**NGHIÊN CỨU REACT NATIVE ĐỂ XÂY DỰNG ỨNG DỤNG DI ĐỘNG ĐA NỀN TẢNG**

APPLICATION OF REACT NATIVE IN DEVELOPING MOBILE APPLICATION ON MULTIPLE OPERATION SYSTEMS

SVTH: Văn Hồng Xuân

Lớp 14IG, Khoa Công nghệ Thông tin, Trường Cao đẳng Công nghệ Thôn tin, Đại học Đà Nẵng, email xuanvan229@Gmail.com

GVHD: TS. Lê Văn Minh

Khoa Công nghệ Thông tin, Trường Cao đẳng Công nghệ Thôn tin, Đại học Đà Nẵng, email hiencit@gmail.com

**Tóm tắt -** ReactJS chỉ là một thư viện được viết bằng Javascript, giúp xây dựng giao diện người dùng (UI) web. Thì React Native là một framework giúp tạo ra những ứng dụng di động mà chỉ sử dụng JavaScript. React Native sử dụng thiếu kế tương tự như React, giúp bạn tạo ra những giao diện phong phú từ những thành phần được khai báo.Với React Native, bạn không xây dựng một Mobile Web App, hay một Hybrid App. Bạn sẽ xây dựng một ứng dụng di đông thật sự mà không thể phân biệt được từ một ứng dụng sử dụng Objective-C cho iOS hoặc Java cho Android.

**Từ khóa –** ReactJS, React Native, Javascript, Hybrid App, Mobile Web App , Objective-C, iOS, Java, Android.**Abstract –** ReactJS is just a library written in Javascript, which builds up the web user interface (UI). Then React Native is a framework for creating mobile applications that use only JavaScript. React Native uses the same design as React, which helps you create rich interfaces from the declared components. With React Native, you do not build a Mobile Web App, or a Hybrid App. You will build a truly portable application that can not be distinguished from an application that uses Objective-C for iOS or Java for Android

**Key words –** ReactJS, React Native, Javascript, Hybrid App, Mobile Web App , Objective-C, iOS, Java, Android.

# Đặt vấn đề

Mỗi hệ điều hành riêng biệt thì phải xây dựng ứng dụng trực tiếp bằng ngôn ngữ mà hệ điều hành đó cung cấp. Đối với android là sử dụng Java, hoặc đối với iOS là dùng Objective-C hay Swift. React Native là một framework cho phép các lập trình viên xây dựng các ứng dụng trên nền tảng Android, Windows và iOS sử dụng ngôn ngữ lập trình javascript nhưng mang lại trải nghiệm native app thực sự.

Với tư duy “học một lần, sử dụng nhiều nơi”, React Native mang lại sự lựa chọn mới hoàn toàn cho các lập trình viên muốn xây dựng ứng dụng di động. Các lập trình viên có sẵn kiến thức về javascript có thể vừa làm web, đồng thời xây dựng ứng dụng di động sử dụng javascript mà vẫn mang lại trải nghiệm native cho người dùng.

# Phương pháp nghiên cứu

## JavaScript là gì

JavaScript là một cross-flatform, ngôn ngữ kịch bản hướng đối tượng. Nó là một ngôn ngữ rất nhỏ và nhẹ.

JavaScript là ngôn ngữ kịch bản phía máy khách (client-side) và ở phía máy chủ (server-side với Nodejs). Do đó, nó được sử dụng trên hầu hết các trang web mà bạn nhìn thấy trên internet.

## ES6 là gì

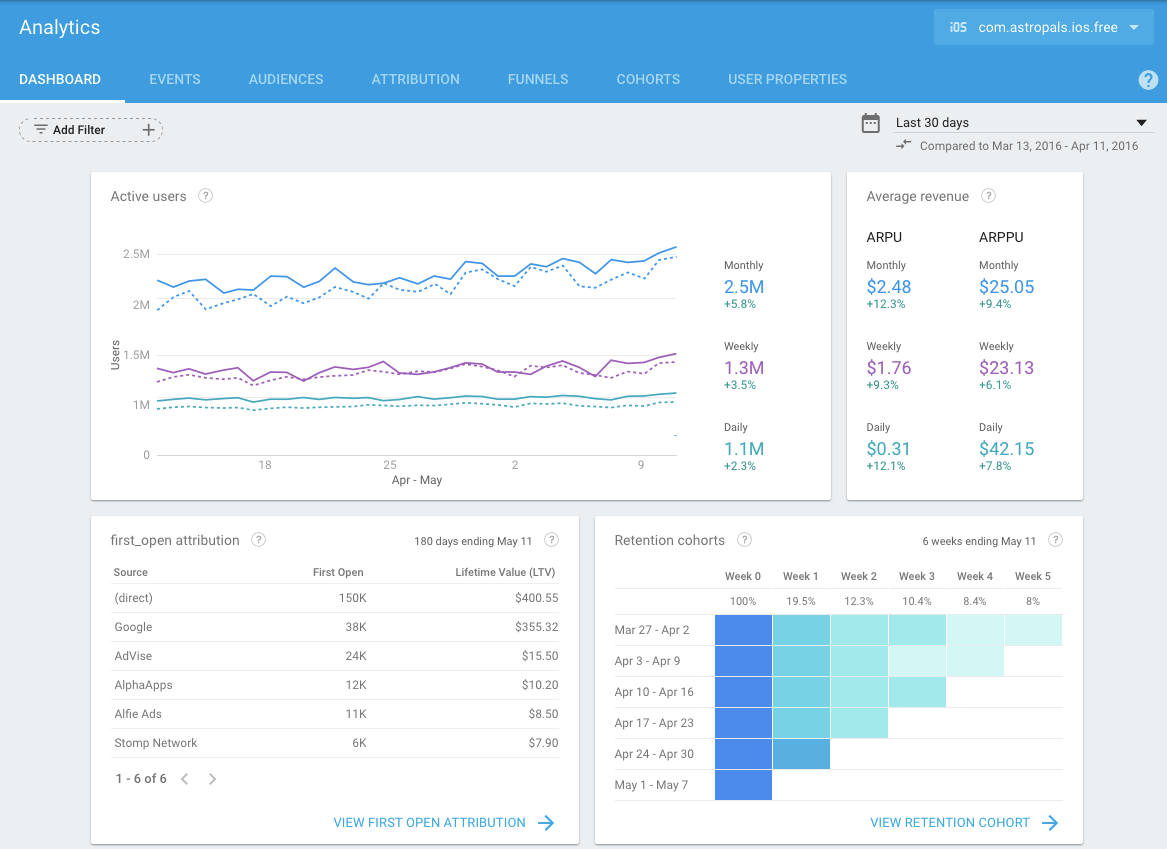
ES6 là chữ viết tắt của ECMAScript 6, đây được coi là một tập hợp các kỹ thuật nâng cao của Javascript và là phiên bản mới nhất của chuẩn ECMAScript. ECMAScript do hiệp hội các nhà sản xuất máy tính Châu Âu đề xuất làm tiêu chuẩn của ngôn ngữ Javascript. Bạn cứ nghĩ xem hiện nay có khá nhiều trình duyệt Browser ra đời và nếu mỗi Browser lại có cách chạy Javascript khác nhau thì các trang web không thể hoạt động trên tất cả các trình duyệt đó được, vì vậy cần có một chuẩn chung để bắt buộc các browser phải phát triển dựa theo chuẩn đó

## ReactJS là gì

Reactjs là một thư viện của javascript, dùng để xây dựng giao diện người dùng. Reactjs chỉ dùng để xây dựng giao diện người dùng, nghĩa là chỉ phục vụ tầng View trong mô hình MVC.React được xây dựng xung quanh các component, chứ không dùng template như các framework khác.Component cho phép chia giao diện ra thành từng thành phần, tái sử dụng những thành phần đó và lập trình các component độc lập.

## Firebase Database là gì

Firebase là một nền tảng di động giúp bạn nhanh chóng phát triển các ứng dụng chất lượng cao, phát triển ứng dụng cho người dùng lớn, và kiếm được nhiều tiền hơn.



Firebase Realtime Database cơ sở dữ liệu đám mây NoSQL đồng bộ hóa. Dữ liệu được đồng bộ hóa trên tất cả các client trong thời gian thực, và luôn có sẵn khi ứng dụng offline.

## React-Native

### React-Native là gì

React Native là một framework giúp tạo ra những ứng dụng di động mà chỉ sử dụng JavaScript.

React Native sử dụng thiếu kế tương tự như React, giúp bạn tạo ra những giao diện phong phú từ những thành phần được khai báo

Với React Native, bạn không xây dựng một Mobile Web App, hay một Hybrid App. Bạn sẽ xây dựng một ứng dụng di đông thật sự mà không thể phân biệt được từ một ứng dụng sử dụng Objective-C cho iOS hoặc Java cho Android.

React Native cho phép bạn xây dựng một ứng dụng một cách nhanh chóng. Thay vì biên dịch lại, bạn có thể reload lại ứng dụng của bạn ngay lập tức. Với reloading, bạn thậm chí có thể chạy lại đoạn code mới trong khi vẫn giữ được trạng thái ứng dụng của bạn.

React Native giống như ReactJS, nhưng nó sử dụng những thành phần riêng biệt thay vì các thành phần của trang web.

Để hiểu được cấu trúc cơ bản của React Native, bạn cần phải hiểu một số khái niệm của React, như JSX, state, props...

### Props and State

Có hai loại dữ liệu để điều khiển các thành phần (Component): props và state.

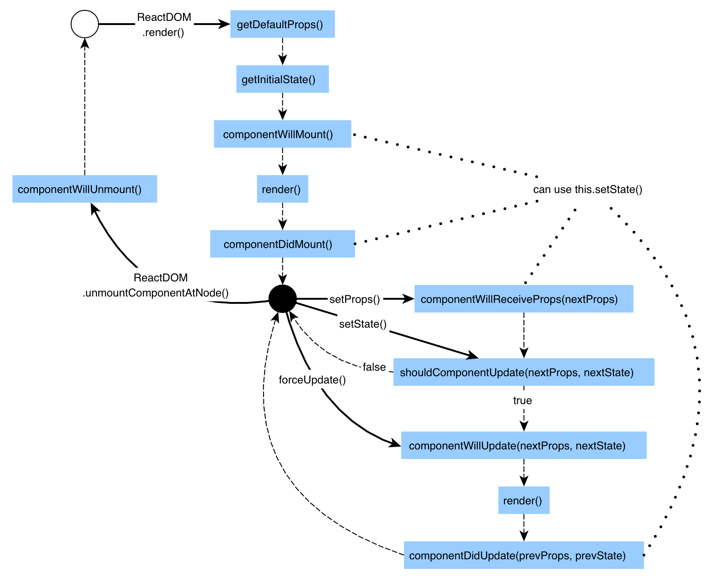
* Props được thiết lập bởi các thành phần cha và nó sẽ cố định trong suốt thời gian tồn tại của một thành phần
* Để dữ liệu có thể thay đổi đươc trong thành phần, chúng ta có thể dùng state

### Style

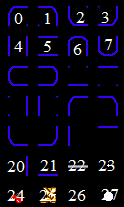
Với React Native, bạn không cần phải sử dụng một ngôn ngữ đặt biệt hoặc cú pháp nào để thiết kế giao diện cho ứng dụng cho bạn. Bạn chỉ cần sử dụng style của JavaScript. Tên style và giá trị thường gần giống với CSS ví dụ như backgroundColor chứ không phải là background-color.

Trong một Component, để thiết kế giao diện, bạn chỉ cần sử dụng StyleSheet.create để định nghĩa tại một ví trí.

### Vòng đời



* getDefaultProps() được gọi đầu tiên, nó được dùng để tạo những props để sử dụng trong Component. Có thể truy cập vào thành phần qua this.props
* getInitialState() là hàm để đặt giá trị state cho component. có thể truy cập vào những thành phần state này bằng this.state
  + Trong ES6 classes, bạn có thể định nghĩa state bằng cách gán giá trị cho this.state được định nghĩa bên trong hàm constructor
* componentWillMount() hàm này được gọi trước render() và sẽ thực hiện một số tác vụ. Hàm này sẽ thực hiện một lần duy nhất
* render() hàm này thực hiện việc render ra thành phần
* componentDidMount() hàm này thực hiện sau khi render. Nếu bạn cần load dữ liệu từ đâu đó về, thì đây là một nơi rất nhanh chóng cho việc đó. Gán các state trong hàm này sẽ gọi phương thức re-rendering
* componentWillUnmount() hàm này sẽ thực hiện một lần duy nhất, khi component sẽ unmount. Hàm này hữu dụng khi bạn cần xóa cá timer không còn sử dụng



**Hình 2:** Loại thành phần ứng với các hằng số

Sau đó chúng ta sẽ sử dụng những khối hình ảnh này để tạo nên bản đồ map thực sự như hình 3.



**Hình 3:** Loại thành phần ứng với các hằng số

Vậy thực tế bản đồ chỉ là một mảng những khối hình ảnh thuộc những loại khác nhau được đặt theo vị trí định sẳn để hiển thị nên bản đồ.

### Tạo bảng đồ

Mỗi khối hình ảnh sẽ là một đối tượng thuộc lớp Tile, có cấu trúc cơ bản bao gồm:

+ Các thuộc tính: vị trí, kích thước, loại.

+ Các hàm: khởi tạo và hiển thị.

Sau khi đã có lớp Tile chúng ta sẽ duyệt file map và tạo các Tile tại các vị trí định sẵn như trong file map quy định cùng với loại cụ thể của nó để tạo nên bảng đồ như hình 3 đã trình bày.

Cấu trúc hàm đọc file sẽ cơ bản như sau:

+ Mở file.

+ Đọc viên gạch đầu tiên.

+ Tạo Tile với thông tin đã lấy được.

+ Đưa vào mảng bản đồ.

+ Đọc viên gạch tiếp theo và bắt đầu lại vòng lặp trên cho tới khi hết file map.

## Xây dựng nhân vật Pacman

### Sự chuyển động của Pacman

Sau mổi vòng lặp nhân vật Pacman sẽ di chuyển một lượng pixel bằng với một hằng vận tốc định sẵn và dịch chuyển theo hướng dựa vào hướng điều khiển của người chơi.

Sự di chuyển này cơ bản sẽ như sau:

+ Nếu phím mũi tên trái đang được ấn tọa độ trục X của Pacman sẽ giảm một lượng bằng độ lớn của hằng vận tốc làm Pacman dịch sang trái. (di chuyển sang trái).

+ Ngược lại đối với phím mũi tên phải.

+ Đối với phím mũi tên lên và phím mũi tên xuống sẽ tương ứng với sự giảm và tăng tọa độ của Pacman theo trục Y.

Đó chỉ là sự chuyển động vật lý của Pacman, sự chuyển động của Pacman con được hiện thực một cách trực quan hơn qua việc hiển thị một chuỗi các hình ảnh liên tiếp để tạo nên sự chuyển động hình ảnh của Pacman:

Các chuổi hình ảnh tương ứng với sự chuyển động sẽ như sau:

+ Chuỗi ảnh chuyển động sang trái:

Description: D:\ntt\SDL2_2_0_3\images\PacmanRight.png

+ Chuỗi ảnh chuyển động sang phải:

Description: D:\ntt\SDL2_2_0_3\images\PacmanLeft.png

+ Chuỗi ảnh chuyển động lên trên:

Description: D:\ntt\SDL2_2_0_3\images\PacmanLeftToUp.png

+ Chuỗi ảnh chuyển động xuống dưới:

Description: D:\ntt\SDL2_2_0_3\images\PacmanRightToDown.png

+ Chuỗi ảnh chuyển động khi pacman die:

Description: D:\ntt\SDL2_2_0_3\images\pacmanDieSheet.png

### Xây dựng lớp PacMan

Đối tượng Pacman có cấu trúc như sau:

+ Các thuộc tính:

- Vị trí: lưu trử vị trí hiện tại của Pacman.

- Kích thước: kích thước của Pacman cho kiểm tra va chạm.

- Trạng thái di chuyển: cho biết hướng di chuyển hiện tại của Pacman.

- Trạng thái hiện tại của Pacman: các trạng thái như sống hay chết, trạng thái ăn được quả cầu sức mạnh hoặc ăn được dày tốc độ.

- Số mạng còn lại: lưu trử số mạng hiện tại của Pacman.

- Điểm số: số điểm mà Pacman đã kiếm được.

- Vận tốc hiện tại: đây sẽ chính là quãng đương mà Pacman sẽ di chuyển sau mỗi vòng lặp.

+ Các hàm:

- Khởi tạo: khởi tạo các tham số bang đầu cho Pacman như vị trí ban đầu, số mạng, điểm số, trạng thái hiện tại...

- Reset: được sử dụng khi Panman chết, hàm này sẽ khôi phục các thông số của Pacman như khởi tạo ban đầu ngoại trừ số điểm hiện tại và số mạng của Pacman sẽ bị giảm đi một.

- Kiểm tra sự kiện bàn phím: từ các sự kiện bàn phím phát hiện được trong hàm main ta sẽ truyền dữ liệu đó vào trong hàm này của Pacman từ đó thay đổi các trạng thái chuyển động cho phù hợp.

- Di chuyển: tùy thuộc vào trạng thái chuyển động hiện tại Pacman sẽ được di chuyển theo hướng phù hợp như đã trình bày trong phần chuyển động của Pacman. Ngoài ra trong hàm này ta sẽ kiểm tra va chạm của Pacman với các thành phần khác trên bản đồ từ đó đưa ra các sự kiện game phù hợp.

- Hiển thị: tùy thuộc vào trạng thái chuyển động hiện tại Pacman sẽ hiển thị những hình ảnh phù hợp với trạng thái chuyển động đó. Với mổi một vòng lặp khác nhau Pacman sẽ hiển thị một hình ảnh khác nhau trong chuổi hình ảnh theo đúng trật tự của nó để tạo ra sự chuyển động trực quan của mình.

## Xây dựng nhân vật Tự động

### Sự chuyển động của Tự động

Sau mổi vòng lặp nhân vật Tự động sẽ di chuyển một lượng pixel bằng với một hằng vận tốc định sẵn và dịch chuyển theo một hướng nhất định dựa vào thuật toán chuyển động của tự động.

Khi còn sống Tự động sẽ cố gắng di chuyển tới vị trí của Pacman còn khi bị giết sẽ di chuyển về vị trí bắt đầu của mình.

Sự chuyển động hình ảnh của Tự động tương tự như Pacman:

Chuỗi ảnh chuyển động sang trái:

Description: C:\Users\xv\Desktop\AI_2_1.png Description: C:\Users\xv\Desktop\AI_2_2.png

Chuỗi ảnh chuyển động sang phải:

Description: C:\Users\xv\Desktop\AI_2_2png.png Description: C:\Users\xv\Desktop\AI_2_1.png

Chuỗi ảnh chuyển động lên trên:

Description: C:\Users\xv\Desktop\AI_2_2.png Description: C:\Users\xv\Desktop\AI_2_1.png

Chuỗi ảnh chuyển động xuống dưới:

Description: C:\Users\xv\Desktop\AI_2_2.png Description: C:\Users\xv\Desktop\AI_2_1.png

Chuỗi ảnh chuyển động khi tự động die:

Description: D:\ntt\SDL2_2_0_3\images\augen_2.png Description: D:\ntt\SDL2_2_0_3\images\augen_1.png Description: D:\ntt\SDL2_2_0_3\images\augen_0.png Description: D:\ntt\SDL2_2_0_3\images\augen_3.png tương ứng cho chuỗi chuyển động sang trái, lên trên, sang phải và xuống dưới.

### Xây dựng lớp Tự động

Đối tượng Tự Động có cấu trúc như sau:

+ Các thuộc tính:

- Vị trí: lưu trử vị trí hiện tại của Tự động.

- Kích thước: kích thước của Tự động cho kiểm tra va chạm.

- Trạng thái di chuyển: cho biết hướng di chuyển hiện tại của Tự động.

- Trạng thái hiện tại của Tự động: các trạng thái như sống hay chết.

- Vận tốc hiện tại: đây sẽ chính là quãng đương mà Pacman sẽ di chuyển sau mỗi vòng lặp.

+ Các hàm:

- Khởi tạo: khởi tạo các tham số bang đầu cho Tự động như vị trí ban đầu, trạng thái hiện tại...

- Reset: được sử dụng khi Tự động chết và đã quay trở lại được vị trí ban đầu của mình, hàm này sẽ hồi sinh Tự động, khôi phục các thông số của Tự động như khởi tạo ban đầu.

- Di chuyển: tùy thuộc vào thuật toán chuyển động Tự động sẽ được di chuyển theo hướng phù hợp. Ngoài ra trong hàm này ta sẽ kiểm tra va chạm của Tự động với các thành phần khác trên bản đồ từ đó đưa ra các sự kiện game phù hợp.

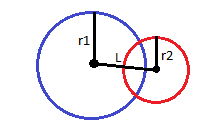
- Hiển thị: tùy thuộc vào trạng thái chuyển động hiện tại Tự động sẽ hiển thị những hình ảnh phù hợp với trạng thái chuyển động đó. Với mổi một vòng lặp khác nhau Tự động sẽ hiển thị một hình ảnh khác nhau trong chuổi hình ảnh theo đúng trật tự của nó để tạo ra sự chuyển động trực quan của mình.

## Một số thuật toán trong game

### Thuật toán kiểm tra va chạm giữa hai hình tròn

Kết quả của thuật toán kiểm tra va chạm cơ bản dựa trên sự so sánh giữa 2 hình tròn dựa vào khoảng cách hiện tại giửa hai tâm hình tròn với tổng bán kính của chúng:

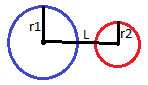
+ Nếu khoảng cánh giữa hai tâm của hình tròn nhỏ hơn tổng bán kính của chúng thì xảy ra va chạm.



**Hình 4:** Va chạm giữa hai hình tròn

(Nếu L < (r1+r2) thì xảy ra va chạm)

+ Ngược lại va chạm không xảy ra



**Hình 5:** Không va chạm giữa hai hình tròn

### Thuật toán kiểm tra va chạm giữa hai hình chử nhật

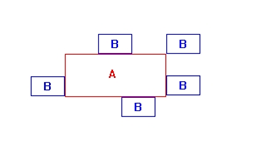
Thuật toán sẽ kiểm tra những trường hợp mà khi hai hình chử nhật thỏa những điều kiện này sẽ không va chạm với nhau:

- Cạnh đáy của hình chử nhật B nằm bên trên cạnh trên của hình chử nhật A.

- Cạnh phải của hình chử nhật B nằm bên trái cạnh trái của hình chử nhật A.

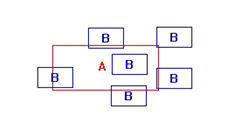
- Cạnh trái của hình chử nhật B nằm bên phải cạnh phải của hình chử nhật A.

- Cạnh trên của hình chử nhật B nằm bên bên dưới cạnh dưới của hình chử nhật A.



**Hình 6:** Không va chạm giữa hai hình chử nhật

Chỉ cần một trong các điều kiện này xảy ra hoặc cùng xảy ra thì không có va chạm xảy ra, ngược lại không có điều kiện nào thỏa mãn thì xãy ra va chạm.



**Hình 7:** Va chạm giữa hai hình chử nhật

# Kết luận

## Kết quả đạt được

Sau thời gian thực hiện, game đã được xây dựng hoàn thiện và đúng kịp hạn.

Game đã hoàn thành đúng như những kết quả đề ra ban đầu về giao diện cũng như các hoạt động của các đối tượng, các chức năng cơ bản nhất của game.

Ngoài ra game còn bổ xung thêm một số thứ: về giao diện bổ xung thêm HighSoccer và About, về sự kiện bổ xung thêm nhấn key “p” để pause game và bản đồ thì được thay đổi hoàn toàn khác sau mổi level.

Game chuyển động mượt mà và xử lý các sự kiện chính xác.

## Những khiếm khuyết

Game còn tương đối ít level và các chức năng còn hạn chế cần phải bổ xung thêm.

Tài liệu tham khảo

Wed hướng dẫn làm game với thư viện SDL [**http://lazyfoo.net/SDL\_tutorials/**](http://lazyfoo.net/SDL_tutorials/)Tài liệu tham khảo 2.

Wed hướng dẫn làm game với thư viện SDL **http://www.aaroncox.net/tutorials/**

Wed thông tin về thư viện SDL **https://wiki.libsdl.org/**

(BBT nhận bài: …/…/2014, phản biện xong: …/…/2014))

**Thông tin về tác giả**

|  |  |
| --- | --- |
| Hinh tác giả 1 | Nguyễn Trọng Công  - Điện thoại: 0965258940 |