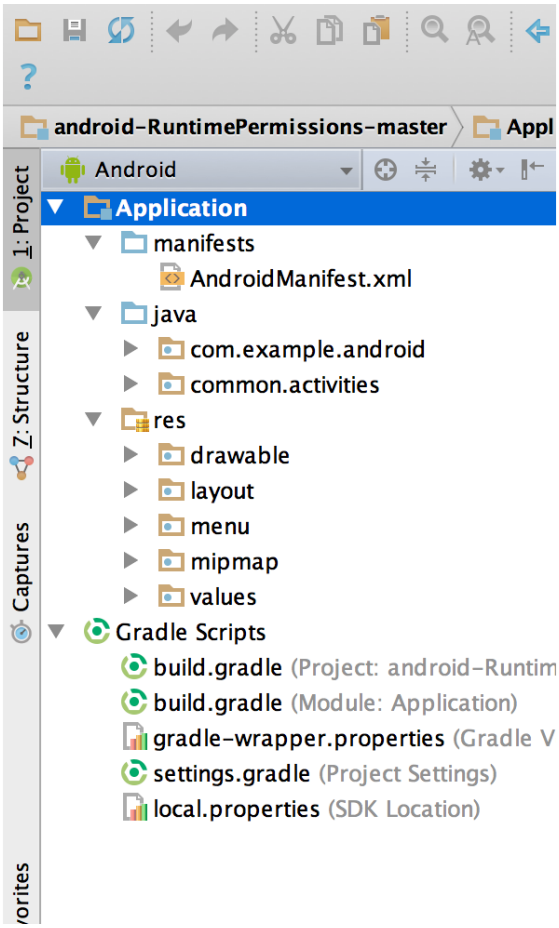
Từ phiên bản 2.2, không còn hỗ trợ công cụ phát triển Android cho Eclipse

Support Ended for Eclipse Android Developer Tools

By Jamal Eason, Product Manager, Android
(Nguồn)

"With the release of Android Studio 2.2, the time has now come to say goodbye to the Eclipse Android Developer Tools. We have formally ended their support and development. There's never been a better time to switch to Android Studio and experience the improvements we've made to the Android development workflow."

https://android-developers.googleblog.com/2016/11/support-ended-for-eclipse-android.html



Hình 1 Xem các tập tin trong project - theo kiểu xem Android

2.1.2Cấu trúc dự án

Mỗi dự án trong Android Studio chứa một hoặc nhiều module với các file mã nguồn và các tập tin tài nguyên. Loại module bao gồm:

- Module ứng dụng Android
- Module thư viện
- Module của Google App Engine

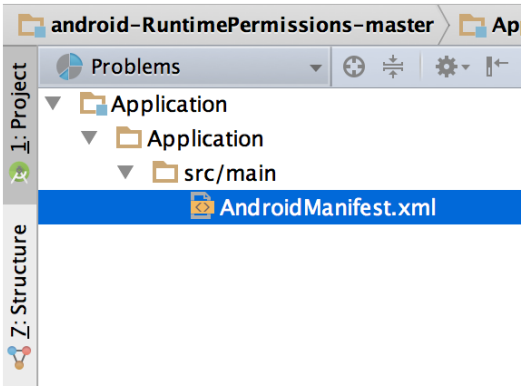
Theo mặc định, Android Studio hiển thị các tập tin trong dự án của bạn dưới kiểu xem Android, như thể hiện trong hình 1. Kiểu xem này được tổ chức thành các module để cung cấp truy cập nhanh đến các file nguồn quan trọng của dự án của bạn.

Tất cả các tập tin được nhìn thấy ở phía trên Gradle Scripts (trong hình 1) và mỗi module ứng dụng có chứa các thư mục sau:

- **manifests**: Chứa các tập tin AndroidManifest.xml.
- **java**: Chứa các tập tin mã nguồn Java, bao gồm cả mã kiểm tra JUnit.
- **res**: Chứa tất cả các nguồn tài nguyên không phải là dạng mã nguồn, chẳng hạn như bố trí XML, UI, và hình ảnh bitmap.

Cấu trúc dự án Android trên đĩa khác với kiểu xem này. Để xem cấu trúc thực tế các tập tin trong dự án, chọn Project từ menu thả xuống (trong hình 1, nó hiển thị là Android).

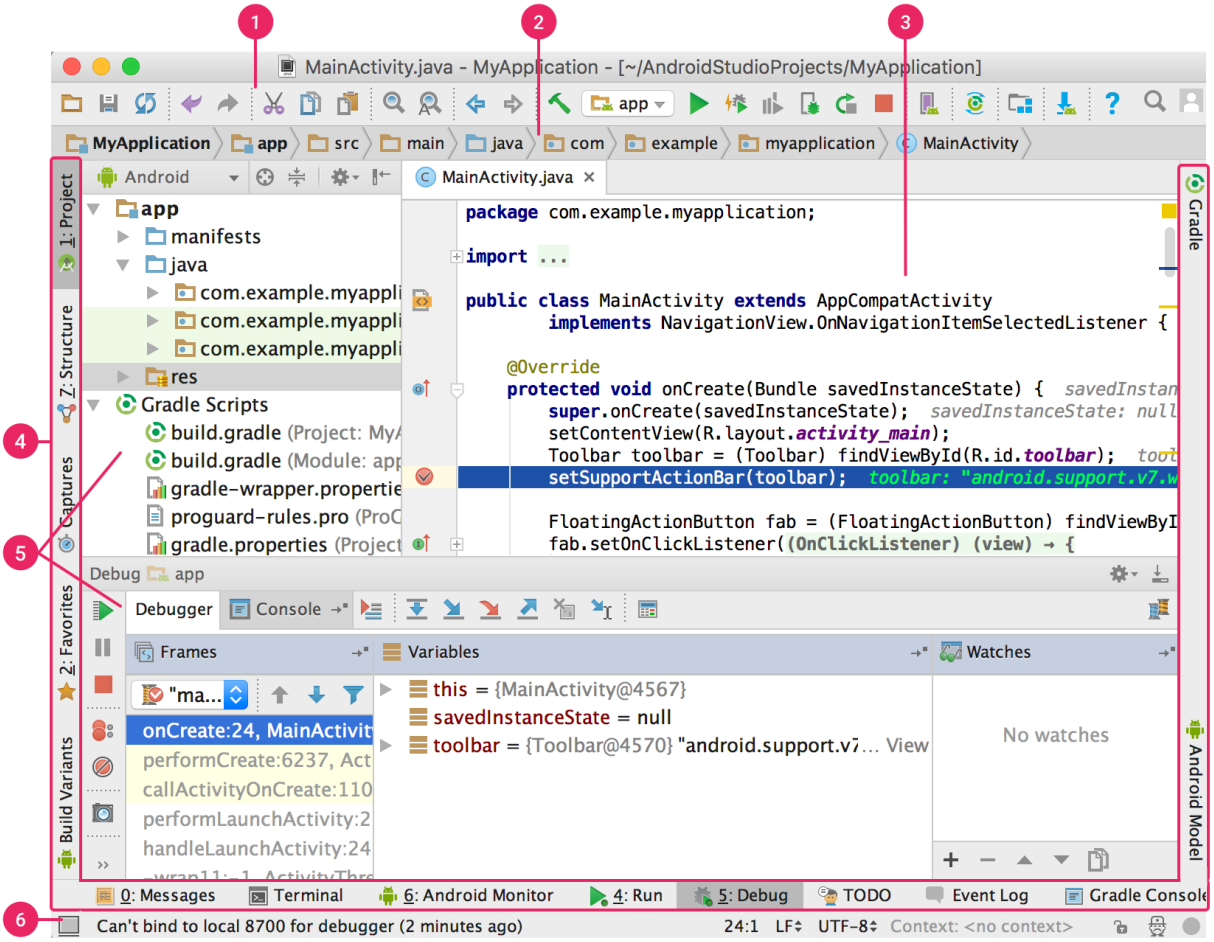
Bạn cũng có thể tùy chỉnh giao diện của các tập tin dự án tập trung vào các khía cạnh cụ thể của phát triển ứng dụng của bạn. Ví dụ, chọn cachs xem Problems của dự án của bạn sẽ hiển thị các liên kết đến các tập tin nguồn chứa bất kỳ mã lệnh và cú pháp lỗi, giả dụ như một phần tử XML thiếu thẻ đóng trong một file layout.



Hình 2 Các tập tin trong dự án dưới kiểu xem Problems

2.1.3Giao diện người dùng

Các cửa sổ chính của Android Studio được tạo thành từ một số khu vực xác định trong hình 3.



Hình 3 Cửa sổ chính của Android Studio

+ 1: Thanh công cụ (**toolbar**) cho phép bạn thực hiện một loạt các hành động, bao gồm cả chạy ứng dụng của bạn và công cụ Android.

+ 2: Các thanh điều hướng (**navigation bar**) giúp bạn điều hướng trong dự án của bạn và các tập tin đang mở để chỉnh sửa. Nó cung cấp một cái nhìn nhỏ gọn hơn của cấu trúc có thể nhìn thấy trong cửa sổ Project.

+ 3: Cửa sổ soạn thảo (**editor window**) là nơi bạn tạo và sửa đổi mã nguồn. Tùy thuộc vào kiểu tập tin, trình biên tập có thể thay đổi. Ví dụ, khi xem một tập tin bố trí giao diện, trình biên tập hiển thị Layout Editor.

+ 4: Các thanh cửa sổ công cụ (**tool window bar**) chạy xung quanh bên ngoài của cửa sổ IDE và chứa các nút cho phép bạn mở rộng hoặc thu gọn cửa sổ công cụ riêng biệt.

+ 5: Các cửa sổ công cụ (**tool windows**) cung cấp cho bạn truy cập vào các nhiệm vụ cụ thể như quản lý dự án, tìm kiếm, kiểm soát phiên bản, và nhiều hơn nữa. Bạn có thể mở rộng hoặc thu gọn chúng.

+ 6: Thanh trạng thái (**status bar**) hiển thị trạng thái dự án của bạn và chính IDE, cũng như bất kỳ cảnh báo hoặc thông điệp nào.

Bạn có thể tổ chức cửa sổ chính để cung cấp cho bản thân nhiều không gian màn hình hơn bằng cách ẩn hoặc di chuyển thanh công cụ và các cửa sổ công cụ. Bạn cũng có thể sử dụng các phím tắt để truy cập hầu hết các tính năng IDE.

Bất cứ lúc nào, bạn cũng có thể tìm kiếm trên mã nguồn, cơ sở dữ liệu, thao tác, các thành phần của giao diện người dùng, và nhiều hơn thế, bằng cách nhấn 2 lần phím Shift, hoặc nhấn vào kính lúp ở góc trên bên phải của cửa sổ Android Studio. Điều này có thể rất hữu ích nếu, ví dụ, bạn đang cố gắng xác định vị trí một thao tác IDE cụ thể mà bạn đã quên làm thế nào để kích hoạt.

- *Cửa sổ công cụ*

Thay vì sử dụng những cách xem định sẵn, Android Studio sử dụng ngữ cảnh của bạn và tự động đưa lên cửa sổ công cụ có liên quan khi bạn làm việc. Theo mặc định, các cửa sổ công cụ thường xuyên được sử dụng nhất được gắn vào thanh cửa sổ công cụ ở các cạnh của chương trình Androdi Studio.

- Để mở rộng hoặc thu gọn cửa sổ công cụ, nhấp vào tên của công cụ trong thanh cửa sổ công cụ. Bạn cũng có thể kéo, ghim, bỏ ghim, ghép và tách cửa sổ công cụ.
- Để trở về bố trí cửa sổ công cụ mặc định, nhấn **Window > Restore Default Layout** hoặc tùy chỉnh cách bố trí mặc định của bạn bằng cách nhấn **Window > Store Current Layout as Default**.
- Để hiển thị hoặc ẩn toàn bộ thanh cửa sổ công cụ, nhấp vào biểu tượng cửa sổ ở góc dưới cùng bên trái của cửa sổ Android Studio.
- Để xác định vị trí một cửa sổ công cụ cụ thể, di chuột qua biểu tượng cửa sổ và chọn cửa sổ công cụ từ menu.

Bạn cũng có thể sử dụng các phím tắt để mở cửa sổ công cụ. Bảng 1 liệt kê các phím tắt cho các cửa sổ thông thường nhất.

Bảng 1. Phím tắt cho một số cửa sổ công cụ hữu ích.

Cửa sổ công cụ	Windows và Linux	Mac
Project	**Alt+1**	**Command+1**

Version Control	**Alt+9**	**Command+9**
Run	**Shift+F10**	**Control+R**
Debug	**Shift+F9**	**Control+D**
Android Monitor	**Alt+6**	**Command+6**
Quay trở lại trình biên tập	**Esc**	**Esc**
Ẩn tất cả cửa sổ công cụ	**Control+Shift+F12**	**Command+Shift+F12**

Nếu bạn muốn ẩn tất cả các thanh công cụ, cửa sổ công cụ, và các tab trình biên tập, nhấp vào **View > Enter Distraction Free Mode**. Thao tác này khởi chạy *Distraction Free Mode*. Để thoát khỏi Distraction Free Mode, bấm **View > Exit Distraction Free Mode**.

Bạn có thể sử dụng *Speed Search* để tìm kiếm và lọc trong hầu hết các cửa sổ công cụ trong Android Studio. Để sử dụng Speed Search, chọn cửa sổ công cụ và sau đó gõ truy vấn tìm kiếm của bạn.

- *Tự động hoàn thành mã lệnh*

Android Studio có ba loại hoàn thành mã, mà bạn có thể truy cập bằng phím tắt.

Bảng 2. Các phím tắt cho tự động hoàn thành mã lệnh

Kiểu	Mô tả	Windows và Linux	Mac
Cơ bản	Hiển thị gợi ý cơ bản cho các biến, các kiểu, phương thức, biểu thức, và nhiều hơn thế. Nếu bạn sử dụng kiểu cơ bản hai lần liên tiếp, bạn sẽ thấy nhiều kết quả hơn, bao gồm cả các thành phần private và các thành phần tĩnh không được tích hợp.	**Control + Space**	**Control + Space**

Thông minh	Hiển thị tùy chọn có liên quan dựa vào ngữ cảnh. Tự động hoàn thành kiểu thông minh là đoán trước của dòng chảy các kiểu và dữ liệu dự kiến. Nếu bạn gọi kiểu thông minh hai lần liên tiếp, bạn sẽ thấy nhiều kết quả hơn, bao gồm các chuỗi.	**Control + Shift + Space**	**Control + Shift + Space**
Câu lệnh	Hoàn thành câu lệnh hiện tại cho bạn, thêm dấu ngoặc đơn bị thiếu, dấu ngoặc kép, dấu ngoặc vuông, định dạng, … cho bạn	**Control + Shift + Enter**	**Shift + Command + Enter**

- *Điều hướng*

Dưới đây là một số mẹo để giúp bạn di chuyển xung quanh Android Studio.

- Chuyển đổi giữa các tập tin mới truy cập bằng cách sử dụng thao tác *Recent Files*. Nhấn **Control + E** (**Command + E** trên máy Mac) xuất hiện các hành động tập tin gần đây. Theo mặc định, các tập tin truy cập lần cuối được chọn. Bạn cũng có thể truy cập vào bất kỳ cửa sổ công cụ thông qua các cột bên trái trong thao tác này.
- Xem cấu trúc của tập tin hiện tại bằng cách sử dụng thao tác *File Structure*. Xuất hiện cấu trúc hành động tập tin bằng cách nhấn **Control + F12** (**Command + F12** trên máy Mac). Sử dụng hành động này, bạn có thể nhanh chóng chuyển tới bất kỳ một phần của tập tin hiện tại của bạn.
- Tìm kiếm và điều hướng đến một class cụ thể trong dự án của bạn bằng cách sử dụng thao tác *Navigate to Clas*. Làm xuất hiện thao tác bằng cách nhấn **Control + N** (**Command + O** trên máy Mac). Điều hướng đến Class hỗ trợ các biểu phức tạp. Nếu bạn gọi nó hai lần liên tiếp, nó cho bạn thấy kết quả bên ngoài các lớp trong dự án.
- Điều hướng đến một tập tin hoặc thư mục bằng cách sử dụng thao tác *Navigate to File*. Làm xuất hiện Navigate to File bằng cách nhấn **Control + Shift + N** (**Command + Shift + O** trên máy Mac). Để tìm kiếm các thư mục thay vì tập tin, thêm / ở cuối biểu thức của bạn.
- Điều hướng đến một phương thức hoặc trường theo tên bằng cách sử dụng *Navigate to Symbol*. Đưa lên thao tác Navigate to Symbol bằng cách nhấn **Control + Shift + Alt + N** (**Command + Shift + Alt + O** trên máy Mac).
- Tìm tất cả thành phần của mã tham chiếu các lớp, phương thức, trường, tham số, hoặc câu lệnh tại vị trí con trỏ hiện tại bằng cách nhấn **Alt + F7**.

- *Định dạng*

Khi bạn chỉnh sửa, Android Studio sẽ tự động được áp dụng định dạng như quy định trong cài đặt định dạng mã lệnh của bạn. Bạn có thể tùy chỉnh các thiết lập định dạng mã của ngôn ngữ lập trình, trong đó có các quy ước cho các tab và thụt lề, khoảng trắng và dòng trống. Để tùy chỉnh các thiết lập định dạng mã của bạn, hãy nhấp vào **File** > **Settings** > **Editor** > **Code Style** (**Android Studio** > **Preferences** > **Editor** > **Code Style** trên máy Mac.)

Mặc dù IDE tự động áp dụng hiệu ứng định dạng khi bạn làm việc, bạn cũng có thể gọi một cách rõ ràng hành động định dạng lại mã (*Reformat Code*) bằng cách nhấn **Control** + **Alt** + **L** (**Opt** + **Command** + **L** trên máy Mac), hoặc tự động chỉnh sửa tất cả các dòng bằng cách nhấn **Control** + **Alt** + **I (Alt** + **Option** + **I** trên máy Mac).

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
```

Hình 4 Mã lệnh trước khi định dạng

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabl(true);

    // Get reference to the drawer layout and set event listener
```

Formatted 7 lines
Show reformat dialog: ⌥⇧⌘L

Hình 5 Mã lệnh sau khi định dạng

- *Quản lý mã nguồn*

Android Studio hỗ trợ một loạt các hệ thống quản lý mã nguồn (version control systems - VCS), bao gồm Git, GitHub, CVS, Mercurial, Subversion, và Google Cloud Source Repositories.

Sau khi đưa ứng dụng của bạn vào Android Studio, sử dụng các tùy chọn trình đơn Android Studio VCS cho phép hỗ trợ VCS cho hệ thống quản lý mã nguồn mong muốn, tạo ra một kho lưu trữ, nhập các tập tin mới vào quản lý mã nguồn, và thực hiện các hoạt động kiểm soát mã nguồn khác:

- Từ menu Android Studio **VCS**, kích **Enable Version Control Integration**.
- Từ trình đơn thả xuống, chọn một hệ thống kiểm soát phiên bản liên kết với thư mục gốc của dự án, và sau đó bấm **OK**.

Menu VCS bây giờ hiển thị một số tùy chọn điều khiển phiên bản dựa trên hệ thống bạn chọn.

Lưu ý: Bạn cũng có thể sử dụng **File > Settings > Version Control** để thiết lập và thay đổi các thiết lập quản lý mã nguồn.

2.1.4Hệ thống xây dựng Gradle

Android Studio sử dụng Gradle là nền tảng của hệ thống xây dựng, với nhiều khả năng Android cụ thể được cung cấp bởi các plugin Android cho Gradle. Hệ thống xây dựng này chạy như một công cụ tích hợp từ trình đơn Android Studio, và độc lập từ dòng lệnh. Bạn có thể sử dụng các tính năng của hệ thống xây dựng để làm những việc sau:

- Tùy chỉnh, cấu hình và mở rộng quá trình xây dựng.
- Tạo nhiều gói cài đặt (APK) cho ứng dụng của bạn, với các tính năng khác nhau bằng cách sử dụng cùng một dự án và module.
- Tái sử dụng mã nguồn và tài nguyên.

Bằng cách sử dụng sự linh hoạt của Gradle, bạn có thể đạt được tất cả những điều này mà không sửa đổi các tập tin nguồn cốt lõi ứng dụng của bạn. Android Studio xây dựng các tập tin được đặt tên *build.gradle*. Chúng là các tập tin văn bản đơn giản có sử dụng cú pháp Groovy để cấu hình xây dựng với các thành phần cung cấp bởi plugin Android cho Gradle. Mỗi dự án có một file xây dựng top-level cho toàn bộ dự án và file xây dựng module-level riêng biệt cho mỗi module. Khi bạn nhập một dự án hiện có, Android Studio sẽ tự động tạo ra các tập tin xây dựng cần thiết.

- *Xây dựng các biến thể*

Hệ thống xây dựng có thể giúp bạn tạo ra các phiên bản khác nhau của cùng một ứng dụng từ một dự án duy nhất. Điều này rất hữu ích khi bạn có cả một phiên bản miễn phí và phiên bản trả tiền cho ứng dụng của bạn, hoặc nếu bạn muốn phân phối nhiều gói ứng dụng cho các cấu hình thiết bị khác nhau trên Google Play.

- *APK Splits*

APK Splits cho phép bạn tạo hiệu quả nhiều gói cài đặt dựa trên mật độ màn hình hoặc ABI. Ví dụ, APK Splits cho phép bạn tạo phiên bản hdpi và mdpi riêng biệt của một ứng dụng trong khi vẫn đang xem xét chúng theo một biến thể duy nhất, và cho phép chúng chia sẻ các thiết lập cho ứng dụng kiểm thử, javac, dx, javac, dx và ProGuard.

- *Resource Shrinking*

Resource Shrinking trong Android Studio sẽ tự động loại bỏ các nguồn tài nguyên không sử dụng từ ứng dụng đóng gói và thư viện phụ thuộc của bạn. Ví dụ, nếu ứng dụng của bạn đang sử dụng dịch vụ Google Play để truy cập chức năng Google Drive, và hiện thời bạn không sử dụng Google Sign-In, Resource Shrinking có thể loại bỏ các tài nguyên dành cho các nút theSignInButton.

Lưu ý: Resource Shrinking hoạt động kếp hợp cùng với các công cụ có chức năng tương tự khác, như ProGuard.

- *Quản lý phụ thuộc*

Các phụ thuộc (Dependencies) cho dự án của bạn được xác định bởi tên trong file *build.gradle*. Gradle chăm sóc việc tìm kiếm dependencies của bạn và làm cho chúng có sẵn trong xây dựng của bạn. Bạn có thể khai báo module phụ thuộc, thư viện phụ thuộc trong tập tin *build.gradle* của bạn. Android Studio cấu hình các dự án sử dụng Maven Central Repository theo mặc định. (Cấu hình này được bao gồm trong tập tin xây dựng top-level cho dự án.)

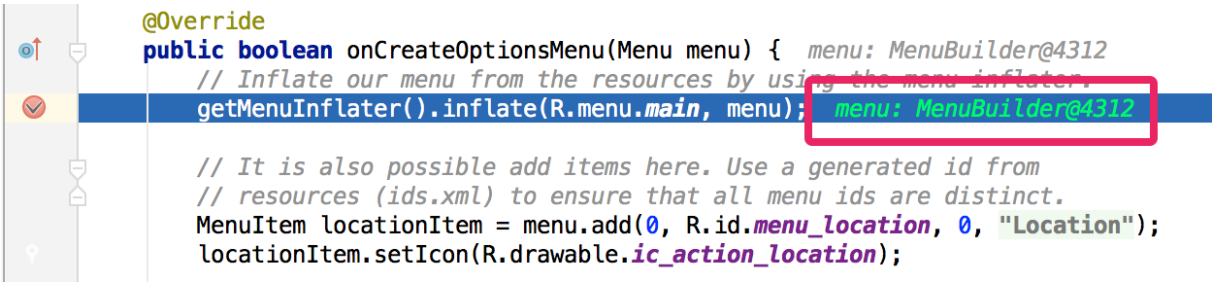
2.1.5Debug

Android Studio giúp bạn trong việc gỡ lỗi và cải thiện hiệu suất của mã nguồn của bạn, bao gồm gỡ lỗi từng dòng và các công cụ phân tích hiệu suất.

- *Inline debugging*

Sử dụng gỡ lỗi từng dòng (inline debugging) để tăng cường hiệu quả duyệt qua các đoạn mã của bạn trong giao diện gỡ lỗi với xác minh từng dòng của tài liệu tham khảo, các biểu thức, và các giá trị biến. Thông tin debug Inline bao gồm:

- Giá trị biến Inline
- Đối tượng liên quan mà tham chiếu một đối tượng đã được chọn
- Các giá trị trả về của phương thức
- Lambda và biểu thức
- Giá trị tooltip



Hình 6 Giá trị biến trong dòng lệnh

Để kích hoạt tính năng inline debugging, trong cửa sổ **Debug**, nhấp vào **Settings** và chọn hộp kiểm cho **Show Values Inline**.

- *Giám sát hiệu suất*

Android Studio cung cấp giám sát hiệu suất để bạn có thể dễ dàng theo dõi bộ nhớ và CPU sử dụng bởi ứng dụng của bạn, tìm các đối tượng deallocated, xác định vị trí rò rỉ bộ nhớ, tối ưu hóa hiệu năng đồ họa, và phân tích các yêu cầu mạng. Với ứng dụng của bạn chạy trên một thiết bị hoặc giả lập, mở cửa sổ công cụ **Android Monitor**, và sau đó nhấp vào tab **Monitors**.

- *Theo dõi việc cấp phát*

Android Studio cho phép bạn theo dõi cấp phát bộ nhớ như việc theo dõi bộ nhớ sử dụng. Theo dõi cấp phát bộ nhớ cho phép bạn giám sát nơi các đối tượng đang được cấp phát khi bạn thực hiện thao tác nào đó. Biết được những phân bổ cho phép bạn tối ưu hóa hiệu suất và sử dụng bộ nhớ ứng dụng của bạn bằng cách điều chỉnh các lời gọi phương thức liên quan đến những thao tác đó.

- *Truy cập file dữ liệu*

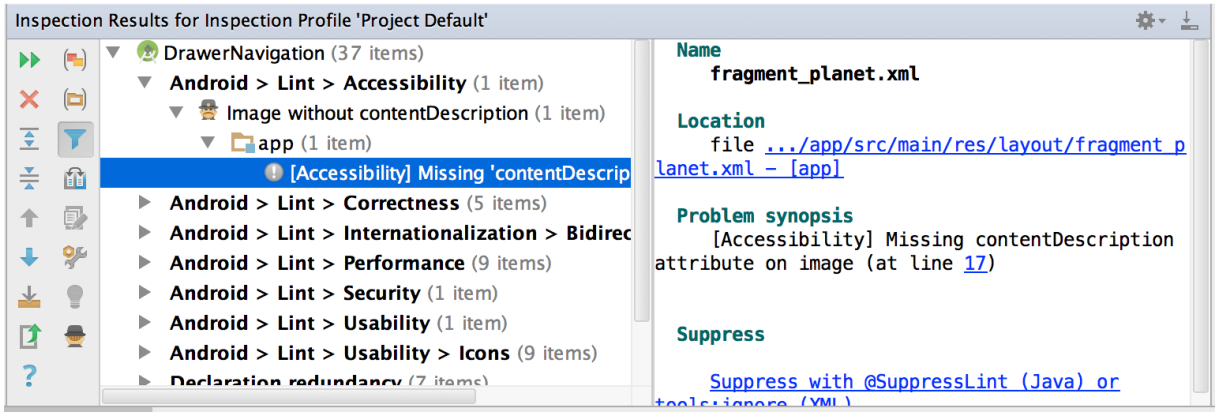
Các công cụ Android SDK, chẳng hạn như Systrace, logcat, và Traceview, tạo ra các dữ liệu về hiệu suất và sửa lỗi để phân tích ứng dụng một cách chi tiết.

Để xem các tập tin dữ liệu đã được tạo sẵn, mở cửa sổ công cụ Captures. Trong danh sách các tập tin được tạo ra, nhấp đúp vào một tập tin để xem các dữ liệu. Kích chuột phải vào bất kỳ file *.hprof* để chuyển đổi chúng sang các định dạng file *.hprof* tiêu chuẩn.

- *Kiểm tra mã nguồn*

Bất cứ khi nào bạn biên dịch chương trình của bạn, Android Studio sẽ tự động chạy cấu hình Lint và kiểm tra IDE khác để giúp bạn dễ dàng xác định vấn đề một cách chính xác với trúc mã nguồn của bạn.

Các công cụ Lint kiểm tra các file mã nguồn dự án Android của bạn cho các lỗi tiềm tàng và cải tiến tối ưu hóa cho đúng đắn, bảo mật, hiệu suất, khả năng sử dụng, khả năng tiếp cận mang tính toàn cầu.



Hình 7 Kết quả một cuộc kiểm tra Lint trong Android Studio

2.2Cài đặt Android Studio

2.2.1Yêu cầu hệ thống máy tính

Windows	Mac	Linux
• Microsoft® Windows® 7/8/10 (32 hoặc 64-bit) • Dung lượng RAM nhỏ nhất là 3 GB, khuyến nghị 8 GB cộng thêm 1 GB cho Android Emulator • Dung lượng ổ cứng nhỏ nhất là 2 GB, khuyến nghị 4 GB (500 MB cho IDE + 1.5 GB cho Android SDK và hệ thống giả lập) • Độ phân giải màn hình nhỏ nhất là 1280 x 800 • Phần tăng tốc giả lập: hệ điều hành 64-bit và vi xử lý Intel® có hỗ trợ Intel® VT-x, Intel® EM64T (Intel® 64), và chức năng Execute Disable (XD)	• Mac® OS X® 10.10 (Yosemite) hoặc cao hơn • Dung lượng RAM nhỏ nhất là 3 GB, khuyến nghị 8 GB cộng thêm 1 GB cho Android Emulator • Dung lượng ổ cứng nhỏ nhất là 2 GB, khuyến nghị 4 GB (500 MB cho IDE + 1.5 GB cho Android SDK và hệ thống giả lập) • Độ phân giải màn hình nhỏ nhất là 1280 x 800	• GNOME hoặc KDE *Đã được kiểm tra trên Ubuntu® 12.04, Precise Pangolin (bản phân phối 64-bit có thể chạy ứng dụng 32-bit)* • Phiên bản 64-bit có khả năng chạy ứng dụng 32-bit • GNU C Library (glibc) 2.19 hoặc cao hơn • Dung lượng RAM nhỏ nhất là 3 GB, khuyến nghị 8 GB cộng thêm 1 GB cho Android Emulator • Dung lượng ổ cứng nhỏ nhất là 2 GB, khuyến nghị 4 GB (500 MB cho IDE + 1.5 GB cho Android SDK và hệ thống giả lập) • Độ phân giải màn hình nhỏ nhất là 1280 x 800 • Phần tăng tốc giả lập: hệ điều hành 64-bit và vi xử lý Intel® có hỗ trợ Intel® VT-x, Intel® EM64T (Intel® 64), và chức năng Execute Disable (XD), hoặc vi xủa lý AMD có hỗ trợ AMD Virtualization™ (AMD-V™)

2.2.2Yêu cầu phần mềm:

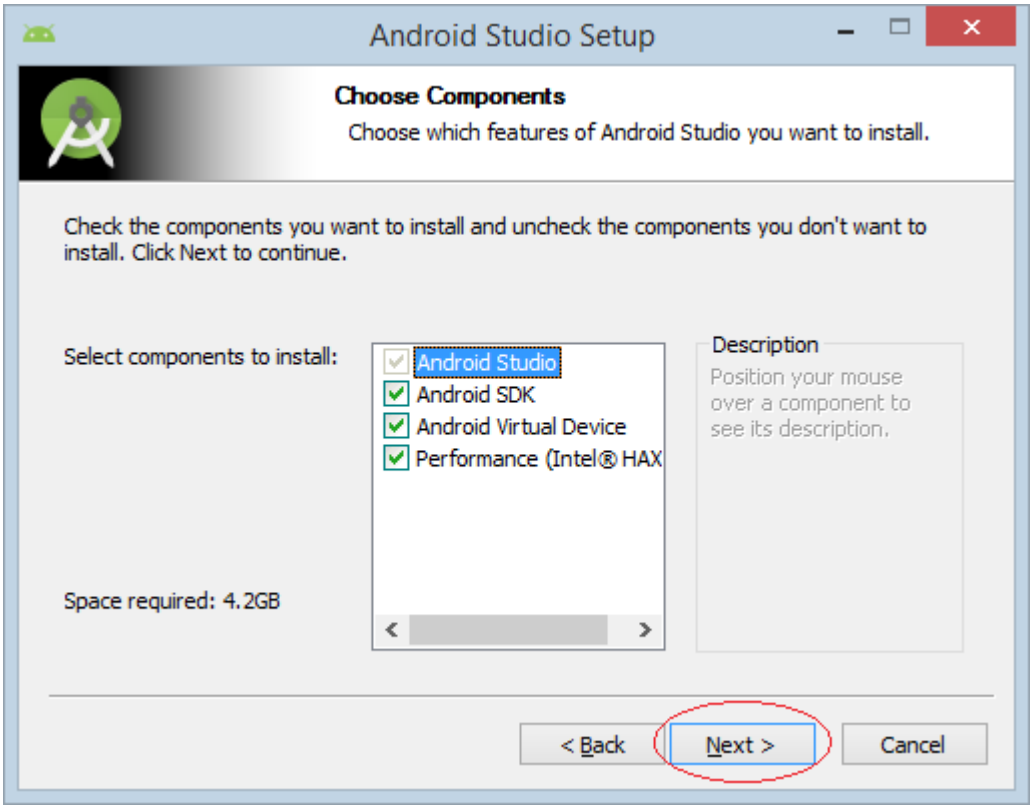
✓ Java SE Development Kit: JDK là một môi trường phát triển để xây dựng các ứng dụng, applet, và các thành phần bằng cách sử dụng ngôn ngữ lập trình Java.
JDK bao gồm các công cụ hữu ích cho phát triển và thử nghiệm các chương trình viết bằng ngôn ngữ Java và chạy trên nền tảng Java.
Download JDK

- ✓ Android Studio: IDE chính thức cho Android. Android Studio cung cấp các công cụ nhanh nhất để xây dựng các ứng dụng trên tất cả các loại thiết bị Android.
  Download Android Studio

2.2.3Các bước cài đặt Android Studio

Sau khi tải về, ta bắt đầu cài đặt Android Studio

Hình 8 Cửa sổ bắt đầu cài đặt chương trình Android Studio



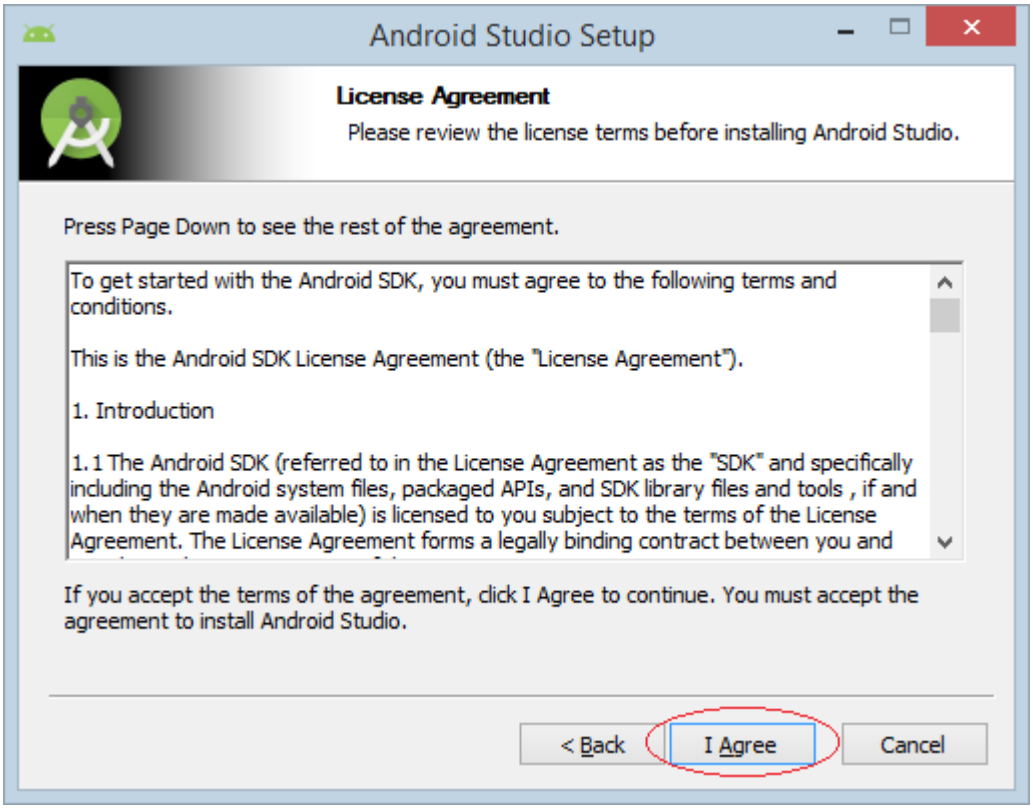
Hình 9 Các tùy chọn cài đặt

Lựa chọn tất cả các options.

Android SDK (software development kit) là một tập hợp các công cụ được sử dụng để phát triển ứng dụng cho Android.

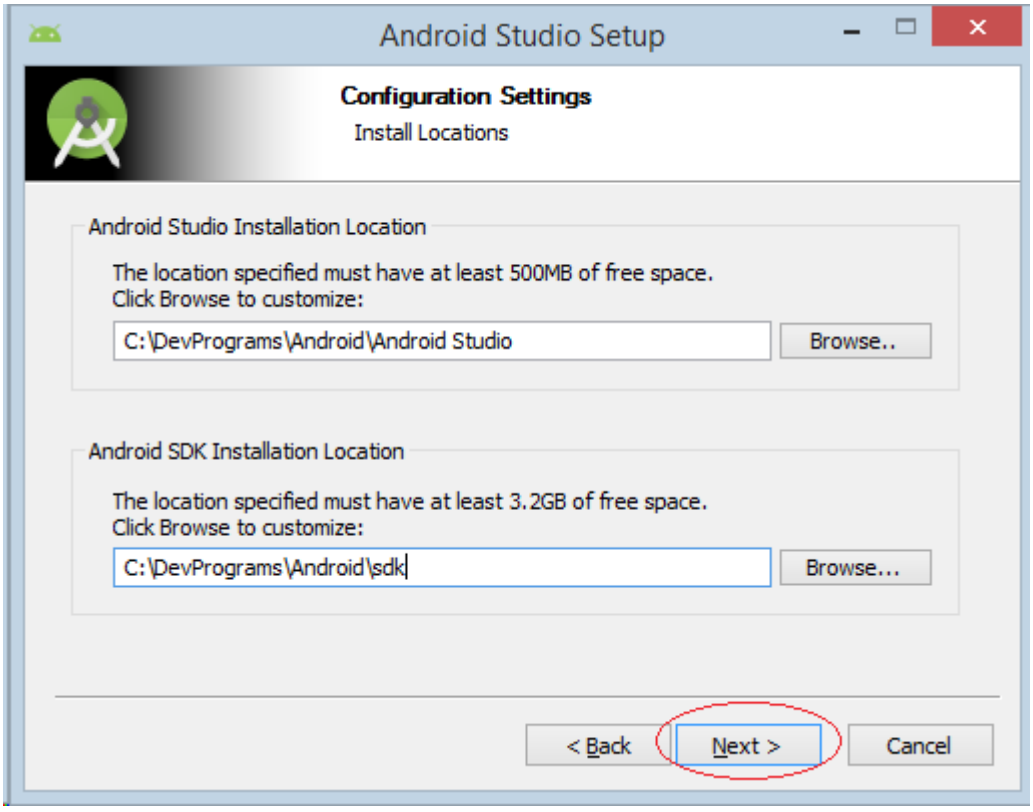
Android Virtual Device (AVD) là một cấu hình thiết bị, nó chạy với bộ giả lập Android (Android emulator). Nó làm việc với bộ giả lập để cung cấp một môi trường thiết bị ảo cụ thể, để cài đặt và chạy ứng dụng Android.

Intel Hardware Accelerated Execution Manager (Intel® HAXM) là một phần cứng hỗ trợ ảo hóa (hypervisor) có sử dụng công nghệ Intel Virtualization Technology (Intel® VT) để tăng tốc độ ứng dụng Android trên máy chạy phần mềm giả lập Android.

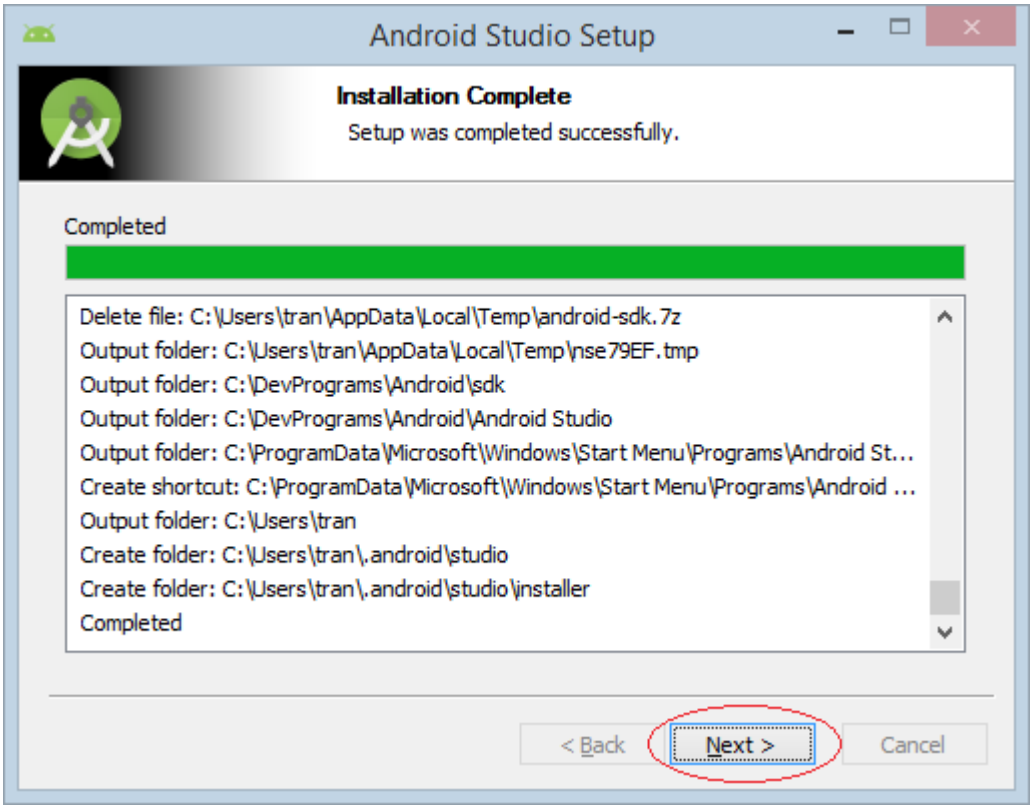


Hình 10 Chấp thuận với giấy phép sử dụng

Đồng ý với với giấy phép sử dụng

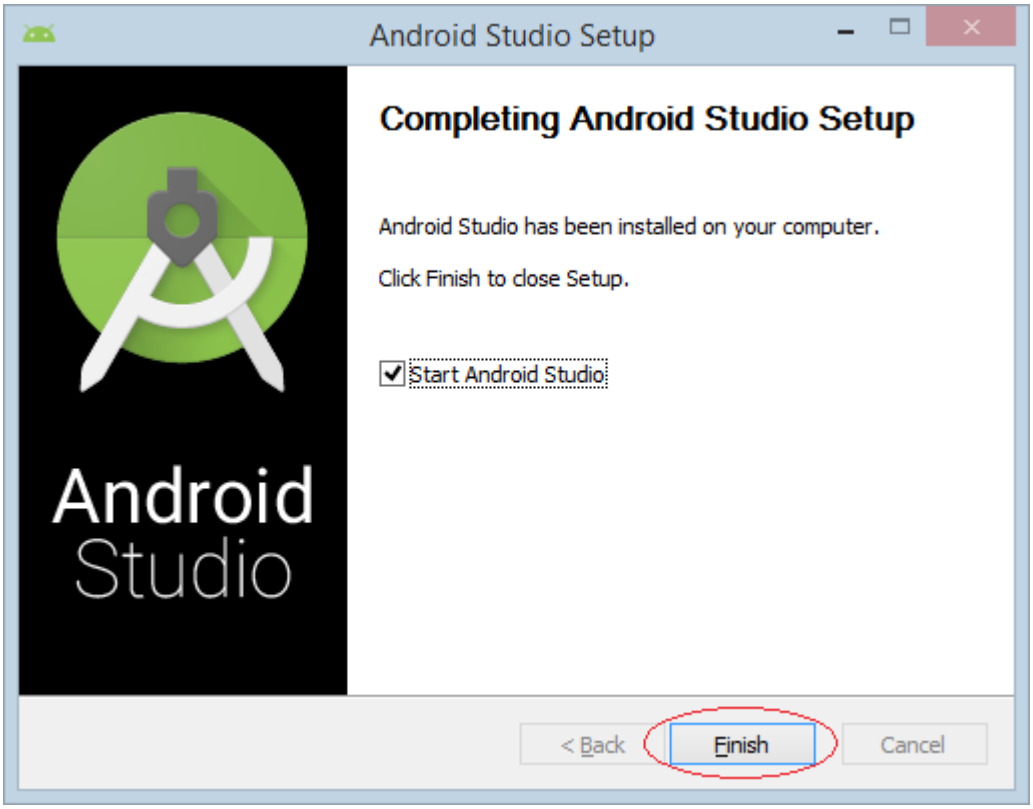


Hình 11 Chọn thư mục cài đặt



Hình 12 Quá trình cài đặt

Thông báo tiến tình cài đặt.



Hình 13 Cửa sổ thông báo cài đặt hoàn tất

Đến đây chương trình đã cài xong và có thể bắt đầu sử dụng.

2.2.4Tạo và quản lý thiết bị ảo (AVD)

Android Virtual Device (AVD) cho phép bạn xác định các đặc tính của một chiếc điện thoại Android, tablet, Android Wear, hoặc thiết bị Android TV mà bạn muốn mô phỏng trong Android Emulator. AVD Manager giúp bạn dễ dàng tạo và quản lý các thiết bị ảo.

- *Giới thiệu AVD*

Một AVD chứa một cấu hình phần cứng, ảnh hệ thống, khu vực lưu trữ, giao diện, và các thuộc tính khác.

- **Cấu hình phần cứng** (Hardware profile)

Các cấu hình phần cứng xác định các đặc tính của một thiết bị như thiết bị thật. Manager AVD có cài đặt sẵn với các cấu hình phần cứng nhất định, chẳng hạn như các thiết bị điện thoại Nexus, ngoài ra bạn có thể định nghĩa và nhập cấu hình phần cứng khi cần thiết. Bạn có thể ghi đè lên một số cài đặt trong AVD của bạn, nếu cần thiết.

Để kiểm tra hiệu quả ứng dụng của bạn, bạn nên tạo một AVD cho từng loại thiết bị mà ứng dụng của bạn được thiết kế để hỗ trợ.

- **Ảnh hệ thống** (System image)

AVD Manager giúp bạn chọn một hình ảnh hệ thống cho AVD bạn bằng cách cung cấp các khuyến nghị. Nó cũng cho phép bạn tải về các hình ảnh hệ thống, một số add-on thư viện, như API của Google, mà ứng dụng của bạn có thể yêu cầu. Hệ thống x86 chạy nhanh nhất trong các giả lập. Android Wear và các thiết bị Android TV có xu hướng chạy tốt nhất trên phiên bản gần đây, trong khi người dùng điện thoại Android và máy tính bảng có xu hướng sử dụng các phiên bản cũ hơn một chút, như thể hiện trong biểu đồ mức API.

Android Studio khuyên bạn nên tạo một AVD cho mỗi cấp API cho ứng dụng của bạn mà có khả năng hỗ trợ dựa trên thiết lập <uses-sdk> trong manifest. Ví dụ, bạn có thể muốn thử nghiệm với tất cả các cấp API bằng hoặc cao hơn so với thiết lập minSdkVersion. Bằng cách kiểm tra với các cấp API cao hơn so với yêu cầu của ứng dụng của bạn, bạn có thể đảm sự tương thích của ứng dụng khi người dùng tải về bản cập nhật hệ thống.

- **Khu vực lưu trữ** (Storage area)

Các AVD có một khu vực lưu trữ chuyên dụng trên máy tính của bạn. Nó lưu trữ các dữ liệu người dùng thiết bị, chẳng hạn như các ứng dụng được cài đặt và thông tin thiết lập (setting), cũng như một thẻ SD mô phỏng. Nếu cần thiết, bạn

có thể sử dụng AVD Manager để xóa sạch dữ liệu người dùng, vì vậy các thiết bị sẽ có dữ liệu giống nhau nếu được tạo mới.

- **Giao diện** (Skin)

Một giao diện giả lập xác định diện mạo của một thiết bị. Manager AVD cung cấp một số giao diện được xác định trước. Bạn cũng có thể định nghĩa riêng của bạn, hoặc sử dụng giao diện được cung cấp bởi các bên thứ ba.

- **AVD và các ứng dụng chức năng** (AVD and app features)

Cũng như với một thiết bị thực, các ứng dụng có thể hoàn toàn sử dụng các tính năng được định nghĩa trong một AVD, chẳng hạn như máy ảnh, nó phải có thiết lập <uses-feature> tương ứng trong manifest. Xem thuộc tính cấu hình thiết bị và thuộc tính AVD cho danh sách các tính năng, bạn có thể xác định trong AVDs của bạn.

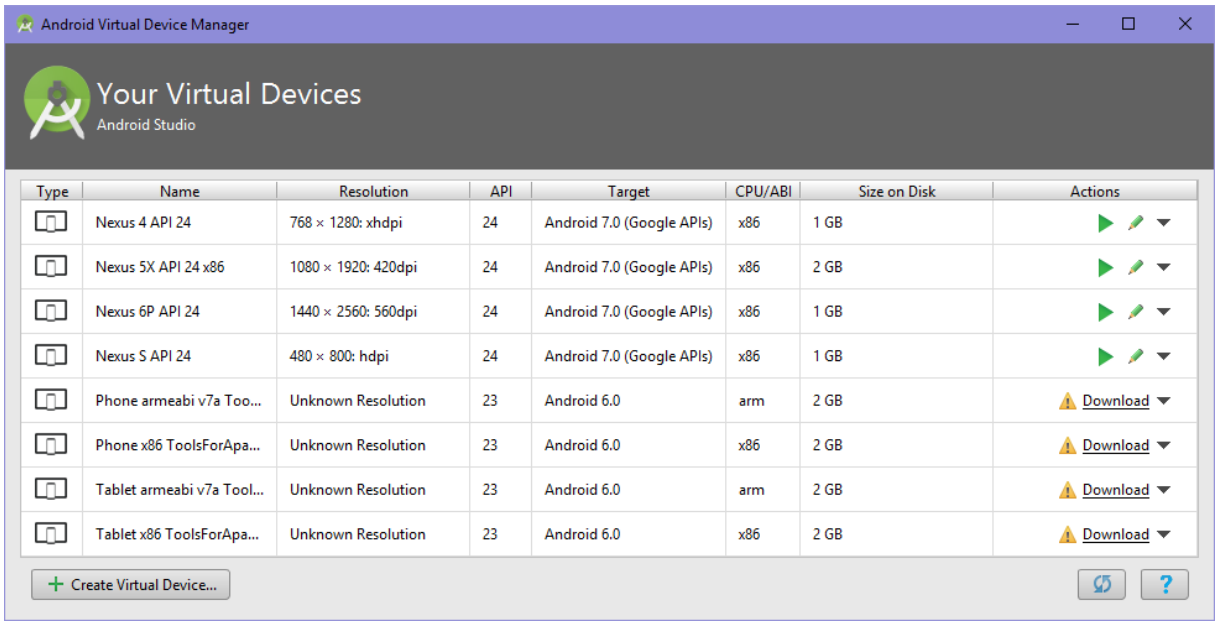
➢ *Xem và quản lý các thiết bị ảo*

AVD Manager cho phép bạn quản lý tất cả AVDs của bạn ở một nơi.

Để chạy AVD Manager, thực hiện một trong các cách sau:

- Trong Android Studio, chọn **Tools > Android > AVD Manager**.
- Nhấn vào biểu tượng **AVD Manager** trong thanh công cụ.

AVD Manager xuất hiện như hình dưới:



Hình 14 Trang Your Virtual Devices - hiện danh sách các thiết bị ảo của bạn

➢ *Thiết lập thiết bị ảo (AVD)*

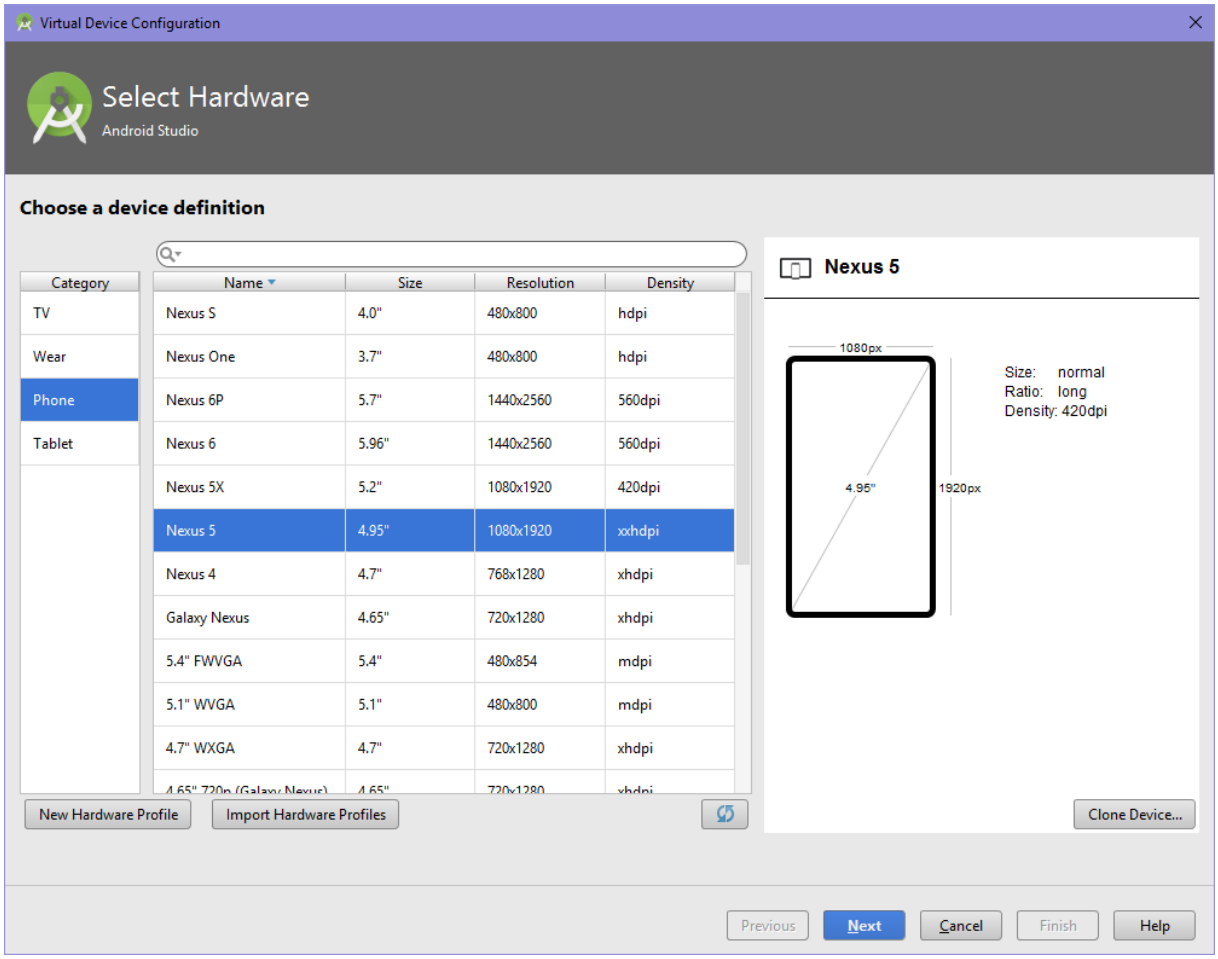
Bạn có thể tạo một AVD mới từ đầu, hoặc sao chép một AVD có sẵn và thay đổi một số đặc tính.

Để tạo một AVD mới:

1. Từ trang danh sách thiết bị ảo của bạn (Your Virtual Devices) trong AVD Manager, nhấp vào **Create Virtual Device**.

   Ngoài ra, chạy ứng dụng của bạn từ bên trong Android Studio. Trong hộp thoại *Select Deployment Target*, nhấp vào **Create New Emulator**.

   Trang *Select Hardware* sẽ xuất hiện.

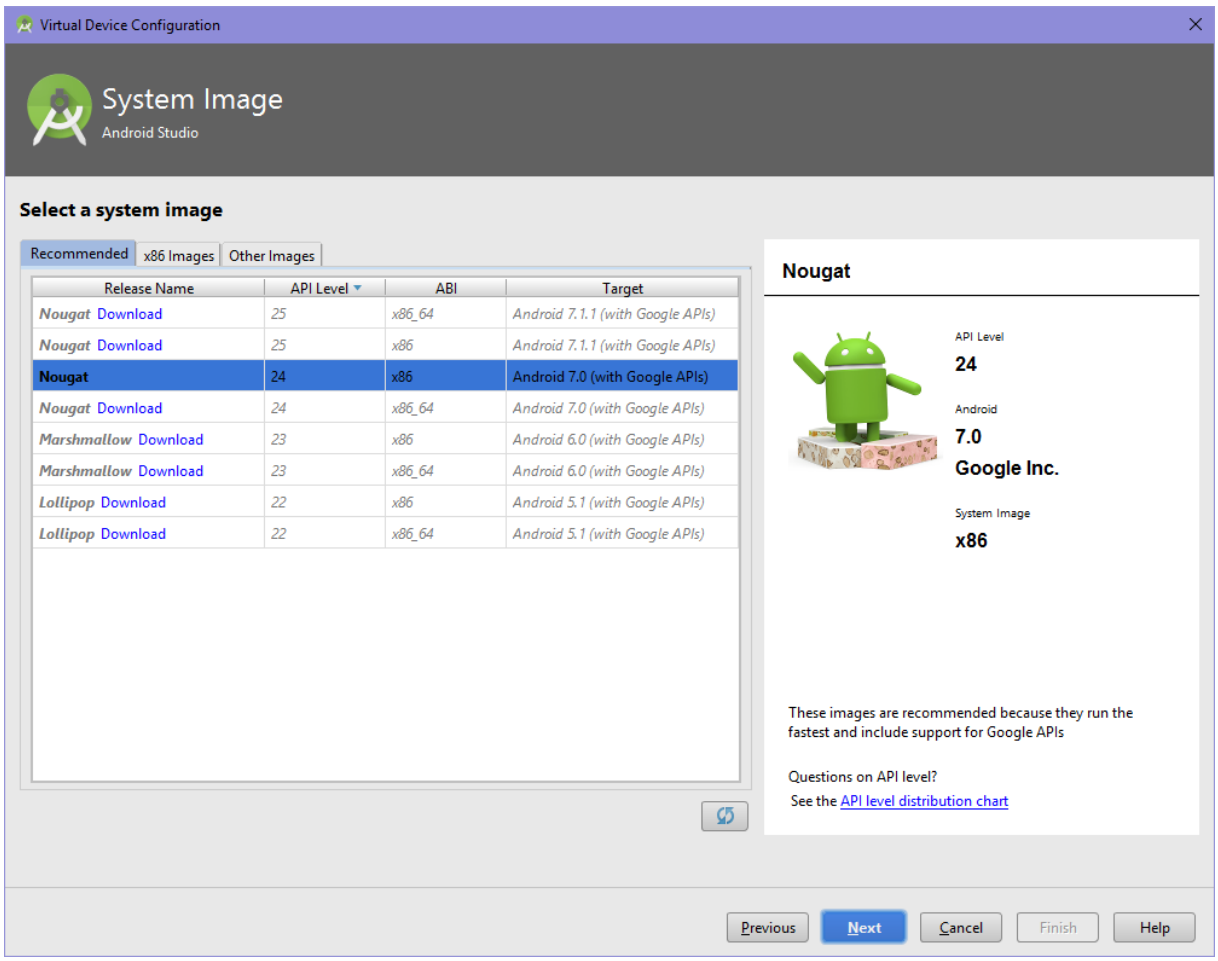


Hình 15 Trang Select Hardware - Lựa chọn cấu hình phần cứng

2. Chọn một cấu hình phần cứng, và sau đó nhấp vào **Next**.

   Nếu bạn không thấy cấu hình phần cứng mà bạn muốn, bạn có thể tạo ra hoặc nhập một cấu hình phần cứng.

   Trang *System Image* xuất hiện.



Hình 16 Trang System Image - Lựa chọn một phiên bản hệ thống sẽ được cài đặt trên AVD

3. Chọn hình ảnh hệ thống cho một cấp độ API cụ thể, và sau đó nhấp vào **Next**.

   Tab **Recommended** thể hiện danh sách hình ảnh hệ thống được khuyến nghị. Các tab khác bao gồm một danh sách hoàn chỉnh hơn. Khung bên phải mô tả hình ảnh hệ thống được chọn.

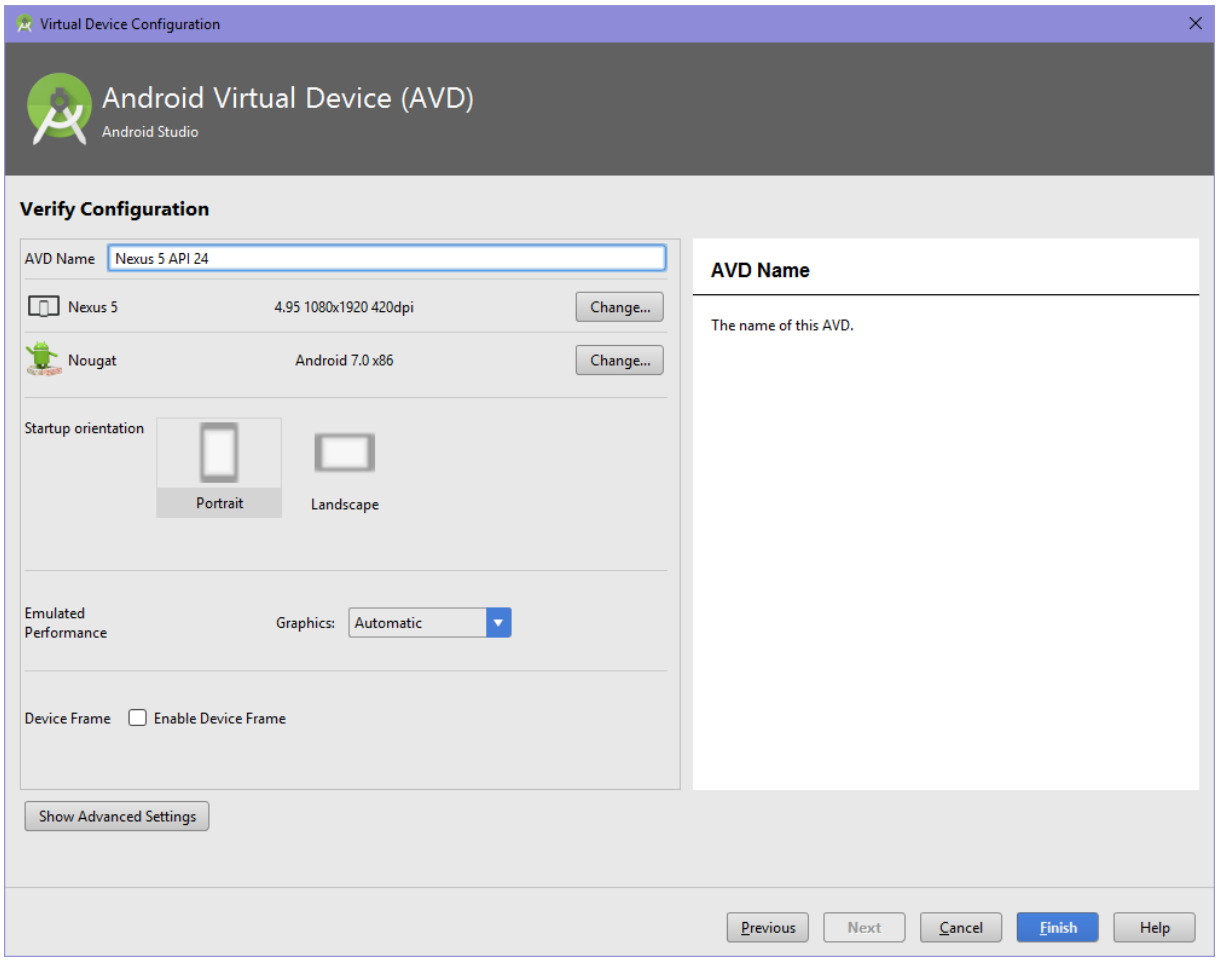
   Nếu bạn nhìn thấy **Download** bên cạnh hệ thống hình ảnh, bạn cần phải bấm vào nó để tải về các hình ảnh hệ thống. Bạn phải kết nối internet để tải nó về.

   Mức API của thiết bị mục tiêu là quan trọng, bởi vì ứng dụng của bạn sẽ không thể chạy trên một hệ thống hình ảnh với một mức độ API ít hơn so với yêu cầu của ứng dụng của bạn, đã được quy định trong thuộc tính minSdkVersion của file manifest. Để biết thêm thông tin về mối liên quan giữa cấp độ API hệ thống và minSdkVersion, xem Versioning Your Apps.

   Nếu ứng dụng của bạn khai báo thành phần <uses-library> trong tập tin manifest, các ứng dụng đòi hỏi phải có một hệ thống hình ảnh, trong đó có sẵn thư viện bên ngoài. Nếu bạn muốn chạy ứng dụng của bạn trên một giả lập, tạo một AVD bao gồm các thư viện cần thiết. Để làm như vậy,

bạn có thể cần phải sử dụng một add-on cho nền tảng AVD; Ví dụ, các API của Google add-on có chứa các thư viện Google Maps.

Trang *Verify Configuration* xuất hiện.



Hình 17 Trang Verify Configuration - Cửa sổ xác nhận các thông tin thiết lập AVD

4. Thay đổi thuộc tính AVD khi cần thiết, và sau đó nhấn **Finish**.

   Nhấp vào **Show Advanced Settings** để hiển thị nhiều hơn các thiết lập, chẳng hạn như giao diện.

   Các AVD mới sẽ xuất hiện trong trang *Your Virtual Devices* hoặc hộp thoại *Select Deployment Target*.

Để tạo một AVD bắt đầu với một bản sao:

1. Từ trang *Your Virtual Devices* của AVD Manager, kích chuột phải vào một AVD và chọn **Duplicate**.

Hoặc bấm Menu ▼ và chọn **Duplicate**.

Trang *Verify Configuration* xuất hiện.

2. Nhấp vào **Change** or **Previous** nếu bạn cần thực hiện thay đổi trong trang *System Image* và *Select Hardware*.
3. Thực hiện thay đổi của bạn, và sau đó nhấn **Finish**.

Các AVD sẽ xuất hiện trong trang *Your Virtual Devices*.

- *Thiết lập cấu hình thiết bị*

AVD Manager cung cấp cấu hình phần cứng được xác định trước cho các thiết bị thông thường nên bạn có thể dễ dàng thêm chúng vào các định nghĩa AVD bạn. Nếu bạn cần phải mô tả một thiết bị khác, bạn có thể tạo một profile phần cứng mới. Bạn có thể định nghĩa một cấu hình phần cứng mới từ đầu, hoặc sao chép một cấu hình phần cứng như một sự bắt đầu. Các cấu hình phần cứng cài đặt sẵn không thể chỉnh sửa.

Để tạo một cấu hình phần cứng mới từ đầu:

1. Trong trang *Select Hardware*, chọn **New Hardware Profile**.
2. Trong trang *Configure Hardware Profile*, thay đổi các thuộc tính cấu hình phần cứng khi cần thiết.
3. Nhấn **Finish**.

   Cấu hình phần cứng mới của bạn sẽ xuất hiện trong trang *Select Hardware*. Bạn có thể tùy chọn tạo một AVD mà sử dụng các cấu hình phần cứng bằng cách nhấn **Next**. Hoặc nhấn **Cancel** để trở về trang *Your Virtual Devices* hoặc hộp thoại *Select Deployment Target*.

Để tạo một cấu hình phần cứng bắt đầu với một bản sao:

1. Trong trang *Select Hardware*, chọn một cấu hình phần cứng và nhấp **Clone Device**.

   Hoặc kích chuột phải vào một cấu hình phần cứng và chọn **Clone**.

2. Trong trang *Configure Hardware Profile*, thay đổi các thuộc tính cấu hình phần cứng khi cần thiết.
3. Nhấn **Finish**.

   Cấu hình phần cứng mới của bạn sẽ xuất hiện trong trang *Select Hardware*. Bạn có thể tùy chọn tạo một AVD mà sử dụng các cấu hình phần cứng bằng cách nhấn **Next**. Hoặc nhấn **Cancel** để trở về trang *Your Virtual Devices* hoặc hộp thoại *Select Deployment Target*.

- *Làm việc với AVD có sẵn*

Từ trang *Your Virtual Devices*, bạn có thể thực hiện các hoạt động sau đây trên một AVD hiện có:

- Để chỉnh sửa một AVD, nhấp vào **Edit** và thực hiện thay đổi của bạn.
- Để xóa một AVD, kích chuột phải vào một AVD và chọn **Delete**. Hoặc bấm Menu và chọn **Delete**.
- Để hiển thị các tập tin .ini và .img có liên quan đến AVD, kích chuột phải vào một AVD và chọn **Show on Disk**. Hoặc bấm Menu và chọn **Show on Disk**.
- Để xem chi tiết cấu hình AVD mà bạn có thể bao gồm trong bất kỳ báo cáo lỗi cho các nhóm Android Studio, kích chuột phải vào một AVD và chọn **View Details**. Hoặc bấm Menu và chọn **View Details**.

➢ *Làm việc với profile thiết bị có sẵn*

Từ trang *Select Hardware*, bạn có thể thực hiện các hoạt động sau đây trên một cấu hình phần cứng hiện có:

- Để chỉnh sửa một cấu hình phần cứng, chọn nó và nhấp vào **Edit Device**. Hoặc kích chuột phải vào một cấu hình phần cứng và chọn **Edit**. Tiếp theo, thực hiện thay đổi của bạn.
- Để xóa một cấu hình phần cứng, kích chuột phải vào nó và chọn **Delete**.

Bạn không thể chỉnh sửa hoặc xóa các cấu hình phần cứng được xác định trước.

➢ *Khởi chạy, tạm dừng và xóa dữ liệu chương trình giả lập*

Từ trang *Your Virtual Devices*, bạn có thể thực hiện các hoạt động sau đây trên một giả lập:

- Để chạy một giả lập mà sử dụng một AVD, kích đúp vào AVD. Hoặc nhấn Launch .
- Để dừng một giả lập đang chạy, kích chuột phải vào AVD và chọn **Stop**. Hoặc bấm Menu và chọn **Stop**.
- Để xóa các dữ liệu cho một giả lập, và quay lại trạng thái giống như khi nó lần đầu tiên được định nghĩa, kích chuột phải vào AVD và chọn **Wipe Data**. Hoặc bấm Menu và chọn **Wipe Data**.

➢ *Nhập và trích xuất profile thiết bị*

Từ trang *Select Hardware*, bạn có thể nhập và trích xuất cấu hình phần cứng:

- Để nhập một cấu hình phần cứng, kích **Import Hardware Profiles** và chọn tập tin XML chứa các định nghĩa trên máy tính của bạn.

- Để trích xuất một cấu hình phần cứng, kích chuột phải vào nó và chọn **Export**. Xác định vị trí nơi bạn muốn lưu trữ các tập tin XML chứa các định nghĩa.

➢ *Các thuộc tính của cấu hình thiết bị*

Bạn có thể chỉ định các thuộc tính sau đây của các cấu hình phần cứng trong trang *Configure Hardware Profile*. Các thuộc tính cấu hình AVD ghi đè các thuộc tính cấu hình phần cứng, và các đặc tính giả lập mà bạn thiết lập trong khi mô phỏng được sẽ chạy đè lên cả hai (Hardware Profile Properties và AVD Properties).

Các cấu hình phần cứng được xác định trước bao gồm cả AVD Manager không thể chỉnh sửa. Tuy nhiên, bạn có thể sao chép chúng và chỉnh sửa các bản sao.

Hardware Profile Property	Mô tả
Device Name	Tên của cấu hình phần cứng. Tên có thể chứa các chữ cái in hoa hoặc chữ thường, số từ 0-9, dấu chấm (.), Dấu gạch dưới (_) dấu ngoặc đơn (()) và khoảng trắng. Tên của tập tin lưu trữ các cấu hình phần cứng được bắt nguồn từ tên cấu hình phần cứng.
Device Type	Chọn một trong các lựa chọn sau: • Phone/Tablet • Android Wear • Android TV
Screen Size	Kích thước vật lý của màn hình, đơn vị inch, đo theo đường chéo màn hình. Nếu kích thước lớn hơn so với màn hình máy tính của bạn, nó sẽ tự giảm kích thước lúc khởi động.
Screen Resolution	Chiều rộng và chiều cao theo kiểu pixel để xác định tổng số điểm ảnh trên màn hình mô phỏng.
Round	Tích chọn tùy chọn này nếu thiết bị có màn hình tròn, chẳng hạn như một số thiết bị Android Wear.
Memory: RAM	Đơn vị kích thước bộ nhớ RAM: B (byte), KB (kilobyte), MB (megabyte), GB (gigabyte), hoặc TB (terabyte).
Input: Has Hardware Buttons (Back/Home/Menu)	Tích chọn tùy chọn này nếu thiết bị của bạn có nút điều hướng phần cứng. Hãy bỏ chọn nó nếu các nút này chỉ thực hiện trong phần mềm. Nếu bạn chọn tùy chọn này, các nút sẽ không xuất hiện trên màn hình. Bạn có thể sử dụng bảng điều khiển bên giả lập để "nhấn" các nút.
Input: Has Hardware	Tích chọn tùy chọn này nếu thiết bị của bạn có một bàn phím cứng. Hãy bỏ

Keyboard	chọn nếu nó không có. Nếu bạn chọn tùy chọn này, một bàn phím sẽ không xuất hiện trên màn hình. Bạn có thể sử dụng bàn phím máy tính của bạn để gửi tổ hợp phím để giả lập.
Navigation Style	Chọn một trong các lựa chọn sau: • None – Không có điều khiển phần cứng. Điều hướng thông qua phần mềm. • D-pad – hỗ trợ Directional Padt. • Trackball • Wheel Các tùy chọn này để điều khiển phần cứng thực tế trên chính thiết bị. Tuy nhiên, những sự kiện được gửi đến thiết bị bằng điều khiển bên ngoài đều giống nhau.
Supported Device States	Chọn một hoặc cả 2 lựa chọn: • Portrait – Thiết bị theo chiều dọc. • Landscape – Thiết bị theo chiều ngang. Nếu bạn chọn cả hai, bạn có thể chuyển đổi giữa các định hướng trong mô phỏng. Bạn phải chọn ít nhất một tùy chọn để tiếp tục.
Cameras	Để bật chức năng camera, chọn một hoặc cả 2 lựa chọn: • Back-Facing Camera • Front-Facing Camera Bạn có thể sử dụng một webcam hoặc một bức ảnh được cung cấp bởi trình giả lập để mô phỏng việc chụp ảnh với camera.
Sensors: Accelerometer	Máy đo gia tốc kế - tích chọn nếu thiết bị có phần cứng giúp xác định phương hướng của nó.
Sensors: Gyroscope	Con quay hồi chuyển được dùng kết hợp với gia tốc kế trong các thiết bị di động hiện đại ngày nay, đặc biệt là điện thoại thông minh và máy tính bảng. Gia tốc kế hỗ trợ việc tính toán gia tốc tuyến tính tương đối so với khung tham chiếu – hệ qui chiếu (frame of reference). Nó dùng để nhận biết thiết bị đang nằm ngang hay đang đứng, từ đó điều chỉnh khung hình thành chế độ portrait hoặc landscape và áp dụng vào các trò chơi cần cảm biến chuyển động như đua xe hoặc các trò chơi tương tác ảo
Sensors: GPS	Tích chọn nếu thiết bị có phần cứng hỗ trợ hệ thống định vị toàn cầu (GPS) hệ thống định vị vệ tinh.
Sensors: Proximity Sensor	Chọn nếu thiết bị có phần cứng để phát hiện nếu điện thoại ở gần khuôn mặt của bạn trong một cuộc gọi điện thoại để vô hiệu hóa input từ màn hình.
Default Skin	Chọn một skin - thiết bị trông như thế nào khi hiển thị trên trình giả lập. Hãy nhớ rằng chỉ định kích thước màn hình là quá lớn so với độ phân giải, thì màn hình sẽ cắt gọn lại, vì vậy bạn không thể nhìn thấy toàn bộ màn hình. Xem Creating Emulator Skins để biết thêm thông tin.

➢ *Các thuộc tính của AVD*

Bạn có thể chỉ định các thuộc tính sau cho các cấu hình AVD trong trang *Verify Configuration*. Các cấu hình AVD xác định sự tương tác giữa máy tính phát triển và trình giả lập, cũng như các thuộc tính bạn muốn ghi đè trong các cấu hình phần cứng.

Các thuộc tính cấu hình AVD ghi đè các thuộc tính cấu hình phần cứng. Các thuộc tính giả lập mà bạn thiết lập trong khi mô phỏng sẽ ghi đè lên cả hai (Hardware Profile Properties và AVD Properties).

AVD Property	Mô tả
AVD Name	Tên của AVD. Tên có thể chứa các chữ cái in hoa hoặc chữ thường, các số từ 0-9, dấu chấm (.), dấu gạch dưới (_), dấu ngoặc đơn ((()), dấu gạch ngang (-), và khoảng trắng. Tên của tập tin lưu trữ các cấu hình AVD có nguồn gốc từ tên AVD.
AVD ID (Advanced)	Các AVD tên tập tin có nguồn gốc từ các ID, và bạn có thể sử dụng ID để tham khảo các AVD từ dòng lệnh.
Hardware Profile	Nhấn **Change** để chọn một cấu hình phần cứng khác nhau trong trang *Select Hardware*.
System Image	Nhấn **Change** để chọn một hệ thống hệ thống khác nhau trong trang *System Image*. Yêu cầu kết nối Internet để tải ảnh hệ thống.
Startup Orientation	Tích chọn để định hướng giả lập ban đầu: • Portrait – Định hướng theo chiều dọc. • Landscape – Định hướng theo chiều ngang. Lựa chọn chỉ được kích hoạt nếu nó được chọn trong cấu hình phần cứng. Khi chạy AVD trong giả lập, bạn có thể thay đổi hướng nếu cả Portrait và Landscape được hỗ trợ trong các cấu hình phần cứng.
Camera (Advanced)	Để bật tùy chọn camera, lựa chọn một hoặc cả hai: • Front • Back Tùy chọn này chỉ có sẵn nếu nó được chọn trong cấu hình phần cứng; nó không có sẵn cho Android Wear và Android TV.
Network: Speed (Advanced)	Chọn một giao thức mạng để xác định tốc độ truyền dữ liệu: • GSM - Global System for Mobile Communications • HSCSD - High-Speed Circuit-Switched Data • GPRS - Generic Packet Radio Service • EDGE - Enhanced Data rates for GSM Evolution • UMTS - Universal Mobile Telecommunications System • HSPDA - High-Speed Downlink Packet Access

	• Full (mặc định) – Truyền dữ liệu với tốc độ mà máy tính bạn cho phép.
Network: Latency (Advanced)	Chọn một giao thức mạng để thiết lập bao nhiêu thời gian (độ trễ) cần cho giao thức để chuyển một gói dữ liệu từ một điểm này đến một điểm khác.
Emulated Performance: Graphics	Chọn cách đồ họa được render trong giả lập: • Hardware – Sử dụng card đồ họa máy tính của bạn để hiển thị nhanh hơn. • Software – Mô phỏng đồ họa trong phần mềm, nó sẽ hữu ích nếu bạn gặp vấn đề với card đồ họa của mình. • Automatic – Hãy để giả lập quyết định lựa chọn tốt nhất dựa trên card đồ họa của bạn.
Multi-Core CPU (Advanced)	Chọn số lượng nhân xử lý trên máy tính mà bạn muốn sử dụng để mô phỏng. Sử dụng nhiều nhân xử lý sẽ tăng tốc độ giả lập.
Memory and Storage: RAM	Dung lượng RAM trên thiết bị. Giá trị này được thiết lập bởi các nhà sản xuất phần cứng, nhưng bạn có thể ghi đè lên nó nếu cần, để việc giả lập nhanh hơn. Nhập kích thước bộ nhớ RAM và chọn hơn đị: B (byte), KB (kilobyte), MB (megabyte), GB (gigabyte), hoặc TB (terabyte).
Memory and Storage: VM Heap	Kích thước VM heap. Giá trị này được thiết lập bởi nhà sản xuất phần cứng, nhưng bạn có thể ghi đè lên nó nếu cần. Nhập kích thước và chọn đơn vị: B (byte), KB (kilobyte), MB (megabyte), GB (gigabyte), hoặc TB (terabyte). Để biết thêm thông tin về máy ảo Android, xem tại Memory Management for Different Virtual Machines.
Memory and Storage: Internal Storage	Kích thước bộ nhớ trong trên thiết bị. Giá trị này được thiết lập bởi nhà sản xuất phần cứng, nhưng bạn có thể ghi đè lên nó nếu cần. Nhập kích thước và chọn đơn vị: B (byte), KB (kilobyte), MB (megabyte), GB (gigabyte), hoặc TB (terabyte).
Memory and Storage: SD Card	Kích thước thẻ nhớ. Để sử dụng thẻ nhớ ảo được quản lý bởi Android Studio, chọn **Studio-managed**, nhập kích thước và chọn đơn vị: B (byte), KB (kilobyte), MB (megabyte), GB (gigabyte), hoặc TB (terabyte). Giá trị nhỏ nhất là 100 MB được khuyến nghị để sử dụng camera. Để biết thêm thông tin, xem tại mksdcard.
Device Frame: enable Device Frame	Select to enable a frame around the emulator window that mimics the look of a real device.
Custom Skin Definition (Advanced)	Chọn một skin - thiết bị trông như thế nào khi hiển thị trên trình giả lập. Hãy nhớ rằng chỉ định kích thước màn hình là quá lớn so với độ phân giải, thì màn hình sẽ cắt gọn lại, vì vậy bạn không thể nhìn thấy toàn bộ màn hình. Xem Creating Emulator Skins để biết thêm thông tin.
Keyboard: Enable Keyboard Input (Advanced)	Chọn tùy chọn này nếu bạn muốn sử dụng bàn phím cứng của bạn để tương tác với các mô phỏng. Nó bị hiệu hóa đối với Android Wear và Android TV.

➢ *Tạo giao diện giả lập (Emulator Skins)*

Một giao diện (Skin) giả lập Android là một bộ các tập tin xác định các thành phần trực quan và điều khiển của một màn hình giả lập. Nếu định nghĩa giao diện có sẵn trong các thiết lập AVD không đáp ứng yêu cầu của bạn, bạn có thể tạo giao diện tùy chỉnh của riêng bạn, và sau đó áp dụng nó cho AVD của bạn.

Mỗi giao diện giả lập chứa:

- Một tập tin hardware.ini
- Các layout file cho hỗ trợ định hướng (theo chiều ngang, chiều dọc) và cấu hình vật lý
- Các file ảnh cho thành phần hiển thị, chẳng hạn như nền (background), các phím và các nút bấm.

Để tạo và sử dụng một giao diện tùy chỉnh:

1. Tạo một thư mục mới, nơi bạn sẽ lưu các tập tin cấu hình giao diện của bạn.
2. Xác định diện mạo trực quan của giao diện trong một tập tin văn bản có tên là layout. Tập tin này xác định nhiều đặc tính của giao diện, chẳng hạn như kích thước và hình ảnh áp dụng cho các nút cụ thể. Ví dụ:

```
parts {
  device {
    display {
      width   320
      height  480
      x       0
      y       0
    }
  }

  portrait {
    background {
      image background_port.png
    }

    buttons {
      power {
        image  button_vertical.png
        x 1229
        y 616
      }
    }
  }
  ...
}
```

3. Thêm các tập tin bitmap của thiết bị trong cùng một thư mục.

4. Bổ sung cấu hình thiết bị phần cứng cụ thể trong tập tin hardware.ini cho các cài đặt thiết bị, chẳng hạn như hw.keyboard và hw.lcd.density.
5. Lưu trữ các tập tin trong thư mục giao diện và chọn các tập tin lưu trữ như một giao diện tùy chỉnh.

Để biết thông tin chi tiết hơn về việc tạo skin giả lập, xem Android Emulator Specification Skin File trong công cụ mã nguồn.

2.3Tạo giao diện (layout) chương trình trong Android Studio

Một layout định nghĩa cấu trúc trực quan cho một giao diện người dùng, chẳng hạn như giao diện người dùng cho một activity hay widget. Bạn có thể khai báo một layout theo hai cách:

- **Khai báo các thành phần giao diện người dùng trong XML**. Android cung cấp một từ khóa XML đơn giản tương ứng với các lớp View và lớp con, chẳng hạn như đối với các vật dụng và bố trí.
- **Khởi tạo layout tại thời điểm ứng dụng chạy**. Ứng dụng của bạn có thể tạo ra các đối tượng View và ViewGroup (và thao tác với các thuộc tính của chúng) bằng cách lập trình (programmatically).

Framework Android cung cấp cho bạn sự linh hoạt để sử dụng một trong hai hoặc cả hai phương pháp để khai báo và quản lý UI ứng dụng của bạn. Ví dụ, bạn có thể khai báo layout mặc định cho ứng dụng của bạn trong XML, bao gồm cả các thành phần view sẽ xuất hiện và thuộc tính của chúng. Sau đó bạn có thể thêm mã lệnh vào ứng dụng của bạn để thay đổi trạng thái của đối tượng view, bao gồm cả những khai báo ở XML, tại thời điểm chương trình chạy.

Lợi thế của việc khai báo giao diện người dùng của bạn trong XML, là nó cho phép bạn phân tách tốt hơn việc trình bày ứng dụng của bạn. Mô tả giao diện người dùng của bạn được với mã lệnh ứng dụng của bạn, điều đó có nghĩa rằng bạn có thể sửa đổi nó mà không cần phải sửa đổi mã lệnh của bạn và biên dịch lại. Ví dụ, bạn có thể tạo các layout XML để cho màn hình có định hướng khác nhau, kích thước màn hình điện thoại khác nhau, và các ngôn ngữ khác nhau. Ngoài ra, khai báo layout trong XML làm cho nó dễ dàng hơn để hình dung cấu trúc giao diện người dùng của bạn, vì vậy nó dễ dàng hơn để gỡ lỗi.

Nói chung, các thẻ XML để khai báo các thành phần giao diện người dùng, theo sát cấu trúc và cách đặt tên của các lớp và phương thức, nơi tên phần tử tương ứng với tên lớp và tên thuộc tính tương ứng với các phương thức. Trong thực tế, sự tương ứng thường là rất trực quan mà bạn có thể đoán những thuộc tính XML

nào tương ứng với một phương thức của lớp nào, hoặc đoán lớp nào tương ứng với một thành phần XML. Tuy nhiên, lưu ý rằng không phải tất cả từ vựng là giống hệt nhau. Trong một số trường hợp, có những khác biệt nhỏ trong việc đặt tên. Ví dụ, các thành phần EditText có thuộc tính *text* tương ứng với phương thức *EditText.setText()*.

2.3.1 Viết XML

Sử dụng các thẻ XML của Android, bạn có thể nhanh chóng thiết kế layout UI và các yếu tố màn hình mà chúng chứa, theo cùng một cách bạn tạo ra các trang web trong HTML - với một loạt các phần tử lồng nhau.

Mỗi tập tin layout phải chứa chính xác một phần tử gốc, là một đối tượng View hoặc ViewGroup. Một khi bạn đã định nghĩa các thành phần gốc (root), bạn có thể thêm các đối tượng layout bổ sung hoặc các widget như thành phần con để từng bước xây dựng một hệ thống phân cấp View – giúp xác định layout của bạn. Ví dụ, đây là một cách bố trí XML có sử dụng một LinearLayout dọc để giữ một TextView và một Button:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >
    <TextView android:id="@+id/text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Hello, I am a Button" />
</LinearLayout>
```

Sau khi bạn đã khai báo bố trí của bạn trong XML, lưu tập tin với phần mở rộng .xml, trong thư mục **res/layout/** dự án Android của bạn, vì vậy nó sẽ được biên dịch đúng.

Thông tin thêm về cú pháp cho layout tập tin XML có sẵn trong tài liệu Layout Resources.

2.3.2Load tài nguyên XML

Khi bạn biên dịch ứng dụng của bạn, mỗi file layout XML được biên soạn vào tài nguyên View. Bạn nên tải các tài nguyên layout từ mã ứng dụng của bạn, trong lời gọi hàm **Activity.onCreate ()** của bạn. Làm như vậy bằng cách gọi **setContentView ()**, truyền cho nó tham chiếu đến tài nguyên layout của bạn từ: **R.layout.*layout_file_name***. Ví dụ, nếu bố trí XML của bạn đã được lưu là **main_layout.xml**, bạn sẽ tải nó cho Activity của bạn như sau:

```
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main_layout);
}
```

Lời gọi phương thức onCreate () trong Activity của bạn được gọi bởi framework Android, khi Activity của bạn bắt đầu chạy (xem thảo luận về vòng đời, trong tài liệu Activities).

2.3.3Thuộc tính

Mỗi đối tượng View và đối tượng ViewGroup hỗ trợ đa dạng các thuộc tính XML của riêng nó. Một số thuộc tính là đặc trưng cho từng đối tượng View (ví dụ, TextView hỗ trợ thuộc tính textSize), nhưng những thuộc tính này cũng được kế thừa bởi bất kỳ đối tượng View nào mở rộng từ lớp này. Một số tính chất chung cho tất cả đối tượng View, bởi vì chúng được thừa kế từ lớp View gốc (ví dụ như huộc tính id). Và, các thuộc tính khác được coi là "thông số layout", đó là các thuộc tính mô tả định hướng layout của các đối tượng View, theo khai báo của đối tượng cha ViewGroup của đối tượng đó.

- *ID*

Bất kỳ đối tượng View nào cũng có một ID liên kết với nó, để nhận diện các View. Khi ứng dụng được biên dịch, ID này được tham chiếu như là một số nguyên, nhưng ID thường được gán trong file layout XML như là một chuỗi, trong thuộc tính id. Đây là thuộc tính XML chung cho tất cả đối tượng View (được xác định bởi các lớp View) và bạn sẽ sử dụng nó rất thường xuyên. Cú pháp cho một ID, bên trong một thẻ XML là:

```
android:id="@+id/my_button"
```

Biểu tượng (@) ở đầu của chuỗi chỉ ra rằng các phân tích cú pháp XML nên phân tích và mở rộng phần còn lại của chuỗi ID và xác định nó như là một nguồn tài nguyên ID. Biểu tượng (+) có nghĩa rằng đây là tên một tài nguyên mới mà phải được tạo ra và thêm vào các nguồn tài nguyên của chương trình (trong file R.java). Có nhiều tài nguyên ID khác được cung cấp bởi các framework Android. Khi tham chiếu một tài nguyên ID Android, bạn không cần các biểu tượng (+), nhưng phải thêm không gian tên **android**, như sau:

```
android:id="@android:id/empty"
```

Với không gian tên **android**, chương trình đang tham chiếu ID từ lớp tài nguyên **android.R** , thay vì lớp tài nguyên cục bộ (local).

Để tạo View và tham chếu đến chúng từ chương trình, cách thức thực hiện phổ biến là:

1. Định nghĩa một view/ widget trong file layout và gán cho nó một ID duy nhất:

```
<Button android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/my_button_text"/>
```

Sau đó tạo một thể hiện (instance) của đối tượng view và bắt nó từ layout (thường được viết trong phương thức **onCreate()**):

```
Button myButton = (Button) findViewById(R.id.my_button);
```

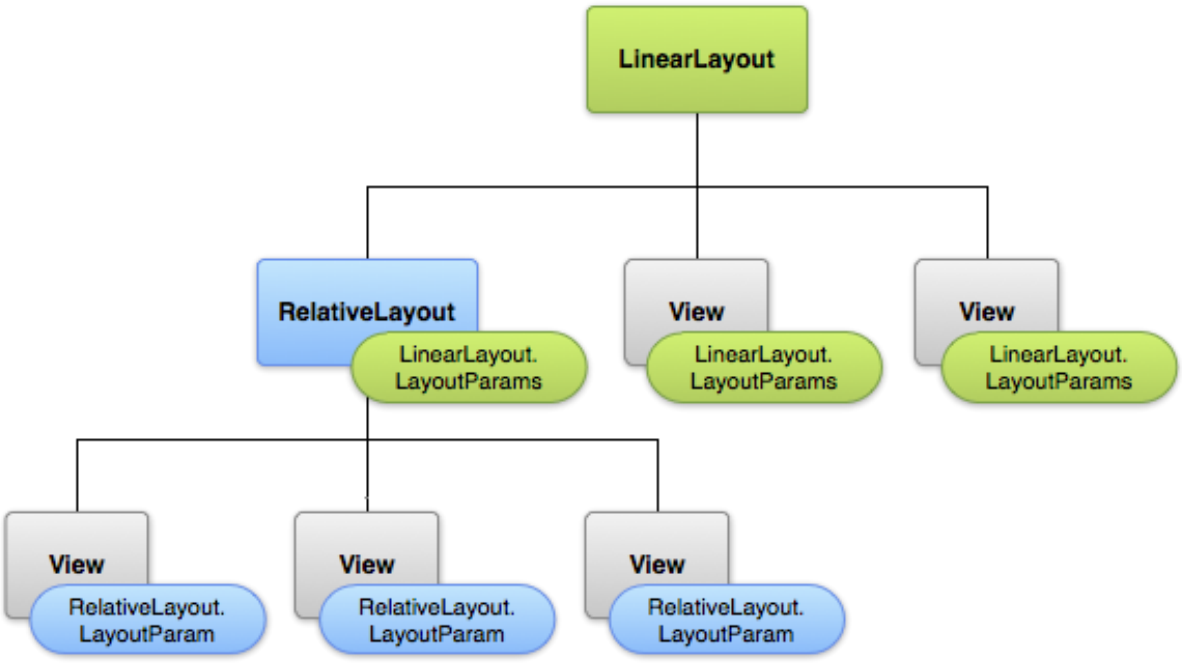
Xác định ID cho đối tượng view là quan trọng khi tạo một RelativeLayout. Trong relative layout, view anh chị em có thể xác định layout của chúng liên quan đến các view khác, được tham chiếu bởi ID duy nhất.

Một ID không cần phải là duy nhất trong suốt toàn bộ cây, nhưng nó phải là duy nhất trong một phần của cây mà bạn đang tìm kiếm (mà thường có thể là toàn bộ cây, vì vậy tốt nhất là hoàn toàn riêng biệt nếu có thể).

- *Các tham số layout*

Các thuộc tính layout XML được đặt tên là **layout_*something*** xác định các tham số layout cho View đó là thích hợp cho các ViewGroup mà nó cư trú.

Mỗi lớp ViewGroup thực hiện một lớp lồng nhau mà mở rộng từ ViewGroup.LayoutParams. Lớp này chứa các kiểu thuộc tính nhằm xác định kích thước và vị trí cho từng View con. Như bạn có thể thấy trong hình dưới, các group view cha mẹ xác định các thông số layout cho từng view con (bao gồm cả view group con).



Lưu ý rằng tất cả các lớp con LayoutParams có cú pháp riêng của mình để thiết lập giá trị. Mỗi phần tử con phải xác định LayoutParams thích hợp với phần tử cha của nó, mặc dù nó cũng có thể xác định LayoutParams khác nhau cho phần tử con của riêng nó.

Tất cả các view group có chiều rộng và chiều cao (**layout_width** và **layout_height**), và từng view cần thiết phải có để xác định chúng. Nhiều LayoutParams cũng bao gồm các tùy chọn **margins** và **bordes**.

Bạn có thể chỉ định chiều rộng và chiều cao với các phép đo chính xác, mặc dù có thể bạn sẽ không muốn làm điều này thường xuyên. Thông thường, bạn sẽ sử dụng một trong các hằng số để thiết lập chiều rộng hoặc chiều cao:

- ***wrap_content***: view của bạn sẽ tự điều chỉnh kích thước của nó theo yêu cầu kích thước của nội dung mà nó chứa.
- ***match_parent:*** view của bạn sẽ lớn hết cỡ mà phần tử cha chứa nó cho phép.

Nói chung, chỉ định chiều rộng và chiều cao layout sử dụng các đơn vị tuyệt đối như các điểm ảnh là không nên. Thay vào đó, sử dụng các phép đo tương đối như các đơn vị mật độ điểm ảnh độc lập (dp), wrap_content, hoặc match_parent, là một phương pháp tốt hơn, vì nó giúp đảm bảo rằng ứng dụng của bạn sẽ hiển thị đúng trên một loạt các màn hình điện thoại có kích cỡ khác nhau. Các đơn vị đo được quy định trong tài liệu Available Resources.

2.3.4Các layout phổ biến

Mỗi lớp con của lớp ViewGroup cung cấp một cách riêng để hiển thị các view mà bạn lồng vào bên trong nó. Dưới đây là một số loại layout phổ biến được xây dựng trong nền tảng Android.

Lưu ý: Mặc dù bạn có thể lồng một hoặc nhiều layout trong một layout khác để đạt được thiết kế giao diện người dùng của bạn, bạn nên cố gắng để giữ cho hệ thống phân cấp layout của bạn càng nông càng tốt. layout của bạn thể hiện ra nhanh hơn nếu nó có ít layout lồng nhau (một hệ thống phân cấp view theo chiều rộng là tốt hơn so với một hệ thống phân cấp view theo chiều sâu).

Linear Layout

Một layout tổ chức các thành phần con của nó thành một hàng ngang hoặc dọc duy nhất. Nó tạo ra một thanh cuộn nếu chiều dài của cửa sổ vượt quá độ dài của màn hình.

Relative Layout

Cho phép bạn xác định vị trí của các đối tượng con tương quan với nhau (phần tử A nằm phía bên trái của phần tử B) hoặc tương quan với phần tử cha (liên kết lên phía trên với phần tử cha).

Web View

Hiển thị các trang web

2.3.5Xây dựng layout với Adapter

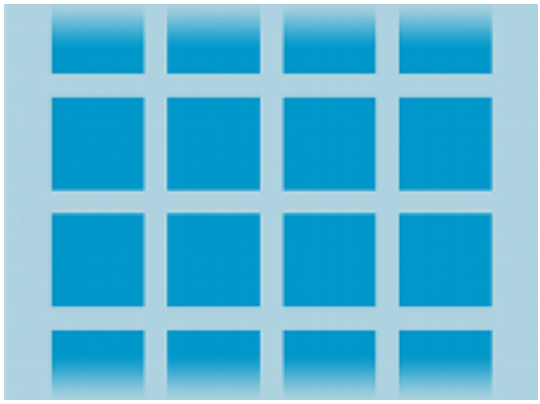
Khi nội dung cho layout của bạn là kiểu động hoặc không xác định trước, bạn có thể sử dụng một layout là lớp con AdapterView để bố trí cho view tại thời điểm chương trình chạy. Một lớp con của lớp AdapterView sử dụng một Adapter để kết nối dữ liệu tới layout của nó. Adapter hoạt động như một người trung gian giữa các nguồn dữ liệu và layout AdapterView - Adapter lấy dữ liệu (từ một nguồn ví dụ như một mảng hoặc một truy vấn cơ sở dữ liệu) và chuyển đổi từng mục vào một view mà có thể được thêm vào trong layout AdapterView.

Layout phổ biến được hỗ trợ bởi adapter bao gồm :

List View

Hiển thị danh sách trên một cột duy nhất có thể cuộn (lên hoặc xuống)

Grid View



Hiển thị một lưới (bao gồm các hàng và cột) có thể cuộn

➢ *Đổ dữ liệu vào adapter view*

Bạn có thể điền một AdapterView như ListView hoặc GridView bằng cách gắn một thực hiện AdapterView vào Adapter – thứ dùng để nhận dữ liệu từ một nguồn bên ngoài và tạo ra một View đại diện cho từng mục dữ liệu.

Android cung cấp một số lớp con của Adapter, rất hữu ích cho việc lấy các loại dữ liệu khác nhau và xây dựng cho view cho AdapterView. Hai adapter phổ biến nhất là:

a) ArrayAdapter

Sử dụng adapter này khi nguồn dữ liệu của bạn là một mảng. Theo mặc định, ArrayAdapter, tạo view cho từng item trong array bằng cách gọi phương thức **toString()** trên mỗi item và đặt nội dung trong một TextView.

Ví dụ, nếu bạn có một mảng chuỗi (string) bạn muốn hiển thị trong một ListView, khởi tạo một ArrayAdapter mới sử dụng một hàm khởi tạo (constructor) để chỉ định layout cho từng chuỗi và mảng chuỗi (string array):

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, myStringArray);
```

Các đối số trong hàm khởi tạo này là:

- Context ứng dụng của bạn
- Layout có chứa một TextView cho từng chuỗi trong mảng
- Mảng chuỗi

Sau đó chỉ cần gọi **setAdapter()** trên ListView của bạn:

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

Để tùy chỉnh sự xuất hiện của từng item, bạn có thể ghi đè lên phương thức **toString()** cho các đối tượng trong mảng của bạn. Hoặc, để tạo ra một view cho từng item với là một cái gì đó khác hơn là một TextView (ví dụ, nếu bạn muốn một ImageView cho mỗi item trong mảng), mở rộng lớp ArrayAdapter và ghi đè **getView()** để trả về kiểu view bạn muốn cho mỗi mục.

a) SimpleCursorAdapter

Sử dụng adater này khi dữ liệu của bạn đến từ một con trỏ (Cursor). Khi sử dụng **SimpleCursorAdapter**, bạn phải xác định layout sử dụng cho từng hàng trong Cursor và cột nào trong Cursor nên được chèn vào view nào của layout. Ví dụ, nếu bạn muốn tạo ra một danh sách các tên người và số điện thoại, bạn có thể thực hiện một truy vấn mà trả về một con trỏ chứa một hàng cho mỗi người và các cột cho tên và số điện thoại. Sau đó bạn tạo một mảng chuỗi, chỉ rõ cột nào trong Cursor bạn muốn xuất hiện trong layout và một mảng số nguyên xác định những view tương ứng với mỗi cột sẽ được bố trí vào:

```
String[] fromColumns = {ContactsContract.Data.DISPLAY_NAME,
                ContactsContract.CommonDataKinds.Phone.NUMBER};
int[] toViews = {R.id.display_name, R.id.phone_number};
```

Khi bạn khởi tạo các SimpleCursorAdapter, truyền layout sử dụng cho từng kết quả, Cursor chứa các kết quả, và hai mảng này:

```
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
      R.layout.person_name_and_number, cursor, fromColumns, toViews, 0);
ListView listView = getListView();
listView.setAdapter(adapter);
```

Sau đó SimpleCursorAdapter tạo ra view cho từng hàng trong Cursor bằng cách sử dụng layout được cung cấp bằng cách chèn từng mục *fromColumns* vào toViews tương ứng.

Nếu trong quá trình chạy ứng dụng của bạn, bạn thay đổi các dữ liệu cơ bản được đọc bởi adapter, bạn nên gọi **notifyDataSetChanged()**. Điều này sẽ thông báo cho view rằng các dữ liệu đã được thay đổi và nó sẽ tự làm mới lại với các thay đổi.

- *Xử lý sự kiện click*

Bạn có thể phản hồi các sự kiện click trên mỗi item trong một AdapterView bằng cách thực hiện các giao tiếp **AdapterView.OnItemClickListener**. Ví dụ:

```
// Create a message handling object as an anonymous class.
private OnItemClickListener mMessageClickedHandler = new OnItemClickListener() {
    public void onItemClick(AdapterView parent, View v, int position, long id) {
        // Do something in response to the click
    }
};

listView.setOnItemClickListener(mMessageClickedHandler);
```

2.4Các điều khiển đầu vào (Input Controls)

Điều khiển đầu vào là các thành phần tương tác trong giao diện người dùng của ứng dụng của bạn. Android cung cấp một loạt các điều khiển bạn có thể sử dụng trong giao diện người dùng của bạn, chẳng hạn như các nút bấm (button), các trường văn bản, thanh tìm kiếm (seek bar), hộp kiểm (checkbox), các nút zoom (zoom button), nút bật tắt (toggle button), và nhiều thành phần khác.

Thêm một điều khiển đầu vào trong giao diện người dùng của bạn là đơn giản như thêm một phần tử XML vào layout XML của bạn. Ví dụ, đây là một cách layout với một trường văn bản và nút bấm:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button android:id="@+id/button_send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send"
        android:onClick="sendMessage" />
</LinearLayout>
```

Mỗi điều khiển đầu vào hỗ trợ một tập hợp các sự kiện đầu vào (input event), do đó bạn có thể xử lý các sự kiện như khi người dùng nhập văn bản hoặc chạm vào một nút.

2.4.1Các điều khiển thông dụng

Dưới đây là danh sách một số điều khiển thông dụng mà bạn có thể sử dụng trong các ứng dụng của bạn. Ấn vào các liên kết để tìm hiểu thêm về việc sử dụng từng điều khiển.

Kiểu điều khiển	Mô tả	Các lớp liên quan
Button	Một nút nhấn được bấm bởi người sử dụng để thực hiện một hành động.	Button
Text field	Một trường văn bản có thể chỉnh sửa. Bạn có thể sử dụng widget AutoCompleteTextView để tạo widget nhập văn bản có cung cấp các đề xuất tự động hoàn tất.	EditText, AutoCompleteTextView
Checkbox	Trạng thái chọn/ không chọn có thể được chuyển đổi bởi người dùng. Bạn nên sử dụng hộp kiểm khi trình bày cho người dùng một nhóm các tùy chọn mà không phải là loại trừ lẫn nhau.	CheckBox
Radio button	Tương tự như hộp kiểm, ngoại trừ một điều – chỉ duy nhất một lựa chọn có thể được chọn trong nhóm.	RadioGroup RadioButton
Toggle button	Một nút bật/ tắt với một chỉ báo (light indicator).	ToggleButton
Spinner	Một danh sách thả xuống cho phép người dùng lựa chọn một giá từ một tập cho trước.	Spinner
Pickers	Một hộp thoại cho phép người dùng lựa chọn một giá trị duy nhất cho một tập hợp bằng cách sử dụng nút up/down hoặc thông qua thao tác vuốt màn hình. Sử dụng widget DatePicker để để nhập giá trị cho ngày (tháng, ngày, năm) hoặc widget TimePicker để nhập giá trị cho thời gian (giờ, phút, buổi sáng (AM)/ buổi tối (PM)), sẽ được định dạng tự động tùy theo vị trí địa lý của người dùng.	DatePicker, TimePicker

2.4.2Button

Một nút bấm bao gồm văn bản hoặc biểu tượng (hoặc cả văn bản và biểu tượng) mà tiếp xảy ra khi người dùng chạm vào nó.

Hình 18 Các button trên Android

Tùy thuộc vào việc bạn muốn một nút với văn bản, một biểu tượng, hoặc cả hai, bạn có thể tạo các nút trong layout của bạn theo ba cách:

- Với văn bản, sử dụng lớp Button :

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    ... />
```

- Với biểu tượng, Sử dụng lớp ImageButton :

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/button_icon"
    ... />
```

- Với cả văn bản và biểu tượng, sủ dụng lớp Button với thuộc tính android:drawableLeft :

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    android:drawableLeft="@drawable/button_icon"
    ... />
```

➢ *Phản hồi lại sự kiện click*

Khi người dùng bấm vào một nút, các đối tượng button nhận được một sự kiện nhấp chuột.

Để xác định các hàm xử lý sự kiện click cho một nút, thêm thuộc tính android:onClick vào thành phần <Button> trong layout XML của bạn. Giá trị của thuộc tính này phải là tên của phương thức mà bạn muốn gọi để phản hồi lại một sự kiện click. Các Activity lưu trữ layout đó phải thực hiện các phương thức tương ứng.

Ví dụ, đây là một bố trí với một nút sử dụng android:onClick:

```
<?xml version="1.0" encoding="utf-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Trong Activity mà chứa layout này, các phương thức sau xử lý sự kiện click:

```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
}
```

Các phương thức bạn khai báo trong thuộc tính android: onClick phải chính xác như trên. Cụ thể, phương thức này phải:

- public
- Kiểu trả về là void
- View là tham số duy nhất của nó

Sử dụng OnClickListener

Bạn cũng có thể khai báo các hàm xử lý sự kiện click theo cách lập trình chứ không phải là trong layout XML. Đây có thể là điều cần thiết nếu bạn khởi tạo các nút trong lúc chạy chương trình hoặc bạn cần khai báo thao tác nhấp chuột trong một lớp Fragment con.

Để khai báo xử lý sự kiện, tạo ra một đối tượng View.OnClickListener và gán nó vào các nút bằng cách gọi setOnClickListener (View.OnClickListener). Ví dụ:

```
Button button = (Button) findViewById(R.id.button_send);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
```

```
    }
});
```

- *Tạo kiểu cho nút của bạn*

Hình dạng nút bấm của bạn (hình nền và phông chữ) có thể thay ở các thiết bị khác nhau, bởi vì các thiết bị của các nhà sản xuất khác nhau thường có kiểu dáng mặc định khác nhau cho các điều khiển đầu vào.

Bạn có thể kiểm soát chính xác cách điều khiển của bạn mang kiểu dáng sử dụng một chủ đề mà bạn áp dụng cho toàn bộ ứng dụng của bạn. Ví dụ, để đảm bảo rằng tất cả các thiết bị chạy Android 4.0 và cao hơn sử dụng các chủ đề Holo trong ứng dụng của bạn, khai báo android:theme="@android:style/Theme.Holo" trong thẻ <application> thuộc file manifest. Đọc bài viết trên blog, Holo Everywhere để biết thêm thông tin về việc sử dụng các chủ đề Holo trong khi hỗ trợ các thiết bị cũ.

Để tùy chỉnh riêng một nút bấm với một nền tảng khác nhau, xác định thuộc tính android:background với nguồn tài nguyên drawable hoặc color. Ngoài ra, bạn có thể áp dụng *style* cho nút, hoạt động một cách tương tự như style trong HTML để xác định nhiều đặc tính định dạng như nền, font chữ, kích thước, và những thứ khác. Để biết thêm thông tin về việc áp dụng định dạng, xem Styles and Themes.

2.4.3Trường văn bản (Text field)

Mỗi trường văn bản dự kiến một loại văn bản đầu vào nhất định, chẳng hạn như địa chỉ email, số điện thoại, hoặc chỉ là thuần văn bản. Vì vậy, điều quan trọng là bạn chỉ định các loại đầu vào cho từng trường văn bản trong ứng dụng của bạn để hệ thống hiển thị các phương thức nhập liệu thích hợp (chẳng hạn như một bàn phím trên màn hình).

- *Xác định loại bàn phím*

Bạn nên luôn luôn khai khai báo phương thức nhập liệu cho các trường văn bản của bạn bằng cách thêm thuộc tính android:inputType trong thẻ <EditText>.

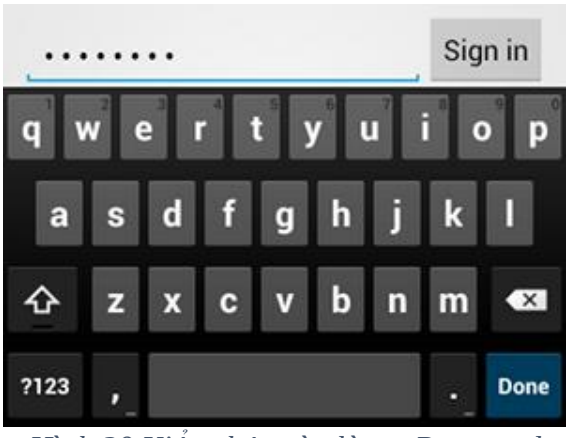
Ví dụ, nếu bạn muốn một phương thức nhập liệu để nhập một số điện thoại, sử dụng giá trị "phone":

```
<EditText
   android:id="@+id/phone"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:hint="@string/phone_hint"
   android:inputType="phone" />
```

Hình 19 Kiểu nhập vào là phone

Hoặc nếu các lĩnh vực văn bản là nhập vào một mật khẩu, sử dụng giá trị "textPassword" để các trường văn bản che giấu phần mật khẩu đã nhập vào của người dùng:

```
<EditText
   android:id="@+id/password"
   android:hint="@string/password_hint"
   android:inputType="textPassword"
   ... />
```



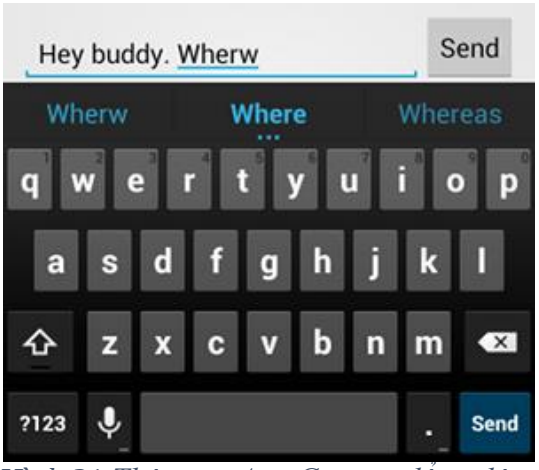
Hình 20 Kiểu nhập vào là textPassword

- *Kích hoạt tính năng gợi ý văn bản và hành vi khác*

Thuộc tính android:inputType cho phép bạn chỉ định các hành vi khác nhau cho phương thức đầu vào. Quan trọng nhất, nếu trường văn bản của bạn là dành cho kiểu văn bản cơ bản (chẳng hạn như cho một tin nhắn văn bản), bạn nên kích hoạt tính năng tự động sửa lỗi chính tả với giá trị "textAutoCorrect".

Bạn có thể kết hợp các hành vi khác nhau và định dạng phương thức nhập liệu với thuộc tính android:inputType. Ví dụ, đây là cách để tạo ra một trường văn bản mà viết hoa chữ đầu tiên của một câu và cũng tự động sửa lỗi chính tả:

```
<EditText
   android:id="@+id/message"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:inputType=
      "textCapSentences|textAutoCorrect"
   ... />
```



Hình 21 Thêm textAutoCorrect để tự động sửa lỗi chính tả

- *Xác định thao tác phương thức đầu vào*

Hầu hết các phương thức nhập liệu cung cấp một nút dùng để thao tác ở góc dưới cùng - phù hợp với trường văn bản hiện hành. Theo mặc định, hệ thống sử dụng nút này đối với một thao tác **Next** hoặc **Done**. Tuy nhiên, bạn có thể chỉ định các thao tác bổ sung mà có thể thích hợp hơn cho trường văn bản của bạn, chẳng hạn như **Send** hoặc **Go**.

Để xác định các nút hành động bàn phím, sử dụng thuộc tính android:imeOptions với giá trị "actionSend" hoặc "actionSearch". Ví dụ:

```
<EditText
   android:id="@+id/search"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:hint="@string/search_hint"
   android:inputType="text"
   android:imeOptions="actionSend" />
```

Hình 22 Nút Send xuất hiện khi bạn khai báo android:imeOptions="actionSend"

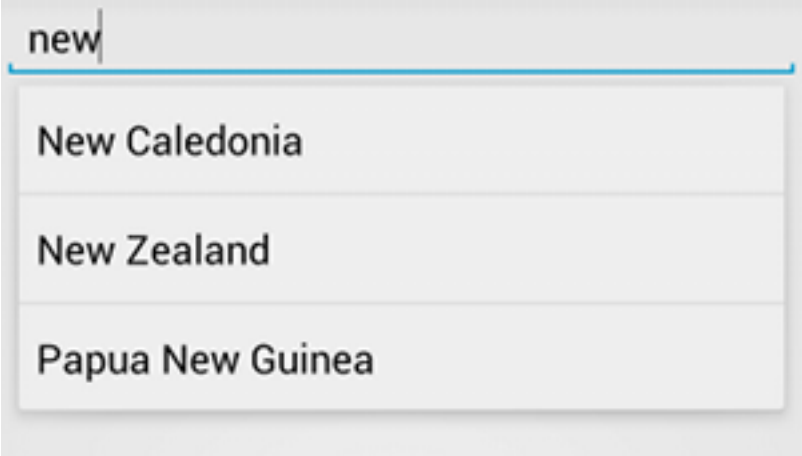
Sau đó bạn có thể bắt sự kiện vào nút bằng cách định nghĩa một TextView.OnEditorActionListener cho các phần tử EditText. Ví dụ:

```
EditText editText = (EditText) findViewById(R.id.search);
editText.setOnEditorActionListener(new OnEditorActionListener() {
   @Override
   public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
      boolean handled = false;
      if (actionId == EditorInfo.IME_ACTION_SEND) {
         sendMessage();
         handled = true;
      }
      return handled;
```

```
    }
});
```

➢ *Cung cấp gợi ý tự động hoàn thành*

Nếu bạn muốn cung cấp gợi ý cho người dùng khi họ nhập liệu, bạn có thể sử dụng một lớp con của EditText là AutoCompleteTextView. Để thực hiện tự động hoàn tất, bạn phải chỉ định một Adapter cung cấp các văn bản đề xuất. Có một số loại adapter có sẵn, tùy thuộc vào nơi các dữ liệu xuất phát, chẳng hạn như từ một cơ sở dữ liệu hoặc một mảng.



Hình 23 Ví dụ về AutoCompleteTextView - đề xuất văn bản

Các bước sau mô tả làm thế nào để thiết lập một AutoCompleteTextView cung cấp văn bản đề xuất từ một mảng, sử dụng ArrayAdapter:

1. Thêm AutoCompleteTextView vào layout của bạn. dưới đây là layout với chỉ duy nhất một trường văn bản:

```
<?xml version="1.0" encoding="utf-8"?>
<AutoCompleteTextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/autocomplete_country"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

2. Khai báo mảng chứa tất cả văn bản đề xuất. Ví dụ, đây là mảng chứa tên các nước khai báo trong file XLM (res/values/strings.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="countries_array">
        <item>Afghanistan</item>
```

```
        <item>Albania</item>
        <item>Algeria</item>
        <item>American Samoa</item>
        <item>Andorra</item>
        <item>Angola</item>
        <item>Anguilla</item>
        <item>Antarctica</item>
        ...
    </string-array>
</resources>
```

3. Trong Activity hoặc Fragment, sử dụng code dưới đây để xác định adapter cung cấp các gợi ý:

```
// Get a reference to the AutoCompleteTextView in the layout
AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.autocomplete_country);
// Get the string array
String[] countries = getResources().getStringArray(R.array.countries_array);
// Create the adapter and set it to the AutoCompleteTextView
ArrayAdapter<String> adapter =
        new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, countries);
textView.setAdapter(adapter);
```

4. Gán adapter cho AutoCompleteTextView bằng cách gọi setAdapter().

2.5Các sự kiện đầu vào (Input Events)

Trên Android, có nhiều hơn một cách để bắt các sự kiện từ sự tương tác của người dùng với ứng dụng của bạn. Khi xem xét các sự kiện trong giao diện người dùng của bạn, cách tiếp cận là bắt các sự kiện từ các đối tượng View cụ thể mà người dùng tương tác. Các lớp View cung cấp các phương tiện để làm làm việc này.

Trong các lớp View khác nhau mà bạn sẽ sử dụng để bố trí layout của bạn, bạn có thể nhận thấy một số lời gọi phương thức hữu ích cho các sự kiện UI. Những phương thức này được gọi bởi framework Android khi các hành động tương ứng xảy ra trên đối tượng đó. Ví dụ, khi một View (chẳng hạn như một Button) được chạm vào, phương thức **onTouchEvent()** được gọi trên đối tượng đó. Tuy nhiên, để làm điều này, bạn phải mở rộng các lớp và ghi đè lên các phương thức. Tuy nhiên, mở rộng tất cả đối tượng View để xử lý một sự kiện như vậy sẽ là không thực tế. Đây là lý do tại sao các lớp View cùng chứa một tập các giao tiếp lồng nhau với callbacks giúp bạn có thể định nghĩa dễ dàng hơn. Các giao tiếp

này, được gọi là bộ nghe sự kiện (event listeners), là cách để bạn bắt các tương tác người dùng với giao diện người dùng của bạn.

Trong khi bạn sẽ thường sử dụng các event listeners để lắng nghe tương tác người dùng để xây dựng một thành phần tùy chỉnh. Có lẽ bạn muốn mở rộng các lớp Button để làm một cái gì đó nhiều hơn như thế. Trong trường hợp này, bạn sẽ có thể để xác định hành vi của sự kiện mặc định cho lớp của bạn bằng cách sử dụng lớp event handlers.

2.5.1Bắt sự kiện (Event Listeners)

Một event listener là một giao tiếp (interfarce) trong lớp View có chứa một lời gọi phương thức duy nhất. Các phương thức này sẽ được gọi bởi các framework Android khi View mà listener đã khai báo được kích hoạt bởi người dùng với các mục trong giao diện người dùng.

Trong các giao tiếp event listener có những phương thức sau đây:

onClick()

Thuộc View.OnClickListener. Nó được gọi khi người dùng hoặc chạm vào item (khi ở chế độ cảm ứng), hoặc lựa chọn vào item với các phím điều hướng và nhấn nút “enter” phù hợp.

onLongClick()

Thuộc View.OnLongClickListener. Nó được gọi khi người dùng hoặc chạm và giữ item (khi ở chế độ cảm ứng), hoặc lựa chọn vào item với các phím điều hướng sau đó nhấn và giữ phím "enter”.

onFocusChange()

Thuộc View.OnFocusChangeListener. Nó được gọi khi người dùng điều hướng ra khỏi item, bằng cách sử dụng phím điều hướng.

onKey()

Thuộc View.OnKeyListener. Nó được gọi khi người dùng lựa chọn va nhấn lên item.

onTouch()

Thuộc View.OnTouchListener. Nó được gọi khi người dùng thực hiện một hành động xác định đủ điều kiện như là một sự kiện cảm ứng, bao gồm việc nhấn, thoát ra, hoặc bất kỳ cử chỉ chuyển động vẽ trên màn hình (bên trong phạm vi của item).

onCreateContextMenu()

Thuộc View.OnCreateContextMenuListener. Nó được gọi khi một menu ngữ cảnh (Context Menu) đang được xây dựng (là kết quả của một "long click"). Xem thêm thông tin về context menus trong hướng dẫn phát triển Menus.

Ví dụ dưới đây cho thấy làm thế nào để đăng ký một bộ bắt sự kiện khi nhấp chuột vào một Button.

```
// Create an anonymous implementation of OnClickListener
private OnClickListener mCorkyListener = new OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
    ...
    // Capture our button from layout
    Button button = (Button)findViewById(R.id.corky);
    // Register the onClick listener with the implementation above
    button.setOnClickListener(mCorkyListener);
    ...
}
```

Chúng ta cũng có thể tìm thấy cách thuận tiện hơn để bổ sung OnClickListener như một phần Activity. Ví dụ:

```
public class ExampleActivity extends Activity implements OnClickListener {
    protected void onCreate(Bundle savedValues) {
        ...
        Button button = (Button)findViewById(R.id.corky);
        button.setOnClickListener(this);
    }

    // Implement the OnClickListener callback
    public void onClick(View v) {
      // do something when the button is clicked
    }
    ...
}
```

Chú ý rằng lời gọi **onClick()** trong ví dụ trên không trả về giá trị, nhưng các phương thức của bộ nghe sự kiện khác phải trả lại một biến kiểu boolean. Lý do phụ thuộc vào sự kiện này. Đây là một vài lý do:

- onLongClick() - Trả về một giá trị kiểu boolean để cho biết bạn đã dùng sự kiện này và nó không cần thực hiện ("long click") thêm nữa. Trả về giá trị *TRUE* để chỉ ra rằng bạn đã xử lý sự kiện này và nó nên dừng lại ở đây; trả về *FALSE* nếu bạn không xử lý nó và / hoặc sự kiện nên chuyển tới bất kỳ bộ nghe sự kiện **on-click** nào khác.
- onKey() - Trả về một giá trị kiểu boolean để cho biết bạn đã dùng sự kiện này và nó không cần được thực hiện thêm. Trả về giá trị *TRUE* để chỉ ra rằng bạn đã xử lý sự kiện này và nó nên dừng lại ở đây; trả về *FALSE* nếu bạn không xử lý nó và / hoặc sự kiện nên chuyển tới bất kỳ bộ nghe sự kiện **on-key** nào khác.
- onTouch() - Trả về một giá trị kiểu boolean để cho biết: liệu bộ nghe của bạn đã dùng sự kiện này hay chưa. Điều quan trọng là sự kiện này có thể có nhiều hành động nối tiếp nhau. Vì vậy, nếu trả về *FALSE*, bạn biết rằng bạn đã không sử dụng và cũng không quan tâm đến hành động tiếp theo từ sự kiện này. Như vậy, bạn không được gọi tới bất kỳ thao tác nào khác bên trong sự kiện này.

2.5.2Xử lý sự kiện (Event Handlers)

Nếu bạn đang xây dựng một thành phần tùy chỉnh từ View, bạn sẽ phải định nghĩa một số phương thức sử dụng như của xử lý sự kiện mặc định. Trong tài liệu về Custom Components, bạn sẽ tìm hiểu một số callbacks thường được sử dụng để xử lý sự kiện, bao gồm:

- onKeyDown(int, KeyEvent) – Được gọi khi một sự kiện nhấn phìm mới xảy ra.
- onKeyUp(int, KeyEvent) – Được gọi khi mọt sự kiện thả phím xảy ra.
- onTrackballEvent(MotionEvent) – Được gọi khi một sự kiện chuyển động trackball xảy ra.
- onTouchEvent(MotionEvent) – Được gọi khi một sự kiện chuyển động màn hình cảm ứng xảy ra.
- onFocusChanged(boolean, int, Rect) – Được gọi khi view được chọn (focus) hoặc bỏ chọn.

Có một số phương thức khác mà bạn nên biết, chúng không phải là một phần của lớp View, nhưng có thể trực tiếp tác động đến cách bạn có thể xử lý các sự kiện. Vì vậy, khi quản lý sự kiện phức tạp hơn bên trong một layout, bạn nên xem xét các phương pháp sau:

- Activity.dispatchTouchEvent(MotionEvent) – Điều này cho phép Activity bắt tất cả các sự kiện chạm màn hình trước khi chúng được gửi đến cửa sổ.
- ViewGroup.onInterceptTouchEvent(MotionEvent) – Điều này cho phép ViewGroup xem các sự kiện như chúng được gửi đến các View con.
- ViewParent.requestDisallowInterceptTouchEvent(boolean) – Gọi điều này trên View cha để xác định rằng nó không nên bắt các sự kiện chạm màn hình với onInterceptTouchEvent(MotionEvent).

2.5.3Chế độ cảm ứng (Touch Mode)

Khi người dùng điều hướng giao diện chương trình với các phím định hướng hoặc một trackball, nó là cần thiết để cung cấp focus cho các item có thể thực hiện thao tác (như nút bấm), vì vậy người dùng có thể biết đang thao tác với cái gì. Nếu thiết bị có khả năng cảm ứng, và người sử dụng bắt đầu tương tác với giao diện bằng cách chạm vào nó, vì vậy nó không còn cần thiết để làm nổi bật item, hoặc focus vào một View cụ thể. Như vậy, có một chế độ tương tác tên là "chế độ cảm ứng."

Đối với một thiết bị có khả năng cảm ứng, một khi người dùng chạm vào màn hình, thiết bị sẽ vào chế độ cảm ứng. Từ thời điểm này trở đi, chỉ có các View mà isFocusableInTouchMode() mang giá trị *TRUE* sẽ được focus, chẳng hạn như các widget chỉnh sửa văn bản. Các View khác có thể chạm, như các nút bấm, sẽ không có focus khi chạm vào; chúng chỉ đơn giản là sẽ thực hiện xử lý sự kiện on-lick khi được nhấn.

Bất cứ lúc nào một người dùng ấn nút trên bàn phím hoặc cuộn với một trackball, thiết bị sẽ thoát khỏi chế độ cảm ứng, và tìm một view để focus. Bây giờ, người dùng có thể tiếp tục tương tác với giao diện chương trình mà không cần chạm vào màn hình.

Các nhà chế độ cảm ứng được duy trì trong suốt toàn bộ hệ thống (tất cả các cửa sổ và các hoạt động). Để truy vấn tình trạng hiện nay, bạn có thể gọi isInTouchMode() để xem liệu các thiết bị đang ở chế độ cảm ứng.

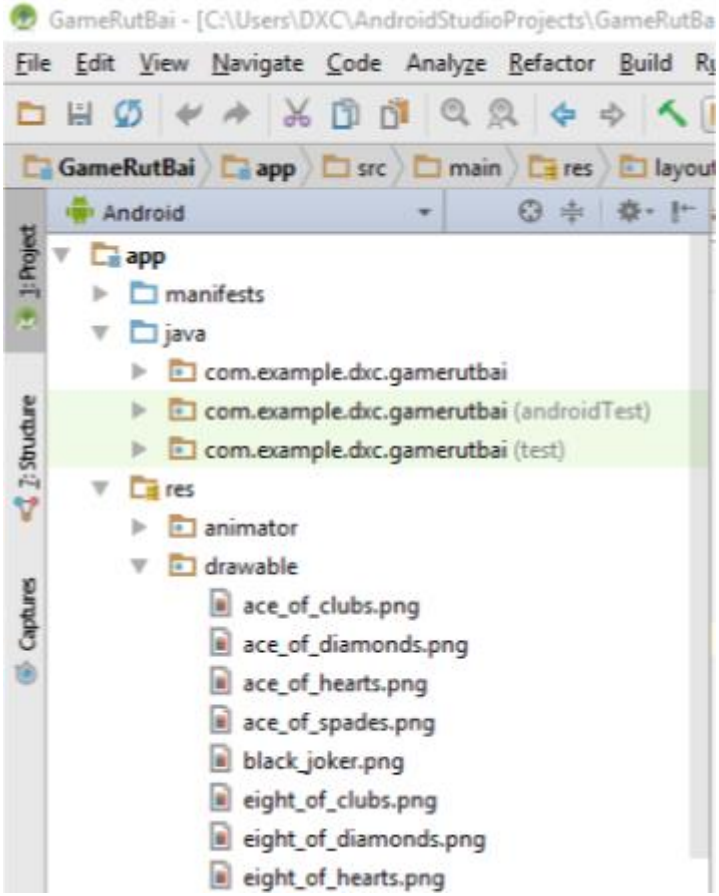
Chương 3 XÂY DỰNG CHƯƠNG TRÌNH THỰC NGHIỆM

3.1Phát biểu bài toán

Những chiếc smart phone chạy hệ điều hành Android ngày càng trở nên không thể thiếu được với chúng ta. Chúng vừa là phương tiện liên lạc cho công việc, vừa là phương tiện học tập và giải trí. Chương trình game đoán lá bài mà người chơi đã chọn sẽ giúp bạn quên đi thời gian chờ đợi lên máy hay hay chờ bạn gái dài đằng đẵng với luật chơi rất đơn giản: đầu tiên người chơi ấn nút "Play", chương trình sẽ tự động rút ra ngẫu nhiên 9 quân bài. Người chơi dùng mắt, tập trung để lựa chọn và ghi nhớ 1 quân bài. Sau 5 giây, chương trình sẽ đoán quân bài người chơi đã chọn và xóa nó đi.

3.2Kỹ thuật lập trình Game Đoán Lá Bài

Trước khi tạo giao diện cho chương trình, chúng ta sẽ nhập ảnh của 54 lá bài vào trong thư mục **drawable** của project.



Hình 24 Nhập 54 lá bài vào drawable

3.2.1Tạo màn hình giao diện trò chơi

Giao diện chính của chương trình sử dụng RelativeLayout mặc định, khai báo trong file **activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.dxc.gamerutbai.MainActivity">

<!--
    Khai báo cho các View con được viết tại đây
-->

</RelativeLayout>
```

Chúng ta chia giao diện thành 2 khu vực chính:

1) Vị trí các quân bài sẽ xuất hiện

Giao diện dược thiết kế bằng **TableLayout**, trên mỗi **TableRow** sẽ chứa 3 **FrameLayout** tương ứng với 3 quân bài.

```
<TableRow
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="horizontal">

    <FrameLayout
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:id="@+id/frmView1">
    </FrameLayout>

    <FrameLayout
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:id="@+id/frmView2"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp">
    </FrameLayout>

    <FrameLayout
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:id="@+id/frmView3">
    </FrameLayout>

    </TableRow>

<TableRow
    <!--
```

```
    Khai báo tương tự như TableRow ở trên
  -->
</TableRow>

<TableRow
  <!--
    Khai báo tương tự như TableRow ở trên
  -->
</TableRow>
```

2) Vị trí các nút bấm

Các nút bấm được thiết kế có kích thước bằng nhau và nằm trên 1 hàng ngang

```
<LinearLayout
  android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:id="@+id/linearLayout1"
  android:layout_alignParentBottom="true"
  android:layout_alignParentLeft="true"
  android:layout_alignParentStart="true">

  <Button
    android:text="@string/btnExit"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginRight="10dp"
    android:layout_marginEnd="10dp"
    android:id="@+id/buttonExit"
    android:layout_weight="1" />

  <Button
    android:text="@string/buttonRutBai"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/buttonRutBai"
    android:layout_weight="1" />

  <Button
    android:text="@string/btnHelp"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/buttonHelp"
    android:layout_marginLeft="10dp"
    android:layout_marginStart="10dp"
    android:layout_weight="1" />
</LinearLayout>
```

3.2.2Tạo giao diện các quân bài

Quân bài được tạo bởi 2 layout tương ứng với mặt trước và sau.

fragment_card_back.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:src="@drawable/mandolin_back"
  android:scaleType="centerInside"
  android:adjustViewBounds="true"
```

```
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="10dp" />
```

fragment_card_front.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ace_of_hearts"
    android:scaleType="centerInside"
    android:adjustViewBounds="true"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="10dp"
    android:id="@+id/imageFrmView_CardFrt" />
```

Để hiển thị 9 lá bài lên tương ứng 9 vị trí trên Table mà chúng ta đã tạo sẵn, cần khai báo trong file **MainActivity.java** như sau.

```
public class MainActivity extends AppCompatActivity {

    private boolean mShowingBack = false;

    private CardFrontFragment cardFrontFragment1 = new CardFrontFragment();
    private CardFrontFragment cardFrontFragment2 = new CardFrontFragment();
    private CardFrontFragment cardFrontFragment3 = new CardFrontFragment();
    private CardFrontFragment cardFrontFragment4 = new CardFrontFragment();
    private CardFrontFragment cardFrontFragment5 = new CardFrontFragment();
    private CardFrontFragment cardFrontFragment6 = new CardFrontFragment();
    private CardFrontFragment cardFrontFragment7 = new CardFrontFragment();
    private CardFrontFragment cardFrontFragment8 = new CardFrontFragment();
    private CardFrontFragment cardFrontFragment9 = new CardFrontFragment();

    private CardBackFragment cardBackFragment1 = new CardBackFragment();
    private CardBackFragment cardBackFragment2 = new CardBackFragment();
    private CardBackFragment cardBackFragment3 = new CardBackFragment();
    private CardBackFragment cardBackFragment4 = new CardBackFragment();
    private CardBackFragment cardBackFragment5 = new CardBackFragment();
    private CardBackFragment cardBackFragment6 = new CardBackFragment();
    private CardBackFragment cardBackFragment7 = new CardBackFragment();
    private CardBackFragment cardBackFragment8 = new CardBackFragment();
    private CardBackFragment cardBackFragment9 = new CardBackFragment();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        getFragmentManager()
            .beginTransaction()
            .add(R.id.frmView1, cardBackFragment1)
            .add(R.id.frmView2, cardBackFragment2)
            .add(R.id.frmView3, cardBackFragment3)
            .add(R.id.frmView4, cardBackFragment4)
            .add(R.id.frmView5, cardBackFragment5)
            .add(R.id.frmView6, cardBackFragment6)
            .add(R.id.frmView7, cardBackFragment7)
            .add(R.id.frmView8, cardBackFragment8)
            .add(R.id.frmView9, cardBackFragment9)
            .commit();

    /**
     * fragment đại diện cho mặt trước của lá bài.
     */
```

```
    public static class CardFrontFragment extends Fragment {

        ImageView imageView;

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {

            View view = inflater.inflate(R.layout.fragment_card_front
                                                    ,container
                                                    ,false);

            String rank,suit,image;

            rank = getArguments().getString("rank");
            suit = getArguments().getString("suit");
            image = getArguments().getString("image");

            Card card = new Card(rank,suit,image,getActivity());

            imageView = (ImageView) view.findViewById(R.id.imageFrmView_CardFrt);
            imageView.setImageResource(card.getResID());

            return view;
        }
    }

    /**
     * fragmen đại diện cho mặt sau của lá bài.
     */
    public static class CardBackFragment extends Fragment {
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            return inflater.inflate(R.layout.fragment_card_back, container, false);
        }
    }
}
```

3.2.3Tạo và lưu trữ quân bài

Đầu tiên chúng ta cần tạo một lớp quân bài

Card.java

```
package com.example.dxc.gamerutbai;

import android.content.Context;

public class Card {
    private String rank;
    private String suit;
    private String image;

    private Context context;

    public Card(String rank, String suit, String image, Context context) {
        this.rank = rank;
        this.suit = suit;
        this.image = image;
        this.context = context;
    }

    public void setRank(String rank) {
        this.rank = rank;
```

```
    }

    public String getRank() {
        return rank;
    }

    public void setSuit(String suit) {
        this.suit = suit;
    }

    public String getSuit() {
        return suit;
    }

    public void setImage(String image) {
        this.image = image;
    }

    public String getImage() {
        return image;
    }

    public Integer getResID() {
        return context.getResources()
                                .getIdentifier(getImage()
                                                        ,"drawable"
                                                        ,context.getPackageName());
    }
}
```

Tiếp đến khai báo 54 quân bài vào trong chương trình

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    // Khai báo các biến ở đây

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

                // Khai báo bắt sự kiện on-click ở đây

    }


    // Hàm lấy 54 quân bài
    private List<Card> getListData() {
                // Tạo mảng chứa các quân bài
        List<Card> deck = new ArrayList<>();

        //Khởi tạo các quân bài
        Card aceClub = new Card("Ace","Club","ace_of_clubs",this);
        Card aceDiamond = new Card("Ace","Diamond","ace_of_diamonds",this);
        Card aceHeart = new Card("Ace","Heart","ace_of_hearts",this);
        Card aceSpade = new Card("Ace","Spade","ace_of_spades",this);

        /**
        * Thực hiện tương tự cho 50 lá bài còn lại
        */

        // Thêm các quân bài vào mảng deck
        deck.add(aceClub);
        deck.add(aceDiamond);
        deck.add(aceHeart);
```

```
        deck.add(aceSpade);

        /**
        * Thực hiện tương tự cho 48 lá bài còn lại
        */

        deck.add(blackJoker);
        deck.add(redJoker);

        return  deck;
    }
}
```

3.2.4Rút ngẫu nhiên 9 quân bài khác nhau

Ý tưởng là:

- Bước 1: chọn ngẫu nhiên 1 quân bài trên cả bộ 54 lá (trong mảng deckClone)
- Bước 2: chuyển quân bài đã chọn ngẫu nhiên ở bước 1 vào mảng có 9 phần tử (deckView) – mảng này dùng để hiện các quân bài sau khi người chơi ấn nút "Play"
- Bước 3: tìm trên deckClone quân bài đồng hạng và chất với quân bài vừa chuyển vào deckView ở bước 2 (ví dụ quân bài ở bước 2 là 9 rô, thì phải tìm được 9 cơ) để chuyển vào mảng deckPrepare – mảng này dùng để thay thế các quân bài sau 5s

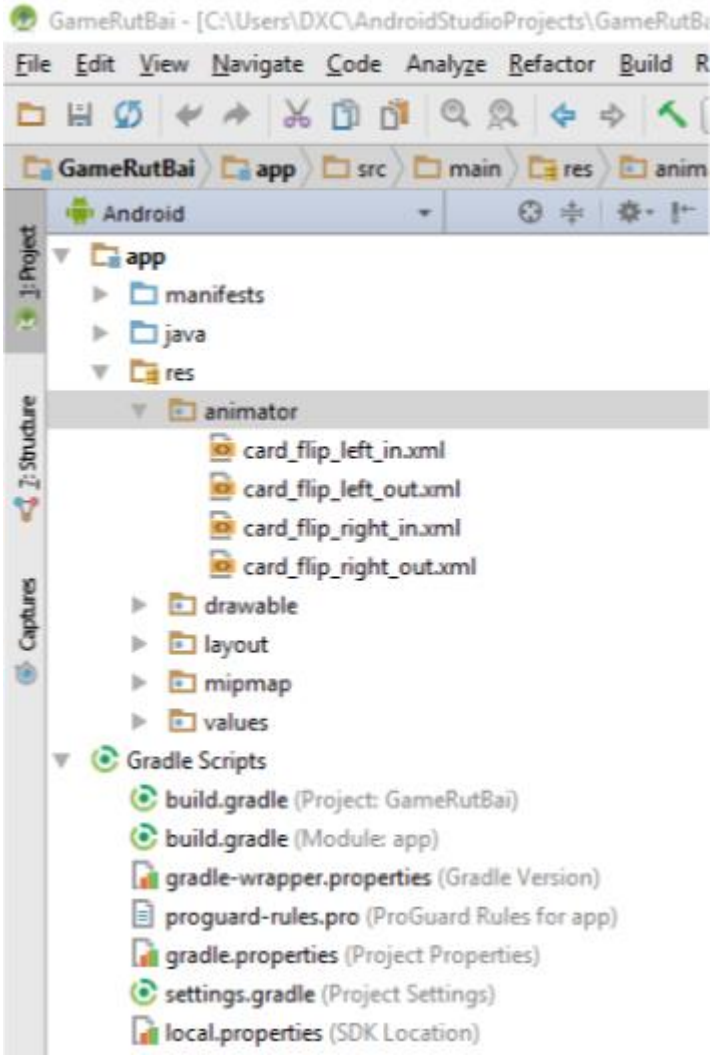
```
final List<Card> deckClone = getListData();
final List<Card> deckView = new ArrayList<>();
final List<Card> deckPrepare = new ArrayList<>();

Random r = new Random();
int batky;

for (int i=0;i<=8;i++) {
   batky = r.nextInt(deckClone.size());
   deckView.add(i, deckClone.get(batky));
   deckClone.remove(deckClone.get(batky));

   int j = 0;
   while (j <= deckClone.size()) {
      if (deckView.get(i).getRank().equals(deckClone.get(j).getRank())) {
         if (deckView.get(i).getSuit().equals("Club") && deckClone.get(j).getSuit().equals("Spade")) {
            deckPrepare.add(i, deckClone.get(j));
            deckClone.remove(deckClone.get(j));
            break;
         } else if (deckView.get(i).getSuit().equals("Spade") && deckClone.get(j).getSuit().equals("Club")) {
            deckPrepare.add(i, deckClone.get(j));
            deckClone.remove(deckClone.get(j));
            break;
         } else if (deckView.get(i).getSuit().equals("Diamond") && deckClone.get(j).getSuit().equals("Heart")) {
            deckPrepare.add(i, deckClone.get(j));
            deckClone.remove(deckClone.get(j));
            break;
         } else if (deckView.get(i).getSuit().equals("Heart") && deckClone.get(j).getSuit().equals("Diamond")) {
            deckPrepare.add(i, deckClone.get(j));
            deckClone.remove(deckClone.get(j));
            break;
         } else if (deckView.get(i).getSuit().equals("Black") &&  deckClone.get(j).getSuit().equals("Red")) {
```

```
            deckPrepare.add(i, deckClone.get(j));
            deckClone.remove(deckClone.get(j));
            break;
        } else if (deckView.get(i).getSuit().equals("Red") &&  deckClone.get(j).getSuit().equals("Black")) {
            deckPrepare.add(i, deckClone.get(j));
            deckClone.remove(deckClone.get(j));
            break;
        }
      }
      j++;
    }
}
```

3.2.5Kỹ thuật xoay úp và thay thế quân bài

- *Tạo hiệu ứng xoay quân bài*

Đầu tiên chúng ta cần tạo các file thực hiện hiệu ứng xoay bài trong thư mục **animator**.



Hình 25 Các file tạo hiệu ứng xoay bài

card_flip_left_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Before rotating, immediately set the alpha to 0. -->
    <objectAnimator
        android:valueFrom="1.0"
        android:valueTo="0.0"
        android:propertyName="alpha"
        android:duration="0" />

    <!-- Rotate. -->
    <objectAnimator
        android:valueFrom="-180"
        android:valueTo="0"
        android:propertyName="rotationY"
        android:interpolator="@android:interpolator/accelerate_decelerate"
        android:duration="1000" />

    <!-- Half-way through the rotation (see startOffset), set the alpha to 1. -->
    <objectAnimator
        android:valueFrom="0.0"
        android:valueTo="1.0"
        android:propertyName="alpha"
        android:startOffset="500"
        android:duration="1" />
</set>
```

card_flip_left_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Rotate. -->
    <objectAnimator
        android:valueFrom="0"
        android:valueTo="180"
        android:propertyName="rotationY"
        android:interpolator="@android:interpolator/accelerate_decelerate"
        android:duration="1000" />

    <!-- Half-way through the rotation (see startOffset), set the alpha to 0. -->
    <objectAnimator
        android:valueFrom="1.0"
        android:valueTo="0.0"
        android:propertyName="alpha"
        android:startOffset="500"
        android:duration="1" />
</set>
```

card_flip_right_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Before rotating, immediately set the alpha to 0. -->
    <objectAnimator
        android:valueFrom="1.0"
        android:valueTo="0.0"
        android:propertyName="alpha"
        android:duration="0" />

    <!-- Rotate. -->
    <objectAnimator
        android:valueFrom="180"
        android:valueTo="0"
```

```
    android:propertyName="rotationY"
    android:interpolator="@android:interpolator/accelerate_decelerate"
    android:duration="1000" />

  <!-- Half-way through the rotation (see startOffset), set the alpha to 1. -->
  <objectAnimator
    android:valueFrom="0.0"
    android:valueTo="1.0"
    android:propertyName="alpha"
    android:startOffset="500"
    android:duration="1" />
</set>
```

card_flip_right_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- Rotate. -->
  <objectAnimator
    android:valueFrom="0"
    android:valueTo="-180"
    android:propertyName="rotationY"
    android:interpolator="@android:interpolator/accelerate_decelerate"
    android:duration="1000" />

  <!-- Half-way through the rotation (see startOffset), set the alpha to 0. -->
  <objectAnimator
    android:valueFrom="1.0"
    android:valueTo="0.0"
    android:propertyName="alpha"
    android:startOffset="500"
    android:duration="1" />
</set>
```

Tạo hàm thực hiện việc xoay lá bài trong file **MainActivity.java**

```
private Handler mHandler = new Handler();

private void flipCard(Integer frmView, CardFrontFragment cardFrontFragment) {
  if (mShowingBack) {
    getFragmentManager().popBackStack();
    return;
  }

  mShowingBack = true;

  getFragmentManager()
      .beginTransaction()
      .setCustomAnimations(
          R.animator.card_flip_right_in, R.animator.card_flip_right_out,
          R.animator.card_flip_left_in, R.animator.card_flip_left_out)
      .replace(frmView, cardFrontFragment)
      .addToBackStack(null)
      .commit();

  mHandler.post(new Runnable() {
    @Override
    public void run() {
      invalidateOptionsMenu();
    }
  });
}
```

➢ *Thay thế quân bài*

Ý tưởng là:

- Ban đầu chương trình sẽ hiện 9 quân bài với mặt sau lên giao diện chính.
- Khi người dùng click vào nút "Play" (được khai báo trong code ở dưới là btnRutBai) chương trình sẽ rút ngẫu nhiên và chuẩn bị các quân bài như ở mục 3.2.4 đã nói ở trên.
- Trước khi thực hiện động tác lật bài, chương trình sẽ thay lá bài phù hợp vào cardFrontFragment.

```
btnRutBai.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        // Trong khi chương trình đang lật và úp bài
        // Không cho người dùng có thể ấn vào nút Play
        btnRutBai.setEnabled(false);

        final List<Card> deckClone = getListData();
        final List<Card> deckView = new ArrayList<>();
        final List<Card> deckPrepare = new ArrayList<>();

        Random r = new Random();
        int batky;

        for (int i=0;i<=8;i++) {
            batky = r.nextInt(deckClone.size());
            deckView.add(i, deckClone.get(batky));
            deckClone.remove(deckClone.get(batky));

            int j = 0;
            while (j <= deckClone.size()) {
                /**
                * code giống như mục 3.2.4
                */
        }

        Toast.makeText(getApplicationContext()
                                ,"Rút bài",Toast.LENGTH_SHORT)
                        .show();

        // Khởi tao biến để truyền thông tin lá bài vào fragment
        final Bundle bundle1 = new Bundle();
        bundle1.putString("rank", deckView.get(0).getRank());
        bundle1.putString("suit",deckView.get(0).getSuit());
        bundle1.putString("image",deckView.get(0).getImage());
        bundle1.putInt("resID",deckView.get(0).getResID());
        /**
        * Thực hiện tương tự cho 8 lá bài còn lại
        */

        // Truyền thông tin lá bài vào fragment tương ứng
        cardFrontFragment1.setArguments(bundle1);
        cardFrontFragment2.setArguments(bundle2);
        cardFrontFragment3.setArguments(bundle3);
        cardFrontFragment4.setArguments(bundle4);
        cardFrontFragment5.setArguments(bundle5);
        cardFrontFragment6.setArguments(bundle6);
        cardFrontFragment7.setArguments(bundle7);
        cardFrontFragment8.setArguments(bundle8);
```

```
cardFrontFragment9.setArguments(bundle9);

// Gọi hàm thực hiện thao tác xoay bài
flipCard(R.id.frmView1, cardFrontFragment1);

mShowingBack = false;
flipCard(R.id.frmView2, cardFrontFragment2);
/**
* Thực hiện tương tự cho 7 lá bài còn lại
*/


Runnable run = new Runnable() {
   @Override
   public void run() {

      flipCard(R.id.frmView1, cardFrontFragment1);
      flipCard(R.id.frmView2, cardFrontFragment2);
      flipCard(R.id.frmView3, cardFrontFragment3);
      flipCard(R.id.frmView4, cardFrontFragment4);
      flipCard(R.id.frmView5, cardFrontFragment5);
      flipCard(R.id.frmView6, cardFrontFragment6);
      flipCard(R.id.frmView7, cardFrontFragment7);
      flipCard(R.id.frmView8, cardFrontFragment8);
      flipCard(R.id.frmView9, cardFrontFragment9);

      mShowingBack = false;

      Toast.makeText(getApplicationContext()
                              ,"Chuẩn bị",Toast.LENGTH_SHORT)
                  .show();
   }
};

// Sau 3s sẽ úp bài
btnRutBai.postDelayed(run,3000);

Runnable run2 = new Runnable() {
   @Override
   public void run() {

      bundle1.putString("rank", deckPrepare.get(0).getRank());
      bundle1.putString("suit",deckPrepare.get(0).getSuit());
      bundle1.putString("image",deckPrepare.get(0).getImage());
      bundle1.putInt("resID",deckPrepare.get(0).getResID());
      /**
      * Thực hiện tương tự cho 8 lá bài còn lại
      */

      cardFrontFragment1.setArguments(bundle1);
      cardFrontFragment2.setArguments(bundle2);
      cardFrontFragment3.setArguments(bundle3);
      cardFrontFragment4.setArguments(bundle4);
      cardFrontFragment5.setArguments(bundle5);
      cardFrontFragment6.setArguments(bundle6);
      cardFrontFragment7.setArguments(bundle7);
      cardFrontFragment8.setArguments(bundle8);
      cardFrontFragment9.setArguments(bundle9);

      mShowingBack = false;
      flipCard(R.id.frmView1, cardFrontFragment1);

      /**
      * Thực hiện tương tự cho 8 lá bài còn lại
      */

      Toast.makeText(getApplicationContext()
                              ,"Lật bài",Toast.LENGTH_SHORT)
```

```
                                .show();
            }
        };

        // Sau 5s sẽ lật bài với các quân bài đã được thay thế
        btnRutBai.postDelayed(run2, 5000);

        Runnable run3 = new Runnable() {
            @Override
            public void run() {
                flipCard(R.id.frmView1, cardFrontFragment1);
                flipCard(R.id.frmView2, cardFrontFragment2);
                flipCard(R.id.frmView3, cardFrontFragment3);
                flipCard(R.id.frmView4, cardFrontFragment4);
                flipCard(R.id.frmView5, cardFrontFragment5);
                flipCard(R.id.frmView6, cardFrontFragment6);
                flipCard(R.id.frmView7, cardFrontFragment7);
                flipCard(R.id.frmView8, cardFrontFragment8);
                flipCard(R.id.frmView9, cardFrontFragment9);

                mShowingBack = false;

                Toast.makeText(getApplicationContext()
                                                ,"Kết thúc",Toast.LENGTH_SHORT)
                                .show();
            }
        };

        // Sau 8s sẽ úp lại bài và kết thúc trò chơi
        btnRutBai.postDelayed(run3, 8000);

        Runnable run4 = new Runnable() {
            @Override
            public void run() {
                btnRutBai.setEnabled(true);
            }
        };

        // Sau 10s sẽ bật lại nút Play để người dùng có thể chơi tiếp
        btnRutBai.postDelayed(run4, 10000);
    }
});
```

3.2.6Chuyển đổi qua lại giữa các màn hình Activity

```
// Sự kiến on-ckick để chuyển sang màn hình HelpActivity
btnHelp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent myIntent = new Intent(MainActivity.this,HelpActivity.class);
        MainActivity.this.startActivity(myIntent);
    }
});

// Đóng màn hình Activity hiện thời
btnExit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();
    }
});
```

3.2.7Kết quả chương trình

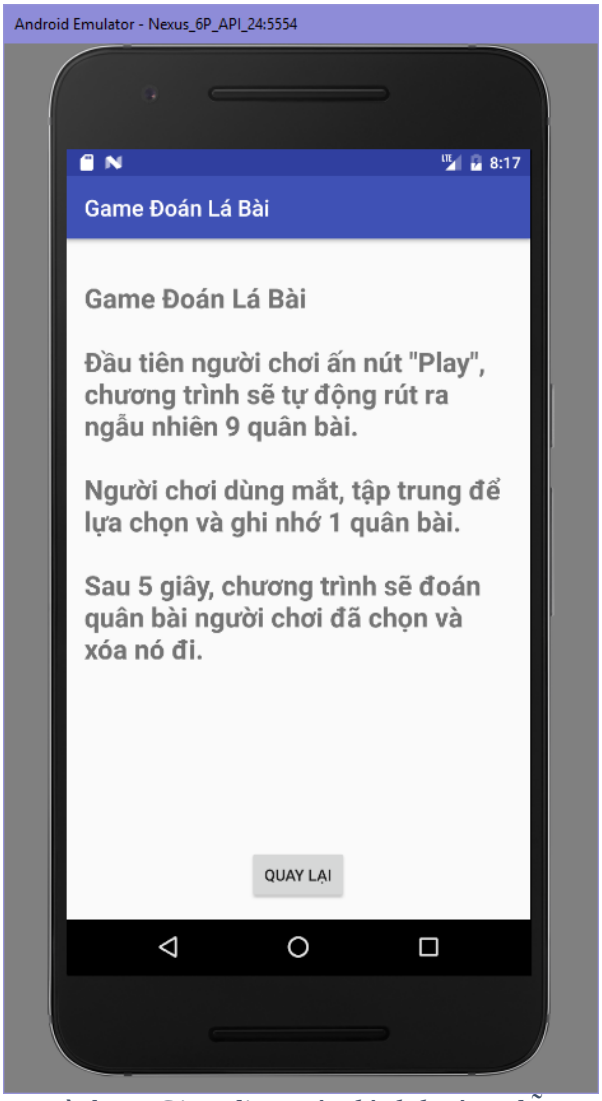
Sau khi thực hiện, được kết quả là một chương trình ứng dụng nhỏ: Game Đoán Lá Bài. Có thể tải về và cài đặt tại địa chỉ sau:
https://cuongdx1903.wordpress.com/

Một vài giao diện của chương trình khi chạy.

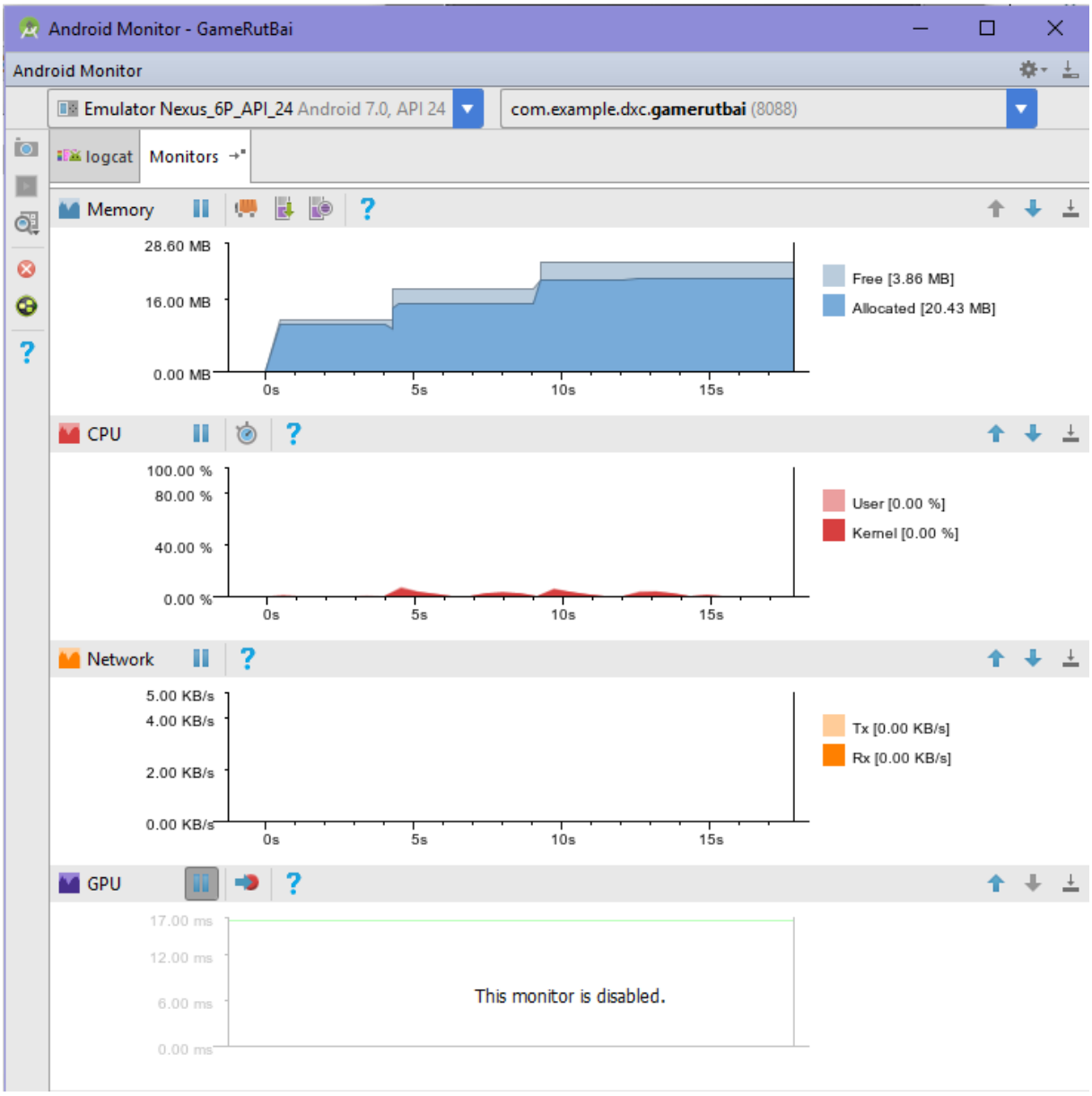
Hình 26 Giao diện màn hình chính

Hình 27 Giao diện khi bắt đầu chơi

Hình 28 Giao diện khi đang xoay bài



Hình 29 Giao diện màn hình hướng dẫn



Hình 30 Thông tin Memory, CPU, Network, GPU khi chạy ứng dụng

KẾT LUẬN

Trên đây là giới thiệu về mặt lý thuyết và thực nghiệm về với việc lập trình trên Android Studio. Luận văn hướng tới mục tiêu xây dựng một ứng dụng nhỏ, có khả năng chạy trên các thiết bị thật; qua đó tìm hiểu lập trình trên Android và có thể phát triển xây dựng ứng dụng khác – có tính thực tế cao. Vì thực hiện đề tài trong khoảng thời gian nhất định, nên còn nhiều vấn đề chưa hoàn chỉnh. Tuy nhiên, luận văn cũng đạt được một số kết quả:

- Về mặt lý thuyết: Tìm hiểu, nghiên cứu cách thức lập trình trên Android Studio. Tìm hiểu cách thức xây dựng một ứng dụng.
- Về thực nghiệm: Sử dụng kiến thức tìm hiểu được, xây dựng thành công một ứng dụng nhỏ: game đoán lá bài đã chọn.

Do thời gian có hạn nên luận văn mới chỉ nghiên cứu được một vài thành phần chính để tham gia xây dựng ứng dụng trên Android.

TÀI LIỆU THAM KHẢO

Tài liệu tham khảo trực tuyến:

[1]. [Online] https://developer.android.com/index.html

[2]. [Online] http://stackoverflow.com/

[3]. [Online] http://o7planning.org/vi/10407/bat-dau-voi-android-can-nhung-gi

[4]. [Online] https://vi.wikipedia.org/wiki/Android_(h%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh)

Tài nguyên:

[1]. Vector Playing Cards, tác giả Byron Knoll: http://byronknoll.blogspot.com/2011/03/vector-playing-cards.html

[2]. *Cards free icon* made by Freepik from http://www.flaticon.com/