**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**------------- o0o -------------**



**BÁO CÁO**

**LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG**

***ĐỀ TÀI***

**GAME MARIO BROS SE**

**Giảng viên hướng dẫn:**

**Thầy Huỳnh Tuấn Anh**

**Lớp: SE346.K11.PMCL**

**Nhóm sinh viên thực hiện:**

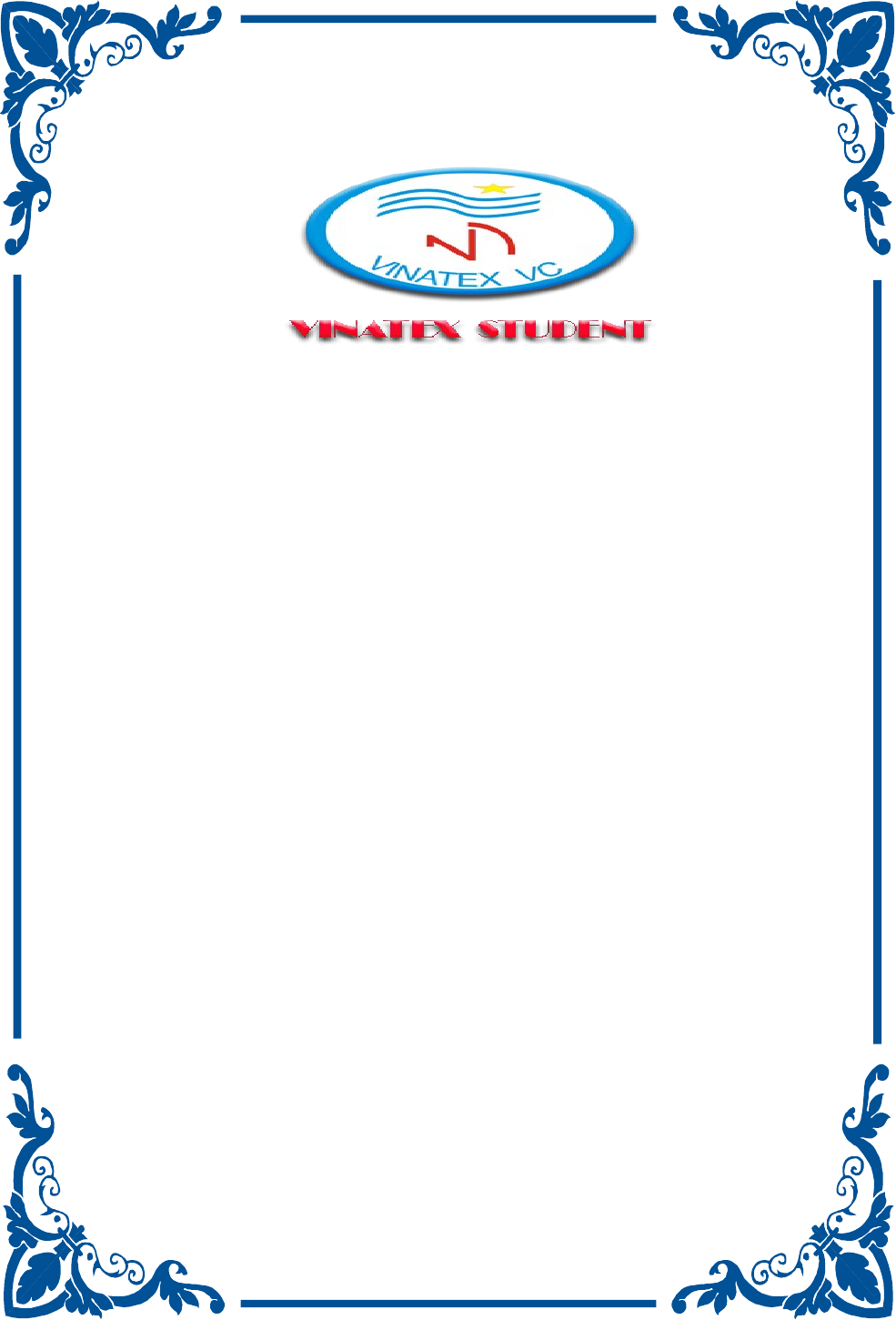
**Nguyễn Huỳnh Sáng – 16521022**

**Bùi Đang Quy – 16521009**

**Nguyễn Ngọc Nhật Minh – 16520742**

**Lê Khắc Hậu Linh – 16520664**

*TP.HCM, 23/12/2019*



**NHẬN XÉT CỦA GIẢNG VIÊN**

**LỜI CẢM ƠN**

Lời đầu tiên, cả nhóm xin gửi lời cảm ơn đến Thầy Huỳnh Tuấn Anh. Thầy đã nhiệt tình giảng dạy trên lớp, hỗ trợ những thông tin cần thiết và giải đáp những thắc mắc cho nhóm và các bạn trong suốt quá trình thực hiện đề tài. Thông qua những bài giảng trên lớp về môn học của Thầy, các thành viên trong nhóm đã tiếp thu được nhiều kiến thức hữu ích liên quan như các khái niệm và kiến thức về lập trình trên thiết bị di động. Qua đó, giúp chúng em có thể hiểu và nắm bắt được cách để phát triển một chương trình phần mềm hoàn chỉnh.

Đề tài “Game Mario Bros SE” hoàn thành là kết quả của quá trình nghiên cứu nghiêm túc của các thành viên của cả nhóm trong quá trình học tập và tiếp thu kiến thức dưới sự hướng dẫn tận tình của giảng viên hướng dẫn và các tài liệu được giảng viên hướng dẫn chia sẻ qua các kênh giao tiếp trong suốt thời gian giảng dạy.

Đồng thời nhóm cũng muốn cảm ơn các anh chị khóa trên, đặc biệt là anh chị trong khoa đã chia sẽ kinh nghiệm quý báu về môn học cũng như những kiến thức liên quan. Cũng xin cảm ơn bạn bè đã tạo điều kiện thuận lợi, mọi người đã đưa ra nhận xét và góp ý chân thành, vô cùng quý giá. Những người đã động viên, hỗ trợ nhóm hoàn thành đề tài.

*Tp. Hồ Chí Minh, tháng 12 năm 2019*

MỤC LỤC

[THÔNG TIN NHÓM 1](#_Toc27996197)

[PHẦN 1: GIỚI THIỆU 2](#_Toc27996198)

[I. GIỚI THIỆU ĐỀ TÀI 2](#_Toc27996199)

[II. MỤC TIÊU 2](#_Toc27996200)

[III. CÁC CHỨC NĂNG CHÍNH 3](#_Toc27996201)

[PHẦN 2: THIẾT KẾ 3](#_Toc27996202)

[I. SƠ LƯỢC VỀ JAVA 3](#_Toc27996203)

[II. SƠ LƯỢC VỀ libGDX 6](#_Toc27996204)

[III. KIẾN TRÚC CHƯƠNG TRÌNH 7](#_Toc27996205)

[1. Vòng đời của 1 Activity 7](#_Toc27996206)

[2. Game Framework 9](#_Toc27996207)

[3. Các lớp và đối tượng 12](#_Toc27996208)

[4. Giao diện 41](#_Toc27996209)

[PHẦN 3: CÀI ĐẶT THỬ NGHIỆM 48](#_Toc27996210)

[I. Môi trường 48](#_Toc27996211)

[II. Nền tảng công nghệ 48](#_Toc27996212)

[III. Thử nghiệm 48](#_Toc27996213)

[PHẦN 4: KẾT LUẬN, HƯỚNG MỞ RỘNG 49](#_Toc27996214)

[I. Kết quả và đánh giá 49](#_Toc27996215)

[II. Hạn chế 49](#_Toc27996216)

[III. Hướng phát triển 49](#_Toc27996217)

[PHẦN 5: TƯ LIỆU THAM KHẢO 50](#_Toc27996218)

# **THÔNG TIN NHÓM**

1. **– Thông tin nhóm**

|  |  |  |  |
| --- | --- | --- | --- |
| MSSV | Họ tên | Email | Vai trò |
| 16521022 | Nguyễn Huỳnh Sáng | 16521022@gm.uit.edu.vn | Trưởng nhóm |
| 16521009 | Bùi Đang Quy | 16521009@gm.uit.edu.vn | Thành viên |
| 16520742 | Nguyễn Ngọc Nhật Minh | 16520742@gm.uit.edu.vn | Thành viên |
| 16520664 | Lê Khắc Hậu Linh | 16520664@gm.uit.edu.vn | Thành viên |

1. **– Phương thức làm việc**

**Quy trình**

* Thống nhất đề tài đồ án
* Phân công nhiệm vụ cho từng thành viên
* Tìm kiếm tài liệu, lên kế hoạch cho các mốc thời gian cho đồ án
* Tiến hành các buổi họp nhóm, trao đổi thông tin, quy trình, bàn bạc kế hoạch thực hiện, phân công nhiệm vụ cho mỗi thành viên và thống nhất thời gian dealine
* Hoàn thành đồ án, cho ra sản phẩm hoàn chỉnh
* Đánh giá và thử nghiệm sản phẩm, hoàn thành báo cáo đồ án

**Công cụ**

* Android Studio
* Github
* Tiled

# GIỚI THIỆU

## GIỚI THIỆU ĐỀ TÀI

Trong thời đại ngày nay, với sự phát triển nhanh chóng của công nghệ thông tin, cùng với sự bùng nổ của nền “**cách mạng công nghiệp 4.0**”, công nghệ thông tin được áp dụng trên mọi lĩnh vực của cuộc sống như kinh tế, chính trị, văn hoá xã hội, giải trí,… Tất cả đều cần đến sự hỗ trợ rất lớn từ công nghệ thông tin và Internet. Như chúng ta đã biết lập trình java đang được xem là một ngôn ngữ rất hot trong các nhóm ngành của công nghệ thông tin. Nhận thức được tầm quan trọng của ngôn ngữ này cũng như nhu cầu giải trí trên thiết bị di động ngày càng trở thành một xu thế mới, để đáp ứng một phần nào đó nhu cầu giải trí, chơi game giải trí trên các thiết bị di động, chúng em đã quyết định sẽ xây dựng và thiết kế một game di động lạ mà quen bằng ngôn ngữ Java – Super Mario.

Lấy ý tưởng từ game Super Mario cổ điển, với 2 nhân vật huyền thoại là Mario và Luigi vượt qua những cây nấm độc Goomba và ăn những cây nấm đặc biệt để trở nên to lớn hơn với mục đích cuối cùng là giải cứu công chúa. Trò chơi Mario Bros được xây dựng với cốt truyện và nội dung tương tự, những được phát triển để có thể chơi được trên đa nền tảng: các thiết bị di động chạy android, iOS và cả các thiết bị PC, laptop nhờ vào ngôn ngữ: Java và sự hỗ trợ của: Android Studio và libGDX.

## MỤC TIÊU

Ứng dụng game Mario Bros SE chạy được trên đa nền tảng: các thiết bị di động chạy android, iOS và cả các thiết bị PC, laptop, được thực hiện dựa trên hai mục đích chính sau:

1. Nghiên cứu về ngôn ngữ lập trình Java
2. Nghiên cứu các công nghệ lập trình trên thiết bị di động
3. Phát triển ứng dụng có tính giải trí cao, có khả năng triển khai vào thực tế, giúp những người dùng thiết bị di động có thêm một lựa chọn để giải trí.

Em hi vọng dựa trên nền tảng lí thuyết mà Thầy truyền thụ lại, kết hợp với sự tìm hiểu của nhóm em, đề tài của chúng em sẽ đạt được mục đích mong đợi.

## CÁC CHỨC NĂNG CHÍNH

* Người dùng khi vào game chọn bắt đầu điều khiển Mario bằng các button trên màn hình di chuyển trái phải nhảy, và bắn fireball
* Người dùng có thể đục các thùng Coin để ăn điểm hay giết các enemy
* Nếu va chạm với enemy hay nhảy xuống vực hoặc rơi vào bẫy Mario sẽ chết và chuyển đến màng hình gameover

# THIẾT KẾ

## SƠ LƯỢC VỀ JAVA

* Ngôn ngữ [lập trình Java](http://hoclaptrinhweb.org/hoc-lap-trinh/hoc-lap-trinh-java.html) ban đầu được phát triển bởi Sun Microsystems do James Gosling khởi xướng và phát hành vào năm 1995 (Java 1.0 [J2SE]). Tính đến thời điểm này (tháng 2/2015) phiên bản mới nhất của Java Standard Edition (JSE) là 8. Với ưu thế về đa nền tảng (multi platform) Java càng lúc càng được ứng dụng rộng rãi trên nhiều thiết bị từ máy tính đến mobile và nhiều thiết bị phần cứng khác...
* Java là ngôn ngữ lập trình hướng đối tượng nên nó cũng có 4 đặc điểm chung của các ngôn ngữ hướng đối tượng:
* **Tính trừu tượng (Abstraction)**: là tiến trình xác định và nhóm các thuộc tính, các hành động liên quan đến một thực thể đặc thù, xét trong mối tương quan với ứng dụng đang phát triển.
* **Tính đa hình (Polymorphism)**: cho phép một phương thức có các tác động khác nhau trên nhiều loại đối tượng khác nhau. Với tính đa hình, nếu cùng một phương thức ứng dụng cho các đối tượng thuộc các lớp khác nhau thì nó đưa đến những kết quả khác nhau. Bản chất của sự việc chính là phương thức này bao gồm cùng một số lượng các tham số.
* **Tính kế thừa (Inheritance)**: Điều này cho phép các đối tượng chia sẻ hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại.
* **Tính đóng gói (Encapsulation)**:  là tiến trình che giấu việc thực thi những chi tiết của một đối tượng đối với người sử dụng đối tượng ấy.
* Bên cạnh đó Java còn có một số đặc tính khác:
* **Độc lập nền (Write Once, Run Anywhere)**: Không giống như nhiều ngôn ngữ lập trình khác như C và C ++, khi Java được biên dịch, nó không được biên dịch sang mã máy cụ thể, mà thay vào đó là mã byte code chạy trên máy ảo Java (JVM). Điều này đồng nghĩa với việc bất cứ thiết bị nào có cài đặt JVM sẽ có thể thực thi được các chương trình Java.
* **Đơn giản**:[học Java](http://hoclaptrinhweb.org/hoc-lap-trinh/hoc-lap-trinh-java.html) thật sự dễ hơn nhiều so với C/C++, nếu bạn đã quen với các ngôn ngữ lập trình hướng đối tượng thì việc học Java sẽ dễ dàng hơn. Java trở nên đơn giản hơn so với C/C++ do đã loại bỏ tính đa kế thừa và phép toán con trỏ từ C/C++.
* **Bảo mật**: Java hỗ trợ bảo mật rất tốt bởi các thuật toán mã hóa như mã hóa một chiều (one way hashing) hoặc mã hóa công cộng (public key) ...
* **Đa luồng**: Với tính năng đa luồng Java có thể viết chương trình có thể thực thi nhiều task cùng một lúc. Tính năng này thường được xử dụng rất nhiều trong lập trình game.
* **Hiệu suất cao**nhờ vào trình thu gom rác (garbage collection), giải phóng bộ nhớ đối với các đối tượng không được dùng đến.
* **Linh hoạt**: Java được xem là linh hoạt hơn C/C ++ vì nó được thiết kế để thích ứng với nhiều môi trường phát triển.
* Ưu điểm của Java: Cũng như nhiều ngôn ngữ lập trình khác, ngôn ngữ Java cũng có cho mình khá nhiều ưu điểm. Trong đó cần được kể tới như:
  + Hướng đối tượng rộng : Hướng đối tượng rộng trong Java chính là tất cả những thứ đều được mở rộng, trong đó thì Java sẽ được dùng dựa trên các mô hình là Object.
  + Có một nền tảng riêng biệt: Java có nền tảng riêng biệt, người ta nói như vậy là bởi khi nhận được một câu lệnh nào đó, thì Java sẽ tự động thực hiện biên tập câu lệnh đó sang những Bite Code ở dạng độc lập. Trong đó, Bite Code độc lập này sẽ được hỗ trợ bởi các dịch bằng Vitual Machile với bất cứ phần mềm, ứng dụng nào có sử dụng tới nó.
  + Thiết kế mẫu đơn giản : Không giống như nhiều ngôn ngữ lập trình khác, Java có thiết kế mẫu khá là đơn giản bởi thế mà những nhà lập trình viên không cần phải mất quá nhiều thời gian theo học. Muốn học tốt, thành thạo về Java thì mỗi người chỉ mất từ 1 đến 3 năm là đã có thể thành công.
  + Tính bảo mật cao : Tính bảo mật cao, chính là một ưu điểm của Java so với các ngôn ngữ khác. Trong đó, khả năng của Java là phát hiện được những thành phần có chứa các virut độc hại, rồi sau đó nó cũng có thể “tiêu diệt” được virut đó. Để thực hiện được điều này, những nhà lập trình viên ra Java đã phát triển cho nó những thuật toán ở mức độ cao nhất.
  + Nhanh và mạnh : Đối với ưu điểm này, Java là một ngôn ngữ có được khả năng xử lý những tình huống bị xảy ra trên máy chủ rất nhanh. Bên cạnh đó, nó cũng có được khả năng truyền dẫn về internet với tốc độ cao, không kém gì những ứng dụng khác.

## SƠ LƯỢC VỀ libGDX

* LibGDX là một framework phát triển ứng dụng game, được viết bằng ngôn ngữ lập trình Java, một số các thành phần được viết bằng C và C++ để có hiệu năng tốt hơn. Nó cho phép phát triển các ứng dụng desktop và mobile trên cùng một code base. LibGDX chạy được đa nền tảng, hỗ trợ Windows, Linux, Mac OS X, Android, iOS, và trên web browsers với WebGL.

LibGDX có nhiều ưu điểm như:

1. Đa nền tảng: chúng ta chỉ cần viết code 1 lần nhưng có thể chạy trên được nhiều nền tảng khác nhau. Một ứng dụng của tính năng này đó là phát triển các ứng dụng cho Android. Để phát triển một ứng dụng cho Android, khi chạy thử ứng dụng, chúng ta cần chạy ứng dụng trên Emulator hoặc trên thiết bị thật. Việc này rất mất thời gian do Emulator chạy rất chậm và quá trình cài đặt cũng như chạy ứng dụng trên thiết bị thật cũng không khá hơn. Với LibGDX, chúng ta có thể chạy ứng dụng trên PC, sau đó chỉ cần với vài dòng code, chúng ta có thể chạy ứng dụng này trên Android với hiệu năng tương đương. Điều này giúp chúng ta kiểm thử và tìm lỗi ứng dụng nhanh hơn và hiệu quả hơn.
2. Hiệu năng: Hiệu năng của LibGDX thực sự rất ấn tượng do LibGDX do các thành phần được viết bằng C và C++.
3. Cộng đồng: cộng đồng sử dụng LibGDX rất tuyệt vời với số lượng người dùng lớn. Các lập trình viên luôn đóng góp và giúp đỡ cho cộng đồng. Việc sửa lỗi cũng được cập nhật rất thường xuyên.
4. Tài liệu và ví dụ: rất đầy đủ với Javadoc. LibGDX cũng cung cấp rất nhiều các ví dụ với đầy đủ các chức năng từ đơn giản đến phức tạp.
5. Mã nguồn: mã nguồn mở với thiết kế rất rõ ràng và phù hợp với việc phát triển ứng dụng cho di động. LibGDX cho phép người lập trình khả năng sử dụng các API từ các lớp thấp đến cao, tùy theo yêu cầu của người sử dụng.
6. Tính năng: LibGDX có rất nhiều tính năng như tạo hình, xử lý đồ họa 2D, 3D, xử lý âm thanh, quản lý các thiết bị vào ra, quản lý file hệ thống. Cùng với đó là các công cụ đi kèm rất hữu ích như Texture Packer và Particle Editor.

## KIẾN TRÚC CHƯƠNG TRÌNH

### Vòng đời của 1 Activity

Các Activity trong hệ thống được quản lý như 1 ngăn xếp activity (activity stack). Khi một activity mới bắt đầu nó được đặt lên đầu của ngăn xếp và trở thành Running Activity (activity đang chạy), đồng thời activity trước đó sẽ nằm ngay phía dưới trong ngăn xếp đó, và sẽ không trở nên visible (nhìn thấy) cho đến khi activity ở trên thoát ra khỏi ngăn xếp.

Một Activity gồm 4 trạng thái chính:

1. Nếu activity ở phía trên của màn hình (hay ở trên cùng của ngăn xếp), thì nó đang ở trạng thái **active** (hoạt động) / **running** (đang chạy). Ví dụ khi ta cần gọi điện thì activity bấm số đó đang ở trạng thái active.
2. Nếu activity không thể tương tác nhưng vẫn nhìn thấy ( khi mà bị che bởi 1 activity khác nhưng người dùng vẫn có thể nhìn thấy nó ở phía sau ) thì activity này đang ở trạng thái **paused** (tạm dừng). Khi ở trạng thái này activity có thể bị xóa bỏ bởi hệ thống khi thiết bị thiếu bộ nhớ. Ví dụ khi có 1 activity khác dạng dialog hiện lên chỉ che đi 1 phần của activity hiện tại thì activity vào trạng thái paused.
3. Nếu activity hoàn toàn bị che khuất bởi activity khác thì nó đang ở trạng thái **stopped** (đã dừng). Activity này vẫn giữ được tất cả trạng thái và thông tin, nhưng không còn hiển thị với người dùng và thường xuyên bị xóa bỏ bởi hệ thống khi thiếu bộ nhớ. Ví dụ khi ta tắt màn hình thì khi đó activity vào trạng thái stopped.
4. Nếu activity ở trạng thái **paused** (tạm dừng) hay **stopped** (đã dừng), hệ thống có thể xóa bỏ activity đó khỏi bộ nhớ bằng cách yêu cầu nó tự kết thúc hoặc xóa bỏ tiến trình của nó. Khi acitivty đó hiển thị lại với người dùng thì sẽ được khởi tạo lại và khôi phục lại trạng thái trước đó.

Hình ảnh sau đây minh họa cho vòng đời của 1 Activity cùng với các trạng thái của nó:



### Game Framework

#### Aspect Ratios (tỉ lệ khung hình)

Để game có thể chạy đúng tỉ lệ trên một màn hình lớp, hay một màn hình có kích thước nhỏ hơn.

Một số tỉ lệ khung hình thường gặp:

A screenshot of a cell phone

Description generated with very high confidence

#### ViewPort

ViewPort sẽ được xây dựng dựa vào kích thước của màn hình để tránh trường hợp device có màn hình lớp có thể nhìn được nhiều hơn trong thế giới game => điều đó không công bằng… Vì vậy Viewport sẽ được quy ước 1 đơn vị worldgame sẽ bằng 1 pixel, tỉ lệ đó có thể thay đổi.

ViewPort và Game World

A close up of a logo

Description generated with high confidence

#### TileMap

Tile Map được tạo từ nhiều Tile. Tile là những nơi chứa những hình ảnh dùng để tạo nên Tile Map được gọi là Tileset. Cũng có thể hiểu Tileset là một Sprite trong một Sprite Sheet.

Đây là một Tile Map 20x15

A picture containing sky, flying

Description generated with high confidence

Tileset là một sprite (mỗi ô vuông) trong sprite sheet

A picture containing photo, indoor

Description generated with high confidence

### Các lớp và đối tượng

#### Class Diagram

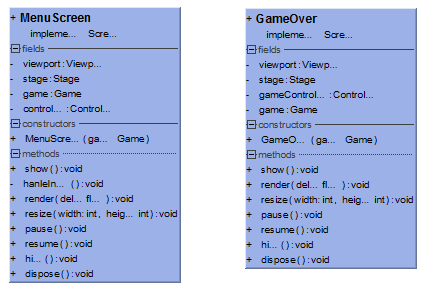
#### Các lớp và đối tượng

##### Lớp game chính



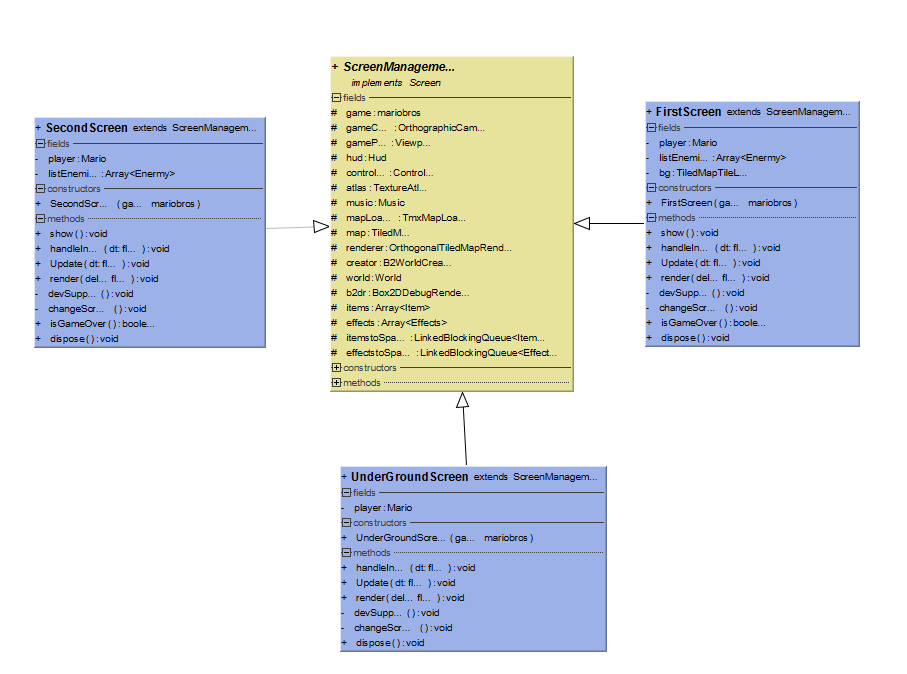
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| mariobros | Lớp đầu tiên được tự động tạo khi chạy libgdx được kế thừa từ lớp Game của thư viện, chỉ khởi tạo một lần | (AssetManager) *manager* | Biến lưu trữ âm thanh trong libGdx | (void)  Create | Khởi tạo batch, load file âm thanh từ máy |
| (SpriteBatch) batch | Biến lưu trữ toàn bộ sprite game, textutes… | (SpriteBatch)  getBatch | Trả về spritebatch của game |
| (void)  render | Vẽ liên tục các sprite lên màn hình |
| (void)  dispose | Giải phóng bộ nhớ khi thoát chương trình |
| (void)  resize | Lớp sử lý sự kiện hiện thị màn mình đúng tỷ lệ định sẵn tránh trường hợp người dùng có màn hình lớp có thể nhìn được nhiều hơn thế giới game |

##### Các lớp Screen



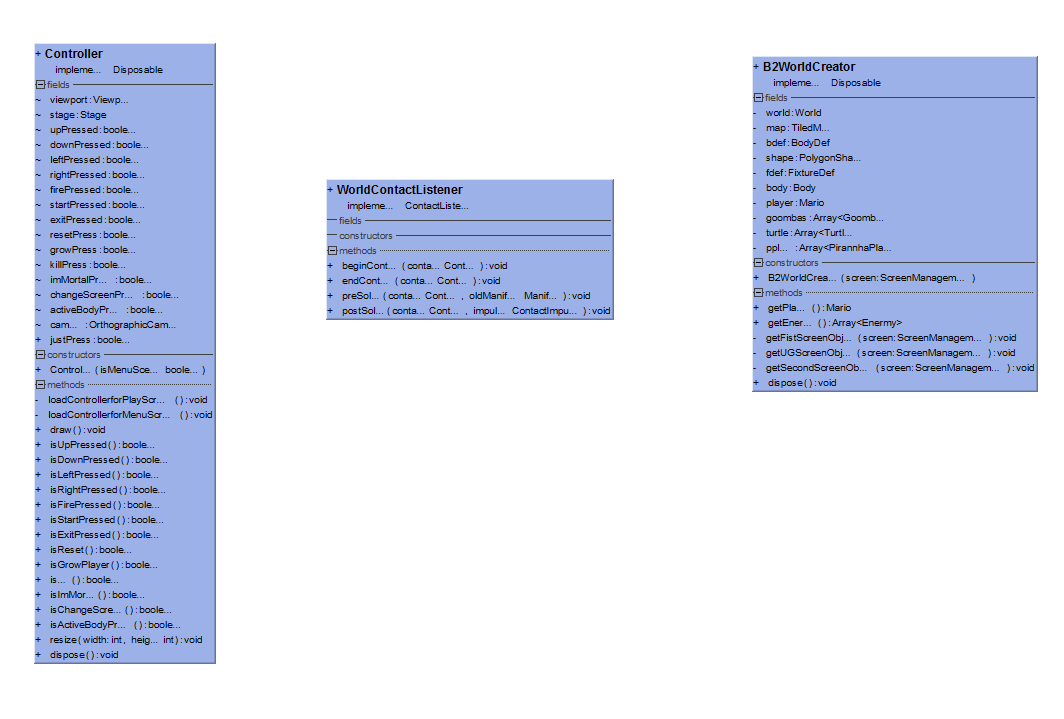
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| Menu Screen | Hiển thị màn hình menu cho người dùng lựa chọn khi bắt đầu game | (Viewport) viewport | Là vùng nhìn thất của người dùng trong thế giới game hay là vùng hiện thị trên màn hình vì thế giới game rất lớn còn vùng nhìn thấy chỉ là vùng nhỏ | (void)  handleInput | Bắt sự kiện người dùng nhấn vào các button hiện thị trên màn hình |
| GameOver | Hiển thị màn hình thông báo game over khi player die , cho người dùng lựa chọn chạm vào màn hình để tiếp tục | (Stage)  stage | Một trong những cách để vẽ lên màn hình game là dùng stage, trong stage chứa viewport và Actor(Đối tượng vẽ lên có thể là label, hình ảnh ,text hay hình chữ nhật) | (void)  render | Vẽ liên tục các sprite lên màn hình |
| (Game) game | Dùng để gọi phương thức game. setScreen  Để thay đổi màn hình hiển thị khi người dùng lựa chọn | (void)  dispose | Giải phóng bộ nhớ khi thoát chương trình |
| (Controller)  controller | Dùng để khởi tạo một đối tượng controller  Hiện thị những nút bấm trên màn hình | (void)  resize | Lớp sử lý sự kiện hiện thị màn mình đúng tỷ lệ định sẵn tránh trường hợp người dùng có màn hình lớp có thể nhìn được nhiều hơn thế giới game |

##### Các lớp Scene



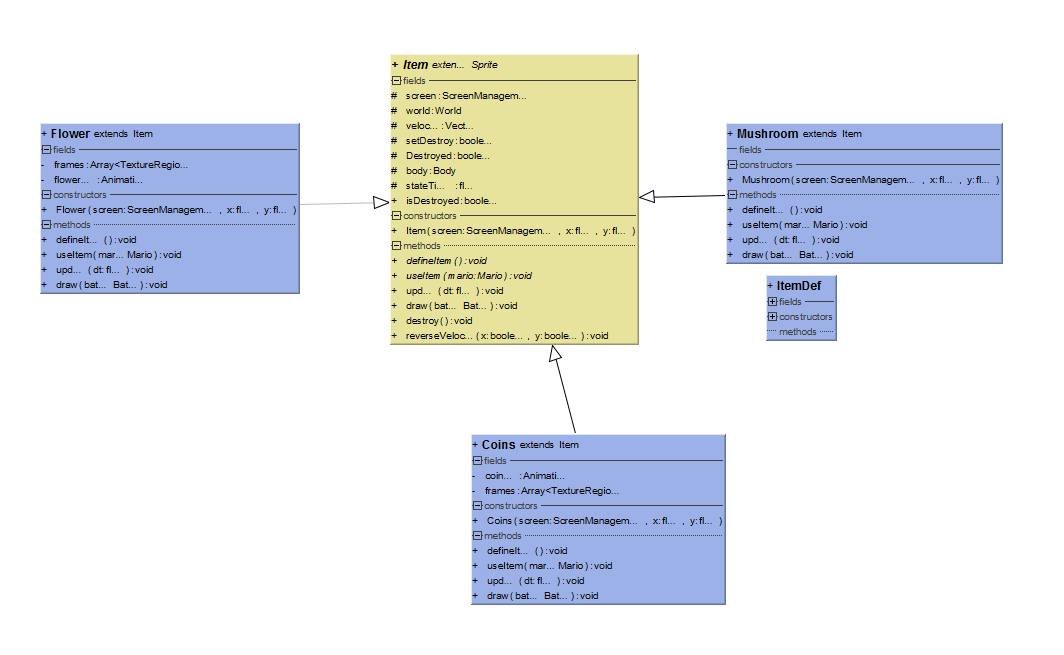
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| Screen  Management | Lớp cha quản lý các scene | mariobros game | Dùng để lấy spritebatch để vẽ | ScreenManagement | Khởi tạo các thuộc tính của game, load các file cần thiến |
| OrthographicCamera gameCam | Camera của game | void spawnItem | Thêm vào hàng đợi loại itemstoSpawn |
| Viewport gamePort | ViewPort của game | void spawnEffect | Thêm vào hàng đợi effectstoSpawn |
| Hud hud | Lưu trữ các Label như Score, gameTime, | void handleSpawningItem | Kiểm tra nếu itemstoSpawn khác rỗng, lấy ra loại item để tạo item phù hợp |
| Controller controller | Vẽ các button điều khiển lên màn hình game | void handleSpawningEffects | Kiểm tra nếu effectstoSpawn khác rỗng, lấy ra loại effect để tạo effect phù hợp |
| TextureAtlas atlas | Lưu trữ textute của các object từ file tải lên | World getWorld | Trả về World game |
| Music music | Dùng để phát các hiệu ứng âm thanh | (void)  resize | Lớp sử lý sự kiện hiện thị màn mình đúng tỷ lệ định sẵn tránh trường hợp người dùng có màn hình lớp có thể nhìn được nhiều hơn thế giới game |
| TmxMapLoader mapLoader | Dùng để tải lên file tilemap |
| TiledMap map | Map của game được tạo từ tile |
| OrthogonalTiledMapRenderer renderer | Vẽ TileMap từ phương thức map |
| B2WorldCreator creator | Lấy vị trí, số lượng của player, enemy,object của game trong file tilemap |
| World world | Thế giới game lưu trữ các “khung xương” hay “body” 1 game chỉ có 1 world duy nhất |
| Box2DDebugRenderer b2dr | Hiển thị các “khung xương” của các đối tượng trong game để cho người lập trình kiểm tra va chạm |
| Array<Item> items | Chứa các Item của scene hiện tại |
| Array<Effects> effects | Chứa các hiện ứng |
| LinkedBlockingQueue<ItemDef> itemstoSpawn | Chứa danh sách các loại Item |
| LinkedBlockingQueue<EffectDef> effectstoSpawn | Chứa danh sách các loại hiệu ứng |
| First  Screen | Scene đầu của game | Mario player | Lưu dữ liệu về Mario | (Contructer) của các lớp | Khởi tạo các phương thức của game, load map, load danh sách enemy, load player, play nhạc nền |
| Under  Ground  Sceen | Scene đặc biệt của game | Array<Enermy> listEnemies | Lưu trữ danh sách Enemy của game | void Update | Cập nhật liên tục những thay đổi trong thế giới game như thêm,xóa object, các logic game |
| Second  Screen | Scene thứ 2 của game | TiledMapTileLayer bg | Lưu trữ 1 layer từ tilemap | void render | Vẽ lên màn hình map, các object game, item, hiệu ứng… |
| void devSupport | Bắt sự kiện các phím hỗ trợ lập trình game thay đổi thế giới game mà không cần quan tâm đến logic của game |
| changeScreen | Thay đổi màn hình game khi qua màn, hoặc mở menu, game over |

##### Các Tool class



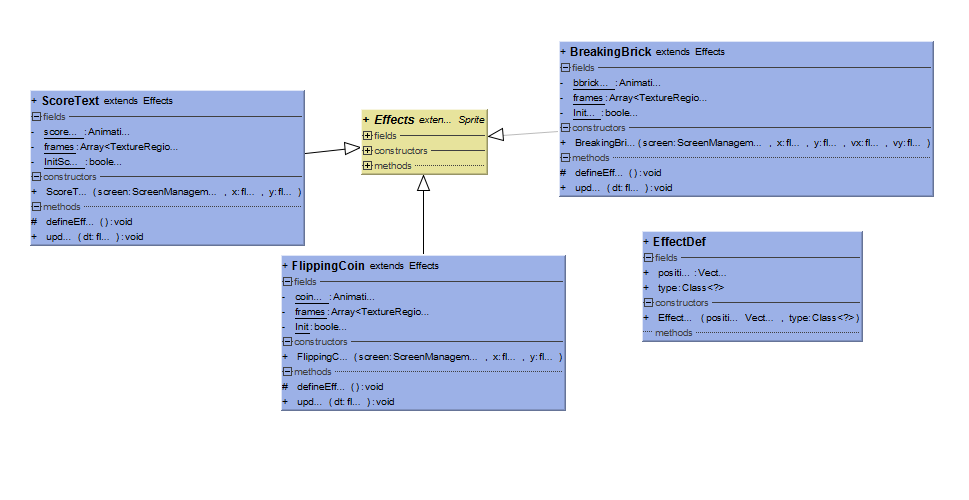
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| WorldContact  Listener | Kiểm tra va chạm của các object trong world game |  |  | void beginContact | Kiểm tra các object nếu đang va chạm nhau |
| Controller | Vẽ các button lên màn hình và bắt các sự kiện bàn phím trên máy tính hay nhấn vào các button trên màn hình của điện thoại | Viewport viewport |  | void endContact | Kiểm tra các object khi kết thúc va chạm |
| (boolean)  upPressed downPressed leftPressed  rightPressed  firePressed  startPressed  exitPressed  resetPress  growPress  killPress  ,imMortalPress  changeScreenPress  activeBodyPress | Trả về là true nến nhấn phím quy định sẵn , khi bỏ phím sẽ trả về là false | Controller | Khởi tạo và bắt các sự kiện ấn phím hay chạm vào button trên màn hình |
| OrthographicCamera  camera | Camera của game |  |  |
| loadControll  …. | Load các nút control của từng screen khác nhau |
| B2WorldCreator | Load vị trí và số lượng các object trong file tilemap | World world | Thế giới game | Mario getPlayer | Lấy dữ liệu mario vừa tạo để set vào screne |
|  |  | TiledMap map | Lưu trữ map của scene gọi vào | Array<Enermy> | Lấy danh sách enemy |
|  |  | Mario player | Tạo player và set vị trí | B2WorldCreator | Khởi tạo các thuộc tính  Kiểm tra scene nào vừa gọi để load object phù hợp |
|  |  | Array<Goomba> | Lưu danh sách goomba lấy từ tile | Get…(1,2,3) object | Lấy object theo scene |
|  |  | Array<Turtle> | Lưu danh sách Turtle lấy từ tile |
|  |  | Array<PirannhaPlant> | Lưu danh sách PirannhaPlant lấy từ tile |

##### Các lớp Item



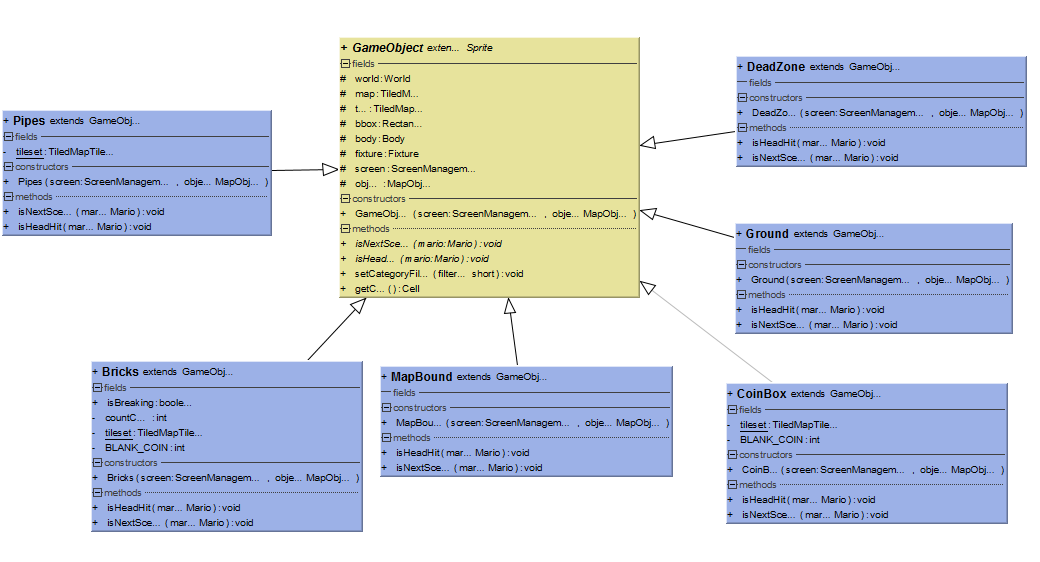
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| Item | Lớp cha, chứa các thuộc tính và phương thức chung của các item | ScreenManagement screen |  | Item | Khởi tạo các định thuộc tính chung của item |
| World world | Thế giới game | void defineItem | Khởi tạo các thuộc tính riêng của từng item |
| Vector2 velocity | Vận tốc của item | void useItem | Xử lý logic khi mario va chạm với item |
| boolean setDestroy | Cờ hiệu để bắt đầu phá hủy body của object trong world | void update | Cập nhật logic Item |
| boolean Destroyed | Cờ hiệu để biết body của object đã bị xóa hay chưa để ngừng update | void draw | Vẽ Item |
| Body body | Là tên của object trong thư viện libgdx | void reverseVelocity | Đảo ngược chiều vector vận tốc khiến item di chuyển ngược lại |
| float stateTimer | Thời gian của các frame |  |
| Coins | Lớp lưu trữ các thuộc tính và phương thức của item coin | Animation coinAni | Lưu trữ animation của coin | (Contructor) | Khởi tạo mặc định  Từng item |
| Flower | Lớp lưu trữ các thuộc tính và phương thức của item flower | Array  <TextureRegion> frames | Lưu trữ danh sách các fames của coin | void defineItem | Khởi tạo các thuộc tính riêng của từng item |
| Mushroom | Lớp lưu trữ các thuộc tính và phương thức của item Mushroom |  |  | useItem | Xử lý logic khi mario va chạm với từng item |

##### Các lớp Effect



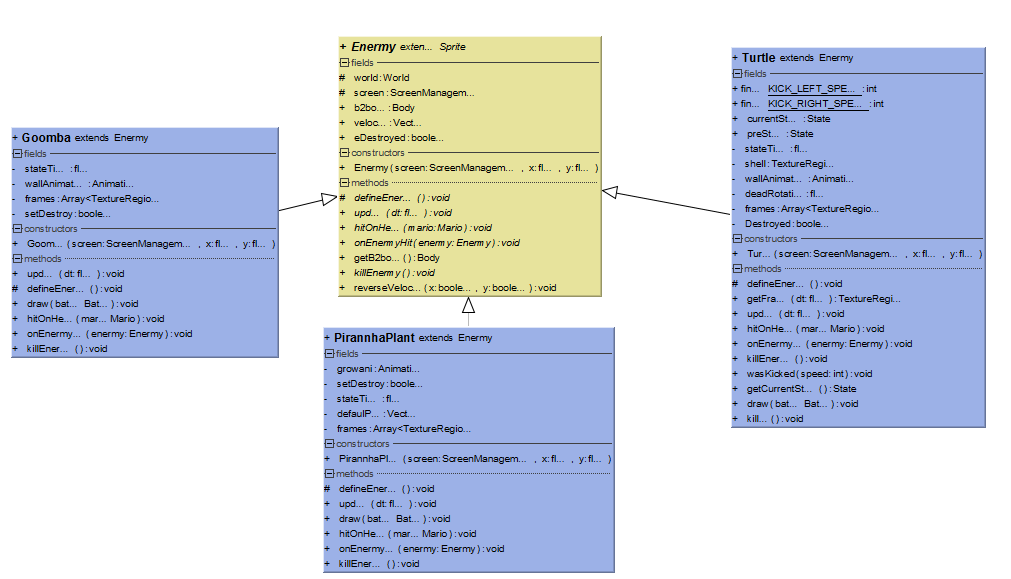
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| Effects | Lớp cha, chứa các thuộc tính và phương thức chung của các item | ScreenManagement screen |  | Effects | Khởi tạo các định thuộc tính chung của Effects |
| World world | Thế giới game | void define Effects | Khởi tạo các thuộc tính riêng của từng Effects |
| Vector2 velocity | Vận tốc của Effects | void update | Cập nhật logic Effects |
| boolean setDestroy | Cờ hiệu để bắt đầu phá hủy body của object trong world | void draw | Vẽ Effects |
| boolean Destroyed | Cờ hiệu để biết body của object đã bị xóa hay chưa để ngừng update | void reverseVelocity | Đảo ngược chiều vector vận tốc khiến Effects  di chuyển ngược lại |
| Body body | Là tên của object trong thư viện libgdx | void reverseVelocity  (Contructor) |  |
| float stateTimer | Thời gian của các frame | Khởi tạo mặc định  Từng item |
| Breaking  Brick | Lớp lưu trữ các thuộc tính và phương thức của Effect bbrick | Animation *bbrickani* | Lưu trữ animation của *bbrickani* | void defineItem | Khởi tạo các thuộc tính riêng của từng Effect |
| Flipping  Coin | Lớp lưu trữ các thuộc tính và phương thức của Effect FlippingCoin | Array  <TextureRegion> frames | Lưu trữ danh sách các fames của fcoin | useItem | Xử lý logic khi mario va chạm với từng item |
| ScoreText | Lớp lưu trữ các thuộc tính và phương thức của Effect ScoreText |  | Lưu trữ danh sách các fames của ScoreText |  |  |

##### Các lớp Static object



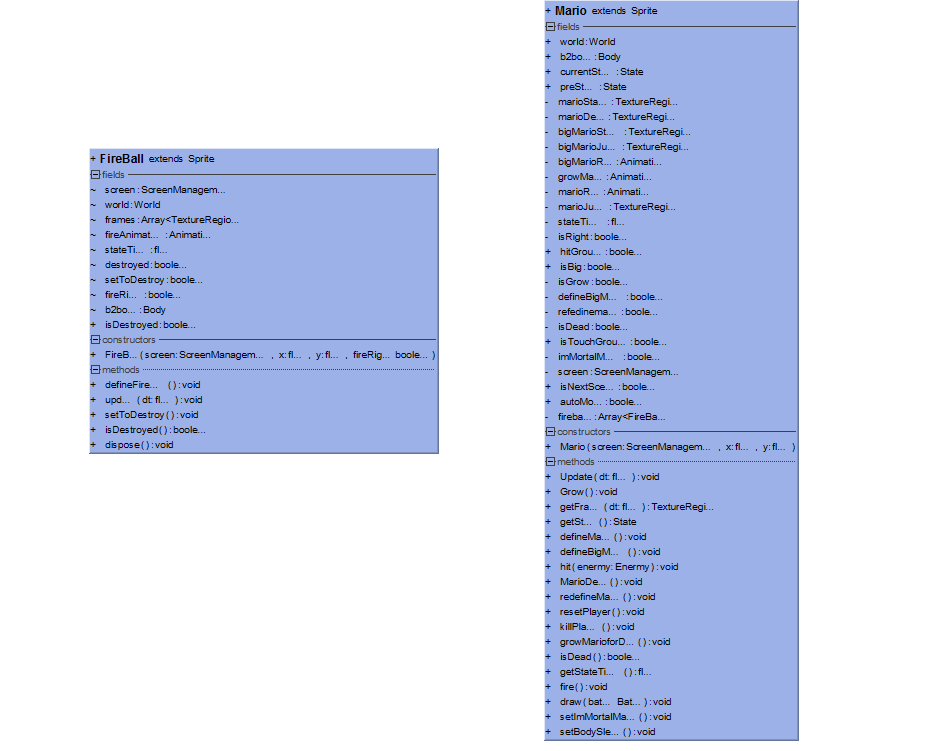
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| Game  Object | Lớp cha chứa những thuộc tính và phương thức chung của các object | World world | World game | GameObject | Khởi tạo các thuộc tính chung của object |
|  |  | TiledMap map | Lưu trữ map của scene gọi vào | void isNextScene | Kiểm tra logic chuyển scene |
|  |  | TiledMapTile tile | Get tile từ map | isHeadHit | Xử lý logic khi đầu mario va chạm object |
|  |  | Rectangle bbox | Lưu trữ vị trí của object trong map | void  setCategory  Filter | Set nhãn cho object để set va chạm |
|  |  | Body body | Body của object | getCell | Lấy ra 1 cell từ map |
|  |  | Fixture fixture | Lưu trữ các fixture của object |  |  |
|  |  | ScreenManagement screen | Lưu trữ thông tin scene gọi vào |  |  |
|  |  | MapObject object | Lưu mapobject được lấy ra từ lớp B2worldcreator |  |  |
| Pipes | Lưu trữ phương thức và thuộc tính của object Pipes |  |  | (Contructor) | Khởi tạo các thuộc tính chung và riêng của từng object |
| Ground | Lưu trữ phương thức và thuộc tính của object Ground |
| DeadZone | Lưu trữ phương thức và thuộc tính của object DeadZone |
| CoinBox | Lưu trữ phương thức và thuộc tính của object CoinBox |
| Bricks | Lưu trữ phương thức và thuộc tính của object Bricks |

##### Các lớp Enemy



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| Enermy | Lớp cha, chứa các thuộc tính và phương thức chung của các Enermy | ScreenManagement screen |  | Enermy | Khởi tạo các định thuộc tính chung của enemy |
| World world | Thế giới game | void defineEnermy | Khởi tạo các thuộc tính riêng của từng enemy |
| Vector2 velocity | Vận tốc của enemy | void killEnermy | Xử lý logic khi mario va chạm với mario hoặc fireball |
| boolean setDestroy | Cờ hiệu để bắt đầu phá hủy body của object trong world | void update | Cập nhật logic Enemy |
| boolean Destroyed | Cờ hiệu để biết body của object đã bị xóa hay chưa để ngừng update | void draw | Vẽ Enemy |
| Body body | Là tên của object trong thư viện libgdx | void hitOnHead | Xử lý logic nến mario va chạm với phần đầu của enemy |
| float stateTimer | Thời gian của các frame |
| void reverseVelocity | Đảo ngược chiều vector vận tốc khiến Enemy di chuyển ngược lại |
| Goomba | Lớp lưu trữ các thuộc tính và phương thức của item Goomba | Animation wallAnimation | Lưu trữ animation của goomba | <Contructor> | Định nghĩa các phương thức chung và riêng của từng object |
| Array<TextureRegion> frames | Lưu trữ danh sách các frame |
| Pirannha  Plant | Lớp lưu trữ các thuộc tính và phương thức của item pplant | Animation growani | Lưu trữ animation của pplant |
| Array<TextureRegion> frames | Lưu trữ danh sách các frame |
| Turtle | Lớp lưu trữ các thuộc tính và phương thức của item Turtle | enum State | Lưu trữ trạng thái của Turtle |  |
| State currentState | Trạng thái hiện tại |
| State preState | Trạng thái trước đó | State getCurrentState | Trả về trạng thái hiện tại |
| TextureRegion shell | Lưu texture shell | wasKicked | Xử lý logic khi bị mario đá vào người |
| Animation wallAnimation | Lưu trữ animation của Turtle | getFrame | Trả về frame theo stage phù hợp |
| float deadRotation | Độ xoay khi turtle die |
| Array<TextureRegion> frames | Lưu trữ danh sách các frame |

##### Lớp Mario và Fireball



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| Fire  Ball | Là vũ khí của mario  Lưu trữ thuộc tính và phương thức của fireball | ScreenManagement screen | Lưu trữ thông tin scene gọi vào | FireBall | Khởi tạo các thuộc tính |
| World world | Thế giới game | defineFireBall | Khởi tạo các thuộc tính |
| Array<TextureRegion> frames | Lưu trữ danh sách các frame | update | Cập nhật logic của fireball |
| Animation fireAnimation | Lưu trữ animation | setToDestroy | Bật cờ hiệu setToDestroy |
| float stateTime | Thời gian các frame của animation |
| boolean destroyed | Cờ hiệu báo đã xóa body khởi wolrd |
| boolean setToDestroy | Cờ hiệu báo bắt đầu xóa body khỏi world |
| boolean fireRight | Cờ hiệu set hướng bắn |
| Body b2body | Body của fireball |

* Lớp Mario

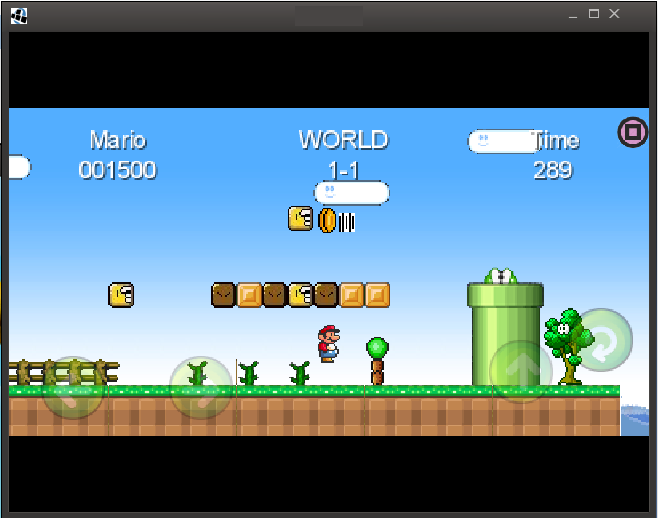
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên lớp | Chức năng | Thuộc tính | Ý nghĩa | Phương thức | Ý nghĩa |
| Mario | Nhân vật chính của chương trình | enum State | Lưu trữ các trạng thái của Mario | Mario | Khởi tạo các thuộc tính của mario |
| World world | Thế giới game | Update | Cập nhật logic của mario |
| Body b2body | Body của mario | Grow | Kiểm tra nếu mario đang nhỏ thì chạy ani lớn lên và chạy hàm defineBigMarrio |
| State currentState | Trạng thái hiện tại của mario | getFrame | Set frame phù hợp khi mario ở các trạng thái khác nhau |
| State preState | Trạng thái trước đó của mario | getState | Trả về trạng thái của mario |
| TextureRegion marioStand | Textute Mario đứng | defineMario | Khởi tạo các phương thức ban đầu của mario |
| TextureRegion marioDead | Textute Mario die | defineBigMario | Khởi tạo các phương thức khi mario ở trạng thái lớn lên |
| TextureRegion bigMarioStand | Textute Big Mario đứng | hit | Xử lý logic khi mario va chạm enemy |
| TextureRegion bigMarioJump | Textute Big Mario nhảy | MarioDead | Xử lý logic khi mario die |
| TextureRegion marioJump | Textute Mario nhảy | redefineMario | Khởi tạo lại các phương thức khi mario từ trạng thái lớn chuyển lại thành nhỏ |
| float stateTimer | Thời gian của các frame | draw | Vẽ mario |
|  |  | Các hàm hỗ trợ lập trình viên | Hỗ trợ lập trình viên thay đổi các thuộc tính của mario mà không cần quan tâm đến logic game |
| boolean isRight | Cờ hiệu để set vị trí Mario trong thế giới game |
| boolean isBig | Cờ hiệu để cho biết Mario đang lớn hay nhỏ |
| boolean isGrow | Cờ hiệu cho biết mario đang lớn lên để chạy ani lớn lên |
| boolean defineBigMario | Cờ hiệu để định nghĩa lại mario |
| ScreenManagement screen | Lưu trữ thông tin scene gọi vào |
| boolean isNextScene | Cờ hiệu để chuyển scene |
| Array<FireBall> fireballs | Lưu trữ danh sách fireball khi mario dùng fireball |

### Giao diện

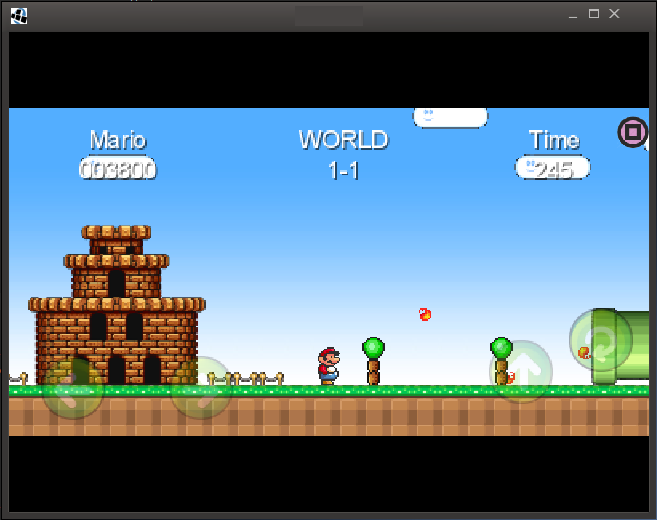
#### GameMenu



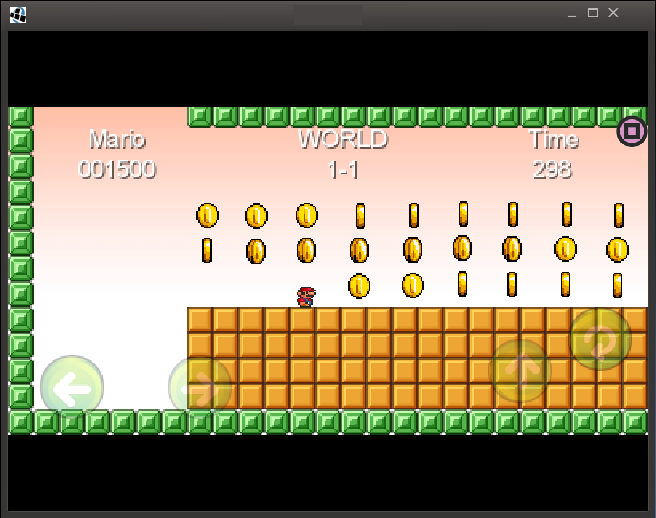
#### FirstScene







#### UnderGroundScene



#### SecondScene





# CÀI ĐẶT THỬ NGHIỆM

## Môi trường

Môi trường cài đặt và sử dụng: Android 5.0 trở lên.

## Nền tảng công nghệ

1. Platform: Android Studio
2. Framework: LibGdx
3. Ngôn ngữ : Java
4. Tool:

* MapEdiptor
* Gdx Texturepacker

## Thử nghiệm

1. Máy ảo:

* **Genymotion**
* **Máy ảo android**.

1. Máy thực: **Samsung S7 Edge**.

# KẾT LUẬN, HƯỚNG MỞ RỘNG

## Kết quả và đánh giá

* Hoàn thành được các yêu cầu cơ bản của một chương trình viết bằng bằng ngôn ngữ Java
* Tích hợp một số nguồn open source như github để tạo độ phong phú cho giao diện và xử lí. Tăng sự phong phú trong giao diện.
* Kích thước phần mềm được thiết kế có thể chạy đa dạng mà không bị lỗi trong hầu hết các màn hình điện thoại hiên nay.
* Phân chia công việc giữa các thành viên rõ ràng đạt hiệu quả tốt.

## Hạn chế

* Các màn chơi trong game còn khá đơn giản, tương đối giống với game Super Mario chưa có nhiều sáng tạo, chưa đủ tính hấp dẫn và tính ứng dụng vào thực tế còn chưa cao, khó có thể phổ biến ứng dụng do sử dụng nhiều hình ảnh cũng như nhân vật có tính bản quyền.
* Các tính năng của game còn tương đối sơ sài.
* Gặp khó khăn trong việc lập trình do công cụ hổ trợ lập trình, máy ảo tốn nhiều tài nguyên và tương đối nặng, đòi hỏi cấu hình của máy lập trình.

## Hướng phát triển

* Phát triển lên CH play ở phiên bản Android và AppStore ở phiên bản iOS.
* Kích thước phần mềm được thiết kế có thể chạy đa dạng mà không bị lỗi trong hầu hết các màn hình điện thoại hiên nay.
* Phát triển thêm nhiều màn chơi, cốt truyện hấp dẫn hơn, thêm hiệu ứng animation đặc biệt, cải thiện chất lượng của game sao cho tương thích với nhiều dòng máy, cũng như cấu hình hơn.
* Tạo thêm các tính năng cho game như đăng nhập bằng tài khoản cá nhân (Facebook, Gmail), cho phép người chơi cùng chơi trực tuyến với nhau, tạo tính cạnh tranh cho game.
* Phân chia công việc giữa các thành viên rõ ràng đạt hiệu quả tốt.

# TƯ LIỆU THAM KHẢO

[1] LibGDX

[**https://libgdx.badlogicgames.com/**](https://libgdx.badlogicgames.com/)

[2] Box2D with libgdx

[**https://github.com/libgdx/libgdx/wiki/Box2d**](https://github.com/libgdx/libgdx/wiki/Box2d)