|  |  |
| --- | --- |
| BỘ CÔNG THƯƠNG  **TRƯỜNG ĐẠI HỌC KINH TẾ**  **KỸ THUẬT CÔNG NGHIỆP** | **KHOA CÔNG NGHỆ THÔNG TIN** |



**BÁO CÁO BÀI TẬP LỚN**

**Xây dựng phần mềm chơi game Tetris**

**Giảng viên hướng dẫn: ThS. Ngô Quang Trí**

**Nhóm SV thực hiện: Đoàn Thị Hồng**

**Lê Tiến Đạt**

**Nguyễn Như Quỳnh**

**Nguyễn Thị Hà My**

**Nguyễn Trí Dũng**

**Khiếu Thị Kim Nguyệt**

**Hà Nôi, 2024**

**HÀ NỘI, NĂM 2024**

**MỤC LỤC**

[**DANH MỤC HÌNH ẢNH** 1](#_Toc172490742)

[**LỜI MỞ ĐẦU** 3](#_Toc172490743)

[**CHƯƠNG I: TỔNG QUAN ĐỀ TÀI** 4](#_Toc172490744)

[1.1. Mô tả về game 4](#_Toc172490745)

[1.2. Yêu cầu đối với sản phẩm 4](#_Toc172490746)

[1.2.1. Yêu cầu về chức năng 4](#_Toc172490747)

[1.2.2. Yêu cầu về giao diện 5](#_Toc172490748)

[**CHƯƠNG II: TỔNG QUAN VỀ ANDROID** 6](#_Toc172490749)

[2.1. Giới thiệu về Android 6](#_Toc172490750)

[2.2. Khái quát về Android 6](#_Toc172490751)

[2.3. Những tính năng nền tảng Android hỗ trợ 7](#_Toc172490752)

[2.4. Kiến trúc Android 7](#_Toc172490753)

[2.5. Activity 9](#_Toc172490754)

[2.5.1. Định nghĩa activity 9](#_Toc172490755)

[2.5.2. Vòng đời Activity 10](#_Toc172490756)

[**CHƯƠNG III: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG** 13](#_Toc172490757)

[3.1. Phân tích hệ thống 13](#_Toc172490758)

[3.1.1. Các tác nhân của hệ thống 13](#_Toc172490759)

[3.1.2. Các chức năng của hệ thống 13](#_Toc172490760)

[3.2. Thiết kế 15](#_Toc172490761)

[3.2.1. Biểu đồ use – case tổng quát 15](#_Toc172490762)

[3.2.2. Biểu đồ hoạt động (Activity Diagram) 17](#_Toc172490763)

[3.2.3. Biểu đồ trạng thái (State Diagram) 20](#_Toc172490764)

[3.2.4. Biểu đồ thành phần (Component Diagram) 20](#_Toc172490765)

[**CHƯƠNG IV: CÀI ĐẶT CHƯƠNG TRÌNH** 21](#_Toc172490766)

[4.1. Giao diện bắt đầu game 21](#_Toc172490767)

[4.2. Giao diện chọn chế độ 21](#_Toc172490768)

[4.3. Giao diện chơi game 22](#_Toc172490769)

[4.4. Giao diện Setting 22](#_Toc172490770)

[**KẾT LUẬN** 23](#_Toc172490771)

[**TÀI LIỆU THAM KHẢO** 24](#_Toc172490772)

# **DANH MỤC HÌNH ẢNH**

[Hình 1: Lược đồ thể hiện các thành phần của hệ điều hành Android 7](#_Toc172490493)

[Hình 2: Lược đồ vòng đời của một Activity 10](#_Toc172490494)

[Hình 3: Biểu đồ user – case tổng quát 16](#_Toc172490495)

[Hình 4: Biểu đồ hoạt động bắt đầu chơi 18](#_Toc172490496)

[Hình 5: Biểu đồ hoạt động chế độ Marathon 19](#_Toc172490497)

[Hình 6: Biểu đồ hoạt động chế độ Sprin 20](#_Toc172490498)

[Hình 7: Biểu đồ trạng thái 21](#_Toc172490499)

[Hình 8: Biểu đồ thành phần 21](#_Toc172490500)

# **LỜI MỞ ĐẦU**

Trong bối cảnh sự phát triển nhanh chóng của các thiết bị di động, sự cạnh tranh về hệ điều hành cho các thiết bị này cũng trở nên vô cùng khốc liệt. Trên thị trường hiện nay có nhiều hệ điều hành khác nhau như iOS, BlackBerry, Symbian, Palm, và Windows Phone 7. Tuy nhiên, Android lại được đặc biệt yêu thích bởi nhiều lý do. Android nổi bật vì tính thân thiện với người dùng, chi phí hợp lý, khả năng tùy biến cao, và kho ứng dụng phong phú. Đặc biệt, ưu thế lớn nhất của Android chính là mã nguồn mở.

Android đã thu hút rất nhiều nhà phát triển từ khắp nơi trên thế giới khi Google phân phối miễn phí Android và các đoạn mã có bản quyền cho bất kỳ ai muốn tham gia phát triển phần mềm cho nền tảng này. Nhận thấy nhu cầu lớn về ứng dụng từ người dùng cuối, cùng với hệ thống phân phối ứng dụng thuận tiện trên Google Play Store, chúng em đã quyết định chọn đề tài “Xây dựng một phần mềm chơi game Tetris” trên Android.

Mục tiêu của chúng em là thông qua đề tài:

* **Học kiến thức mới và thử sức với Android**: Qua việc phát triển game Tetris bằng Android Studio, chủng em mong muốn nắm vững các kiến thức về lập trình Android đã học từ môn Công nghệ lập trình đa nền tảng cho ứng dụng di động, từ đó tự tin hơn trong việc phát triển ứng dụng di động.
* **Nâng cao khả năng làm việc nhóm**: Quá trình tìm hiểu và nghiên cứu và hợp tác sẽ giúp các thành viên rèn luyện và cải thiện khả năng làm việc nhóm, hợp tác hiệu quả với nhau để đạt được mục tiêu chung.
* **Tạo nền tảng cho các ứng dụng sau này**: Việc phát triển game Tetris không chỉ dừng lại ở một sản phẩm hoàn chỉnh mà còn là bước đệm quan trọng, tạo nền tảng vững chắc cho những ứng dụng sẽ phát triển trong tương lai.

Với điều kiện thời gian cũng như kinh nghiệm kiến thức còn hạn chế bài tập lớn này không tránh được thiếu sót. Chúng em rất mong nhận được sự chỉ bảo đóng góp ý kiến của các thầy cô để đồ án của chúng em có thể hoàn thiện hơn

Chúng em xin chân thành cảm ơn!

*Hà Nội, ngày… tháng … năm 2024*

**Nhóm sinh viên**

# **CHƯƠNG I: TỔNG QUAN ĐỀ TÀI**

## **1.1. Mô tả về game**

Tetris là một trong những trò chơi điện tử kinh điển nổi tiếng trên toàn thế giới. Trò chơi được tạo ra vào những năm 1980 bởi nhà phát triển game người Nga Alexey Pajitnov và đã nhanh chóng trở thành một hiện tượng với sự phổ biến lớn.

**Mục đích của trò chơi:** Mục đích của Tetris đơn giản nhưng gây nghiện: xếp các khối hình có kích thước và hình dạng khác nhau thành các hàng hoặc cột đầy đủ. Khi một hàng hoặc cột được xếp đầy, nó sẽ biến mất và tạo ra không gian trống cho những khối hình tiếp theo.

**Cách chơi:**

* Người chơi bắt đầu với một không gian trống và các khối hình (gọi là Tetrimino) rơi từ trên xuống. Các Tetrimino có thể xoay và di chuyển sang trái, phải hoặc rơi xuống dưới cùng của không gian chơi.
* Nhiệm vụ của người chơi là xoay và di chuyển các Tetrimino để xếp chúng sao cho tạo thành các hàng hoặc cột hoàn chỉnh.
* Khi một hàng hoặc cột được xếp đầy, nó sẽ biến mất, và những khối hình phía trên nó sẽ rơi xuống để lấp đầy khoảng trống vừa tạo ra.
* Trò chơi kết thúc khi không còn không gian để các Tetrimino rơi xuống.

**Đặc điểm nổi bật:**

* **Tốc độ tăng dần:** Khi người chơi tiến bộ và đạt được điểm số cao, tốc độ rơi của các Tetrimino sẽ tăng, tạo ra thử thách ngày càng lớn.
* **Điểm số:** Người chơi được tính điểm dựa trên số hàng hoặc cột mà họ đã xóa. Càng nhiều hàng/cột được xóa cùng một lúc, điểm số càng cao.
* **Đồ họa đơn giản nhưng cuốn hút:** Tetris có đồ họa đơn giản nhưng sắc nét, với các khối hình màu sắc tươi sáng và nền đen trắng làm nổi bật chúng.

Tetris không chỉ là một trò chơi giải trí đơn thuần mà còn là một thử thách trí tuệ và phản xạ nhanh. Với sự phổ biến lâu dài và ảnh hưởng sâu rộng, Tetris đã trở thành một biểu tượng trong thế giới game điện tử.

## **1.2. Yêu cầu đối với sản phẩm**

### **1.2.1. Yêu cầu về chức năng**

* **Play (Chơi)**
  + Bắt đầu trò chơi mới.
  + Tạm dừng và tiếp tục trò chơi.
  + Hiển thị điểm số hiện tại..
* **Setting (Cài đặt)**
  + Điều chỉnh âm thanh ( bật/tắt âm thanh).
  + Điều chỉnh tốc độ rơi của các khối.
  + Chọn chế độ.

### **1.2.2. Yêu cầu về giao diện**

* **Giao diện Chờ**
  + Hiển thị logo và tên trò chơi.
  + Các nút chức năng chính: Play, Setting,
* **Giao diện Setting**
  + Các tuỳ chọn điều chỉnh âm thanh.
  + Các tuỳ chọn điều chỉnh tốc độ rơi.
  + Lựa chọn chế độ điều khiển.
  + Nút quay lại giao diện chờ.
* **Giao diện Chơi Game**
  + Khu vực hiển thị lưới trò chơi.
  + Khối Tetris rơi và khối đã sắp xếp.
  + Hiển thị điểm số hiện tại.
  + Nút tạm dừng.
  + Hiển thị khối tiếp theo.
  + Hiển thị thông báo khi đạt kỷ lục mới hoặc kết thúc trò chơi.
* **Giao diện Chọn Chế độ Chơi**: Các chế độ chơi khác nhau: Marathon, Sprint.

# **CHƯƠNG II: TỔNG QUAN VỀ ANDROID**

## **2.1. Giới thiệu về Android**

Android đã ra đời từ những năm 2000, nhưng chỉ được biết đến nhiều từ năm 2005, khi Google mua lại Android INC.Chính sự nhạy bén trong việc nắm bắt xu thế mới của Google, đã thúc đẩy ngành công nghiệp lập trình di động lên 1 tầm cao mới, trở thành 1 trào lưu mạnh mẽ và trên đà phát triển hơn nữa. Vậy thì có những nguyên nhân nào tạo cho Android bước phát triển thần kỳ này:

Đầu tiên, phải kể đến Android là một mã nguồn mở. Android INC 2001 mới chỉ có chục nhân viên. Nhưng Google với triết lý kinh doanh của mình là "mở" đã đưa Android trở thành hệ điều hành di động phổ biến nhất thế giới. Đây chính là điều kiện tiên quyết. Các nhà phát triển phần cứng chỉ cần trả 1 chi phí rất nhỏ, là họ có thể cài đặt và tùy chọn hệ điều hành này trên điện thoại của mình. Và nhanh chóng, 1 liên minh phần cứng được thành lập với những hảng sản xuất hàng đầu thế giới: Samsung, LG, Motorola, SonyErricsson.

Các nhà lập trình phần mềm thì sao?

Quá dễ dàng cho họ bởi vì chỉ cần nắm lập trình Java thì ta có thể dễ dàng tiếp cận với Android gần như là ngay lập tức, sau khi học thêm 1 số khái niệm trong Android là đã có thể xây dựng được các ứng dụng Android cần thiết. Tại sao lập trình Android lại dễ?

Chỉ cần tải và cài đặt bộ công cụ Android SDK(cho Windows) thì có thể xây dựng ngay 1 ứng dụng này. Thậm chí khi đang phân vân là mình có nên mua 1 thiết bị Android để thử ứng dụng đó hay không? Thì trong bộ SDK đã có sẵn trình giả lập cho bạn dùng rồi.

## **2.2. Khái quát về Android**

Android là hệ điều hành điện thoại di động phát triển dựa trên nền tảng của Linux. Hệ điều hành Android bao gồm 12.000.000 dòng mã trong đó có 3.000.000 dòng XML, 2.800.000 dòng C, 2.100.000 dòng Java, và 1.750.000 dòng C++.

Bất kỳ một hãng sản xuất phần cứng nào cũng đều có thể tự do sử dụng hệ điều hành Android cho thiết bị của mình, miễn là các thiết bị ấy đáp ứng được các tiêu chuẩn cơ bản do Google đặt ra (có cảm ứng chạm, GPS, 3G,...).

Các nhà sản xuất có thể tự do thay đổi phiên bản Android trên máy của mình mà không cần phải xin phép hay trả bất kì khoản phí nào nhưng phải đảm bảo tính tương thích ngược (backward compatibility).

## **2.3. Những tính năng nền tảng Android hỗ trợ**

Android là nền tảng cho thiết bị di động bao gồm một hệ điều hành, midware và một số ứng dụng chủ đạo. Bộ công cụ Android SDK cung cấp các công cụ và bộ thư viện các hàm API cần thiết để phát triển ứng dụng cho nền tảng Android.

* Application framework: cho phép việc tái sử dụng và thay thế mã nguồn ở dạng component một cách dễ dàng.
* Dalvik virtual macine: máy ảo Java được tối ưu hóa cho thiết bị di động.
* Intergrated browser: trình duyệt web tích hợp được xây dựng dựa trên [WebKit](http://webkit.org/) engine.
* Optimized graphics: các tính năng đồ họa được tối ưu bởi một thư viện đồ họa 2D bên dưới; đối với đồ họa 3D, Android sử dụng thư viện OpenGL ES 1.0 nếu thiết bị có hỗ trợ.
* SQLite: DBMS dùng để lưu trữ dữ liệu có cấu trúc.
* Hỗ trợ các định dạng hình ảnh, âm thanh, video phổ biến như MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF.
* Hỗ trợ băng tầng GSM (tùy vào phần cứng thiết bị)
* Hỗ trợ Bluetooth, EDGE, 3G, WiFi (tùy vào phần cứng thiết bị).
* Ngoài ra còn có khả năng của các thiết bị như máy chụp ảnh, thiết bị định vị toàn cầu, la bàn, và bộ cảm biến gia tốc.

Cung cấp môi trường phát triển phần mềm đầy đủ, bao gồm phần mềm giả lập thiết bị, các công cụ gỡ rối (debugging), theo dõi bộ nhớ và năng suất hoạt động, plugin cho Eclipse IDE.

## **2.4. Kiến trúc Android**

A screenshot of a computer program

Description automatically generated

Hình 1: Lược đồ thể hiện các thành phần của hệ điều hành Android

* Tầng Applications:

Hệ điều hành Android tích hợp sẵn một số ứng dụng cơ bản như email client, SMS, lịch điện tử, bản đồ, trình duyệt web, sổ liên lạc và một số ứng dụng khác. Ngoài ra tầng này cũng chính là tầng chứa các ứng dụng được phát triển bằng ngôn ngữ Java.

* Tầng Application Framework:

Ở tầng này, các nhà phát triển ứng dụng có thể truy cập vào phần cứng thiết bị, thông tin định vị vị trí, chạy các dịch vụ nền, đặt các cảnh báo, thông báo vào thanh trạng thái, v.v…. và quan trọng nhất, đó là các API của framework.

Một vài thành phần chính của tầng Application Framework:

* Activity Manager: quản lý vòng đời của tất cả các ứng dụng.
* Window Manager: quản lý những cửa sổ khác nhau.
* Content Providers: cho phép các ứng dụng có thể truy xuất dữ liệu từ các ứng dụng khác hoặc chia sẽ dữ liệu của chúng.
* View System: Chứa các block xây dựng của Android UI.
* Packages Manager: theo dõi các ứng dụng được cài đặt trên thiết bị.
* Telephony Manager: chứa các API . Người phát triển ứng dụng được phép sử dụng đầy đủ bộ API được dùng trong các ứng dụng tích hợp sẵn của Android. Kiến trúc ứng dụng của Android được thiết kế nhằm mục đích đơn giản hóa việc tái sử dụng các component. Qua đó bất kì ứng dụng nào cũng có thể công bố các tính năng mà nó muốn chia sẻ cho các ứng dụng khác (VD: Ứng dụng email có muốn các ứng dụng khác có thể sử dụng tính năng gởi mail của nó). Phương pháp tương tự cho phép các thành phần có thể được thay thế bởi người sử dụng.
* Resources Manager: cung cấp khả năng truy xuất các tài nguyên non-code như hình ảnh hoặc file layout.
* Location Manager: quản lý vị trí, sử dụng GPS hoặc cell tower.
* Notification Manager: cung cấp khả năng hiển thị custom alert trên thanh status bar.
* Tầng Libraries – Runtime

Libraries: chạy trên đỉnh của nhân chứa tất cả code cung cấp những tính năng chính của Android OS. Một vài library hữu ích:

* Surface Manager: quản lý hiển thị nội dung 2D và 3D.
* Media Framework: được cung cấp bởi PacketVideo. Là bộ thư viện hỗ trợ trình diễn và ghi các định dạng âm thanh và hình ảnh phổ biến.
* SQLite: một DBMS nhỏ gọn và mạnh mẽ.
* Graphics Libraries (Open GL|ES, SGL): Open GL|ES, SGL là hai phần lõi của những thư viện đồ họa. Open GL|ES là thư viện đồ họa 3D, SGL là thư viện đồ họa 2D.
* FreeType: được sử dụng để tạo phông chữ.
* WebKit: engine trình duyệt mã nguồn mở.
* SSL: được sử dụng cho bảo mật mạng.
* System C library: được xây dựng từ BSD của bộ thư viện hệ thống C chuẩn (libc), được điều chỉnh để tối ưu hóa cho các thiết bị chạy trên nền Linux.

Runtime: hệ điều hành Android tích hợp sẵn một tập hợp các thư viện cốt lõi cung cấp hầu hết các chức năng có sẵn trong các thư viện lõi của ngôn ngữ lập trình Java. Mọi ứng dụng của Android chạy trên một tiến trình của riêng nó cùng với một thể hiện của máy ảo Dalvik. Máy ảo Dalvik thực tế là một biến thể của máy ảo Java được sửa đổi, bổ sung các công nghệ đặc trưng của thiết bị di động. Nó được xây dựng với mục đích làm cho các thiết bị di động có thể chạy nhiều máy ảo một cách hiệu quả. Trước khi thực thi, bất kì ứng dụng Android nào cũng được convert thành file thực thi với định dạng nén Dalvik Executable (.dex). Định dạng này được thiết kế để phù hợp với các thiết bị hạn chế về bộ nhớ cũng như tốc độ xử lý. Ngoài ra máy ảo Dalvik sử dụng bộ nhân Linux để cung cấp các tính năng như thread, quản lý bộ nhớ mức thấp.

* Tầng Linux Kernel:

Hệ điều hành Android được xây dựng trên bộ nhân Linux 2.6 cho những dịch vụ hệ thống cốt lõi như: security, memory management, process management, network stack, driver model. Bộ nhân này làm nhiệm vụ như một lớp trung gian kết nối phần cứng thiết bị và phần ứng dụng.

## **2.5. Activity**

### **2.5.1. Định nghĩa activity**

Actitvity là thành phần quan trọng nhất và đóng vai trò chính trong xây dựng ứng dụng Android. Do gần như mọi activity đều tương tác với người dùng, lớp Activity đảm nhận việc tạo ra một cửa sổ (window) để người lập trình đặt lên đó một giao diện UI với setContentView(View). Một activity có thể mang nhiều dạng khác nhau: Một cửa sổ toàn màn hình (full screen window), một cửa sổ floating (với windowsIsFloating) hay nằm lồng bên trong 1 activity khác (với ActivityGroup).

Có 2 phương thức mà gần như mọi lớp con của Activity đều phải thực hiện:

* **onCreate(Bundle)** - Nơi khởi tạo activity. Quan trọng hơn, đây chính người lập trình gọi setContentView(int) kèm theo layout để thể hiện UI của riêng mình. Đồng thời còn có findViewById(int) giúp gọi các widget (buttons, text boxes, labels,..) để dùng trong UI.
* **onPause()** - Nơi giải quyết sự kiện người dùng rời khỏi activity. Mọi dữ liệu được người dùng tạo ra tới thời điểm này cần phải được lưu vào ContentProvider.
* Để có thể sử dụng Context.startActivity(), mọi lớp activity đều phải được khai báo với tag <activity> trong file AndroidManifest.xml.

<activity

android:name=\*"game.modules.tetris.Logo"\*

android:label=\*"@string/app\_name"\* >

<intent-filter>

<action android:name=\*"android.intent.action.MAIN"\* />

<category android:name=\*"android.intent.category.LAUNCHER"\*/>

</intent-filter>

</activity>

### **2.5.2. Vòng đời Activity**

Hệ điều hành Android quản lý Activity theo dạng stack: khi một Activity mới được khởi tạo, nó sẽ được xếp lên đầu của stack và trở thành ***running activity***, các Activity trước đó sẽ bị tạm dừng và chỉ hoạt động trở lại khi Activity mới được giải phóng.

Activity bao gồm 4 trạng thái:

* + - ***active (running):*** Activity đang hiển thị trên màn hình (foreground).
    - ***paused:*** Activity vẫn hiển thị (visible) nhưng không thể tương tác (lost focus). VD: một activity mới xuất hiện hiển thị giao diện đè lên trên activity cũ, nhưng giao diện này nhỏ hơn giao diện của activity cũ, do đó ta vẫn thấy được 1 phần giao diện của activity cũ nhưng lại không thể tương tác với nó.
    - ***stop:*** Activity bị thay thế hoàn toàn bởi Activity mới sẽ tiến đến trạng thái ***stop`***
    - ***killed:*** Khi hệ thống bị thiếu bộ nhớ, nó sẽ giải phóng các tiến trình theo nguyên tắc ưu tiên. Các Activity ở trạng thái ***stop*** hoặc ***paused***cũng có thể bị giải phóng và khi nó được hiển thị lại thì các Activity này phải khởi động lại hoàn toàn và phục hồi lại trạng thái trước đó.



Hình 2: Lược đồ vòng đời của một Activity

Dựa vào lược đồ trên ta thấy có 3 vòng lặp quan trọng:

* ***Entire lifetime:*** Từ phương thức **onCreate( )** cho tới **onDestroy( )**
* ***Visible liftetime:*** Từ phương thức **onStart( )** cho tới **onStop( )**
* ***Foreground lifetime:*** Từ phương thức **onResume( )** cho tới **onPause( )**

Toàn bộ vòng đời của 1 activity được định nghĩa nhờ các phương thức sau:

public class DemoActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// Gọi khi mới tạo activity để setup các view, binding dữ liệu,...

// Kèm theo sau luôn là onStart().

}

@Override

protected void onRestart() {

super.onRestart();

// Gọi sau khi activity bị stopped và trước khi được khởi động lại.

// Kèm theo sau luôn là onStart().

}

@Override

protected void onStart() {

super.onStart();

// Gọi khi activity hiện lên trước mắt người dùng.

// Kèm theo sau là onResume() nếu activity hiện lên nền hay onStop() nếu bị ẩn đi.

}

@Override

protected void onResume() {

super.onResume();

// Gọi khi activity bắt đầu tương tác với người dùng và đang trên

// cùng của activity stack.

// Kèm theo sau luôn là onPause().

}

@Override

protected void onPause() {

super.onPause();

// Gọi khi hệ thống sắp khởi động lại 1 activity khác trước đó.

// kèm theo sau là onResume() nếu activity trở lại trên cùng hay

// onStop() nếu bị ẩn đi.

}

@Override

protected void onStop() {

super.onStop();

// Gọi khi activity không còn hiển thị trước người dùng

// Kèm theo sau là onRestart() nếu activity hiện lên trở lại hay onDestroy()

// nếu sắp xoá activity đi.

}

@Override

protected void onDestroy() {

super.onDestroy();

// Gọi ngay trước khi kết thúc activity, khi hàm finish() được gọi

// hoặc khi hệ thống yêu cầu buộc phải kết thúc.

}

}

# **CHƯƠNG III: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**

## **3.1. Phân tích hệ thống**

### **3.1.1. Các tác nhân của hệ thống**

|  |  |
| --- | --- |
| A black stick figure with text  Description automatically generated | Người dùng truy cập vào game, thực hiện các thao tác trong game như: Play(Bắt đầu chơi), Xem hướng dẫn, tắt bật âm thanh, di chuyển các khối, xoay các khối,Pause(Tạm dừng), Resume(Tiếp tục), Quit (Thoát)  Người dùng được hỗ trợ chơi game offline |

### **3.1.2. Các chức năng của hệ thống**

* Quản lý khung game

Giao diện game được chia thành từng ô vuông nhỏ. Khung game được gán cho kích thước 26 hàng x 16 cột. Để lưu gạch hiện trên khung game ta sẽ dùng mảng 2 chiều để lưu giá trị màu của gạch.

Nhưng phải lưu ý trước khi gạch hiện ra thì gạch phải ở trong hàng đợi. Mà gạch chiếm nhiều nhất là 4 hàng x 4 cột. Vậy hàng đợi phải cao 4 hàng. Vậy toàn mảng để lưu lại vị trí gạch rơi phải là mảng (26 + 4) hàng x 16 cột.

* Quản lý điều khiển gạch
* Gạch rơi hoặc đi xuống
* Kiểm tra vị trí mới:

+ Tọa độ hiện tại:Giả sử tọa độ của khối hiện tại là (x,y).

+ Tọa độ mới:Khi di chuyển xuống, tọa độ mới sẽ là (x,y+1).

* Kiểm tra va trạm

+ Mảng khung game: Đây là mảng lưu trạng thái hiện tại của trò chơi, trong đó mỗi ô có thể có một màu sắc hoặc trống (thường được biểu diễn bằng màu đen hoặc giá trị 0).

+ Mảng của khối: Đây là mảng mô tả hình dạng và kích thước của khối hiện tại.

+ Kiểm tra từng ô của khối: Bạn sẽ kiểm tra từng ô của khối để xác định xem khi di chuyển sang phải một ô, có bất kỳ ô nào của khối chạm vào một ô không trống trong mảng khung game hay không.

* Xử lý va chạm

+ Nếu có va chạm: Nếu bất kỳ ô nào của khối chạm vào một ô không trống trong mảng khung game, không cho phép khối rơi xuống.

+ Nếu không có va chạm: Nếu tất cả các ô của khối đều không chạm vào ô không trống trong mảng khung game, cho phép khối rơi xuống một ô.

* Sang trái
* Kiểm tra vị trí mới

+ Tọa độ hiện tại:Giả sử tọa độ của khối hiện tại là(x,y).

+ Tọa độ mới:Khi di chuyển sang phải, tọa độ mới sẽ là (x-1,y).

* Kiểm tra va trạm

+ Mảng khung game: Đây là mảng lưu trạng thái hiện tại của trò chơi, trong đó mỗi ô có thể có một màu sắc hoặc trống (thường được biểu diễn bằng màu đen hoặc giá trị 0).

+ Mảng của khối: Đây là mảng mô tả hình dạng và kích thước của khối hiện tại.

+ Kiểm tra từng ô của khối: Bạn sẽ kiểm tra từng ô của khối để xác định xem khi di chuyển sang trái một ô, có bất kỳ ô nào của khối chạm vào một ô không trống trong mảng khung game hay không.

* Xử lý va chạm

+ Nếu có va chạm: Nếu bất kỳ ô nào của khối chạm vào một ô không trống trong mảng khung game, không cho phép di chuyển sang trái.

+ Nếu không có va chạm: Nếu tất cả các ô của khối đều không chạm vào ô không trống trong mảng khung game, cho phép di chuyển khối sang trái một ô

* Sang phải
* Kiểm tra vị trí mới

+ Tọa độ hiện tại:Giả sử tọa độ của khối hiện tại là(x,y).

+ Tọa độ mới:Khi di chuyển sang phải, tọa độ mới sẽ là (x+1,y).

* Kiểm tra va trạm

+ Mảng khung game: Đây là mảng lưu trạng thái hiện tại của trò chơi, trong đó mỗi ô có thể có một màu sắc hoặc trống (thường được biểu diễn bằng màu đen hoặc giá trị 0).

+ Mảng của khối: Đây là mảng mô tả hình dạng và kích thước của khối hiện tại.

+ Kiểm tra từng ô của khối: Bạn sẽ kiểm tra từng ô của khối để xác định xem khi di chuyển sang phải một ô, có bất kỳ ô nào của khối chạm vào một ô không trống trong mảng khung game hay không.

* Xử lý va chạm

+ Nếu có va chạm: Nếu bất kỳ ô nào của khối chạm vào một ô không trống trong mảng khung game, không cho phép di chuyển sang phải.

+ Nếu không có va chạm: Nếu tất cả các ô của khối đều không chạm vào ô không trống trong mảng khung game, cho phép di chuyển khối sang phải một ô

* Xoay gạch
* Lưu lại mảng của gạch trước khi xoay:Trước khi thực hiện xoay khối, lưu lại trạng thái hiện tại của mảng khối.
* Xoay khối:Thực hiện xoay khối. Tùy thuộc vào quy tắc của trò chơi, điều này có thể là xoay theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ.
* Kiểm tra va chạm:Kiểm tra mảng khung game để xem mảng khối mới có trùng với bất kỳ ô nào không trống (màu đen) hay không.
* Xử lý va chạm:

+ Nếu có va chạm: Khôi phục lại mảng khối cũ.

+ Nếu không có va chạm: Giữ nguyên mảng khối đã thay đổi.

## **3.2. Thiết kế**

### **3.2.1. Biểu đồ use – case tổng quát**

A diagram of a person with text

Description automatically generated

Hình 3: *Biểu đồ user – case tổng quát*

**Đặc tả User case**

**+User case Play**

|  |  |
| --- | --- |
| Mô tả | Cho phép người dùng vào màn hình chơi game |
| Tiền điều kiện | Trò chơi Tetris đã khởi động , đang ở trạng thái màn hình chờ và người chơi sẵn sàng tham gia |
| Luồng sự kiện | 1. Nếu người chơi vừa vào màn hình chờ hoặc ấn nút EXIT khi game tạm dừng hoặc ấn nút EXIT khi game kết thúc    1. Hệ thống hiển thị nút PLAY    2. Người chơi click nút PLAY    3. Hệ thống hiển thị các chế độ chơi tính điểm (MARATHON) và tính thời gian (SPRINTS)    4. Nếu người chơi chọn chế độ chơi tính điểm (MARATHON)    5. Hệ thống chạy game và tính điểm khi ăn các khối    6. Nếu người chơi chọn chế độ chơi tính thời gian(SPRINTS)    7. Hệ thống sẽ chạy game và bắt đầu tính thời gian chơi game 2. Nếu người chơi đã bị thua    1. Hệ thống sẽ thông báo bạn thua và hiển thị nút PAUSE    2. Người chơi bấm vào nút pause sẽ hiển thị thông báo game tạm dừng và hiển thị ra các nút EXIT, nút RESUME và nút RESTART       1. Nếu người chơi bấm vào nút RESTART   2.2.2. Khung game xóa hết gạch và trở về trạng thái sẵn sàng  2.2.3. Hệ thống bắt đầu game mới với các thông số ban đầu   * 1. Nếu người chơi bấm nút EXIT      1. Game sẽ trở về màn hình chờ   3. Nếu game đang được chơi, người chơi bấm nút PAUSE  3.1. Hệ thống sẽ hiển thị thông báo game tạm dừng và hiển thị ra các nút EXIT, nút RESUME và nút RESTART  3.2.1. Người chơi bấm nút RESUME  3.2.1.1. Hệ thống chuyển về trạng thái trước khi nhấn PAUSE  3.2.1.2. Thông báo game tạm dừng sẽ bị ẩn đi  3.2.2. Nếu người chơi bấm nút EXIT  3.2.2.1. Hệ thống trở về màn hình chờ  3.2.3. Nếu người chơi bấm nút RESTART  3.2.3.1. Khung game xóa hết gạch và trở về trạng thái sẵn sàng  3.2.3.2 Hệ thống bắt đầu game mới với các thông số ban đầu |
| Hậu điều kiện | Người chơi bắt đầu chơi ván mới và các khối gạch bắt đầu rơi xuống. |

**+User case Setting**

|  |  |
| --- | --- |
| Mô tả | Cho phép người chơi thực hiện cài đặt lại hệ thống (chế độ chơi, bật/tắt âm thanh, độ nhanh chậm của các khối) |
| Tiền điều kiện | Trò chơi Tetris đã khởi động đang ở trạng thái màn hình chờ người chơi muốn cài đặt trạng thái của trò chơi khi chơi |
| Luồng sự kiện | 1. Người chơi vào màn hình chờ 2. Hệ thống game hiển thị nút SETTING 3. Người chơi click chọn nút 4. Hệ thống sẽ hiển thị cài đặt hệ thống xoay (Rotation System), Hệ thống ngẫu nhiên (Randomize System), xem nhật kí điểm (Show Points Log), tốc độ của khối di chuyển tự động (DAS) và tốc độ chuyển hướng của các khối (ARR), tốc độ rơi tự do của các khối (Soft Drop) và bật/tắt âm thanh, Reset All Scores and Times 5. Người chơi sẽ chọn một cách chơi, thứ tự rơi các khối,chọn hiển thị điểm, chọn tốc độ di chuyển và xoay các khối,chọn tốc độ rơi tự do của các khối hoặc để mặc định.    1. Hệ thống sẽ xử lí các lựa chọn của người chơi    2. Người chơi ấn EXIT SETTING trở về màn hình chờ. |
| Hậu điều kiện | Hệ thống thực hiện cài đặt lại theo lựa chọn của người chơi |

**+User case Exit**

|  |  |
| --- | --- |
| Mô tả | Cho phép người chơi thoát game ra màn hình điện thoại |
| Tiền điều kiện | Trò chơi Tetris đã khởi động đang ở trạng thái màn hình chờ |
| Luồng sự kiện | 1. Người chơi vào màn hình chờ hoặc ấn EXIT khi game tạm dừng hoặc bị thua  2. Người chơi bấm nút EXIT game sẽ chở về màn hình điện thoại |
| Hậu điều kiện | Trở về màn hình điện thoại |

### **3.2.2. Biểu đồ hoạt động (Activity Diagram)**

* Biểu đồ hoạt động bắt đầu chơi

A diagram of a diagram

Description automatically generated

Hình 4: Biểu đồ hoạt động bắt đầu chơi

* Biểu đồ hoạt động chế độ marathon

A diagram of a flowchart

Description automatically generated

Hình 5: Biểu đồ hoạt động chế độ Marathon

* Biểu dồ hoạt động chế độ sprinA diagram of a company

  Description automatically generated with medium confidence

Hình 6: Biểu đồ hoạt động chế độ Sprin

### **3.2.3. Biểu đồ trạng thái (State Diagram)**

A diagram of a diagram

Description automatically generated

Hình 7: Biểu đồ trạng thái

### **3.2.4. Biểu đồ thành phần (Component Diagram)**

**A computer screen shot of a computer screen

Description automatically generated**

Hình 8: Biểu đồ thành phần

# **CHƯƠNG IV: CÀI ĐẶT CHƯƠNG TRÌNH**

## **4.1. Giao diện bắt đầu game**

A screenshot of a game

Description automatically generated

## **4.2. Giao diện chọn chế độ**

A screenshot of a video game

Description automatically generated

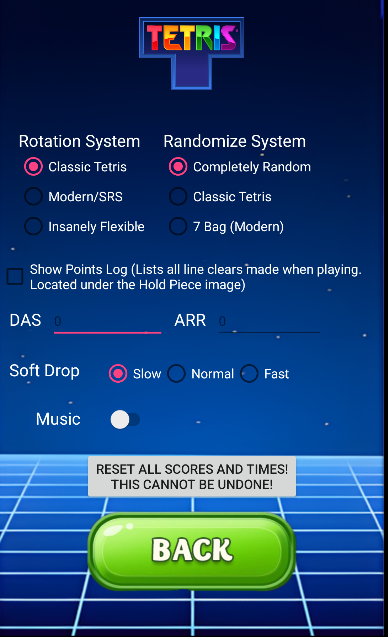
## **4.3. Giao diện chơi game**

A screenshot of a video game

Description automatically generatedA screenshot of a video game

Description automatically generated

## **4.4. Giao diện Setting**

****

# **KẾT LUẬN**

Sau thời gian tìm hiểu, nghiên cứu tài liệu chúng em đã hoàn thành thành bài tập lớn với đề tài “Xây dựng phần mềm chơi game Tetris”. Nhóm chúng em đã đạt được những kết quả sau:

- Thực hiện đầy đủ những chức năng cần có của một phần mềm chơi game Tetris cơ bản.

- Đã tìm hiểu được những kiến thức cơ bản nội dung và cách lập trình với Android.

- Nắm bắt được cách thức hoạt động và triển khai cài đặt game Tetris.

- Nắm vững được ngôn ngữ lập trình Java và cách dùng Android Studio để cài đặt chương trình cho ứng dụng di động.

- Áp dụng một số kỹ thuật để viết chương trình như: hiển thị view trên activity, share references.

- Chương trình được thiết kế đơn giản, dễ dàng cài đặt và sử dụng, truy cập dữ liệu nhanh chóng, thao tác đơn giản.

Hướng phát triển của đề tài trong tương lai:

* Chức năng người chơi tùy chỉnh, tự thiết kế các khối gạch
* Chức năng tăng giảm số cấp để thắng game
* Chức năng chọn cấp cao hơn
* Lưu game để chơi lại cho từng user có nhu cầu

# **TÀI LIỆU THAM KHẢO**

[1]. NEOS. Thanh, Lập trình hướng đối tượng JAVA core dành cho người mới bắt đầu học lập trình, 2017.

[2] TS. Lê Văn Phùng, Kỹ thuật phân tích và thiết kế hệ thống thông tin hướng cấu trúc, NXB Thông tin và Truyền thông.

[3]. Nguyễn Văn Vỵ. *Bài giảng phát triển phần mềm hướng đối tượng và sử*

*dụng lại*. Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội. 2008.

[4]. Các trang web:https://androidcoban.com/category/ebook-android,

http://developer.Android.com/

[5]. Ths.Lê Thị Kiều Oanh, Ths.Đào Thị Phương Anh, *Tài liệu học tập Cơ sở*

*dữ liệu*, Trường Đại học Kinh tế Kỹ thuật Công nghiệp, 2019.

[6]. Unlocking Android, Professional Android Application Development