ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN - BỘ MÔN THỊ GIÁC MÁY TÍNH & ĐIỀU KHIỂN HỌC THÔNG MINH

***

BÁO CÁO ĐỒ ÁN

***Môn*: Đồ họa ứng dụng**

Nội dung đồ án	**Tìm hiểu rendering engine PixiJS và demo với game Brick Out**	
***Nhóm thực hiện*:**	***MSSV***	***Họ & tên***
	18120515	**Vũ Hồng Phúc**
	19120716	**Lê Trọng Việt**

Mục lục

I. Tổng quan về PixiJS.

1) Tổng quát về PixiJS

- PixiJS là một công cụ hiển thị trên nền tảng web, cho phép sử dụng sức mạnh của webGL hoặc Canvas để render các nội dung lên màn hình một cách liên tục
- Cung cấp các API hỗ trợ tương tác như tương tác bằng chuột hay tương tác chạm vào màn hình
- Thường được sử dụng để làm các game 2D trên nền tảng web, trực quan hóa dữ liệu, phục vụ cho giáo dục, …
- Cũng có thể được sử dụng để làm các ứng dụng cho mobile hay desktop
- Tốc độ của PixiJS tương đối nhanh vì nó được tối ưu cho việc render các đối tượng 2D với webGL
- Việc triển khai sử dụng PixiJS tương đối đơn giản, chỉ cần có một trình duyệt là có thể sử dụng, khác với một số công cụ khác như Unity cần installer hoặc appstore, Flash thì cần player, …
- PixiJS chỉ thích hợp cho việc render đối tượng 2D, còn với các đối tượng 3D có các công cụ khác như BabylonJS hay ThreeJS
- Ngoài ra, PixiJS là một công cụ để render, nên nó không được sử dụng như là một Development Environment, Audio Library hay Data Store.

2) Các thành phần chính

Một số thành phần chính thường xuất hiện khi sử dụng PixiJS

a) Renderer

- Dùng để vẽ biểu đồ cảnh và tất cả các đối tượng, renderer mặc định của PixiJS dựa trên WebGL.
- Gồm nhiều thành phần quản lý cụ thể, các thành phần được thêm mặc định khi tạo Renderer như PIXI.ContextSystem để quản lý context, PIXI.EventSystem để quản lý các sự kiện, …
- Có thể tọa Renderer với cú pháp

RendererName = new PIXI.Renderer(Options)

Với các options là các tham số truyền vào như options.width, options.hight, options.view, …

- Hỗ trợ nhiều phương thức như registerPlugin để thêm một plugin, clear để dọn dẹp các đối tượng trong renderer, render để tiến hành kết xuất đối tượng.

b) Container

- Container như một thùng chứa để chứa tất cả các đối tượng con bên trong nó, ví dụ các đối tượng Graphics, hay Sprite, …
- Có thể tạo container với cú pháp

ContainerName = new PIXI.Container()

- Chứa các thuộc tính như: children, height, interactiveChildren, sortableChildren, sortDirty, width
- Container có các phương thức như addChild(children), addChildAt(child,index), getChildAt(index), render(renderer), …

c) Loader

- Cung cấp công cụ để tải lên các tài nguyên như hình ảnh, âm thanh để phục vụ cho ứng dụng
- Tạo một Loader:

loaderName = new PIXI.Loader (baseUrl, concurrency)

Với baseUrl là url cơ sở dẫn tới tài nguyên, concurrency là số tài nguyên được tải lên đồng thời

- Có 2 phương thức chính thường được sử dụng là **add(items)** để thêm các tài nguyên vào loader và **load()** để load các tài nguyên lên

d) Application

- Sử dụng để gom Loader, Ticker, Renderer thành một để dễ thao tác
- Tạo một Application

appName = **new PIXI.Application** (options)

- Có các thành phần như **stage** (container chứa các đối tượng), **ticker** (cập nhật các đối tượng), …
- Có các phương thức hay được sử dụng như **destroy(), render(), start(), stop(),...**

e) Các thành phần thường sử dụng khác

- Interaction: Hỗ trợ làm việc với các tương tác của người dùng như các tương tác bằng chuột hay chạm vào màn hình
- Ticker: Cập nhật và xử lý trạng thái của các đối tượng qua các vòng lặp
- Sprite: Đối tượng đơn giản và phổ biến nhất trong PixiJS, gồm một hình ảnh được render lên màn hình ví dụ như các con quái trong một trò chơi
- Graphic: Làm việc với các đối tượng đồ họa

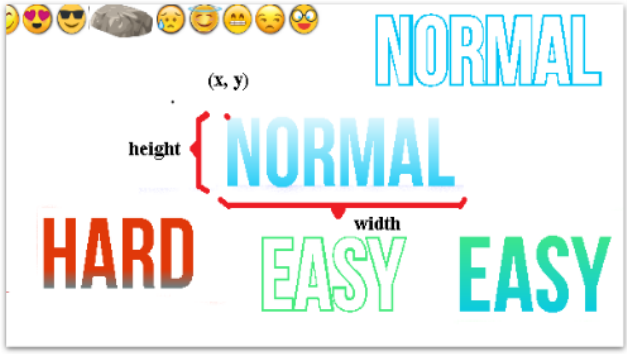
II. Demo game Brick Out.

1) Spirte và spirtesheet.

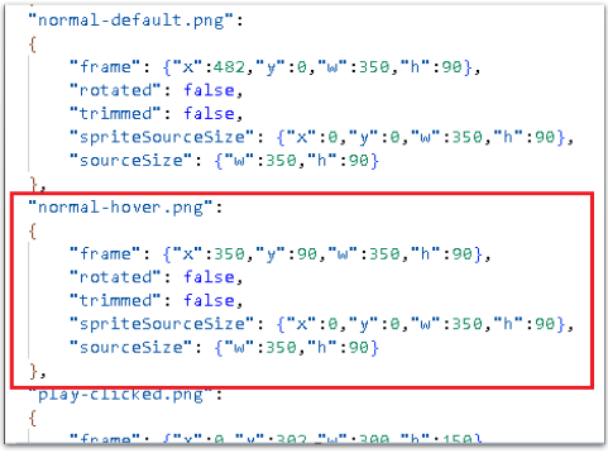
- Sprite là đối tượng 2D, thường được kết hợp với nhau tạo thành 1 phần chuyển động trong game, có thể dịch chuyển nó bằng cách đặt tọa độ trong hàm vẽ lên màn hình.

Aladdin Genesis sprite sheet

- Sprite là cách phổ biến để tạo ra các chuyển động lớn và phức tạp. Thực tế trong 1 game, thường có từ 10 đến hàng trăm Sprite, việc tải riêng từng ảnh sẽ tốn năng suất, bộ nhớ và mất nhiều thời gian để xử lý. Để quản lý các Sprite tốt hơn và tránh sử dụng quá nhiều hình ảnh người ta tạo ra sprite sheet.
- Sprite sheet là tập hợp của nhiều sprite bên trong nó, bao gồm các thông tin hỗ trợ định vị được các sprite trong sprite sheet, muốn vẽ được sprite trong sprite sheet cần các thông tin như: X, Y, WIDTH, HEIGHT để định vị sprite. Dưới đây là một phần spirtesheet của demo game Brick Out.



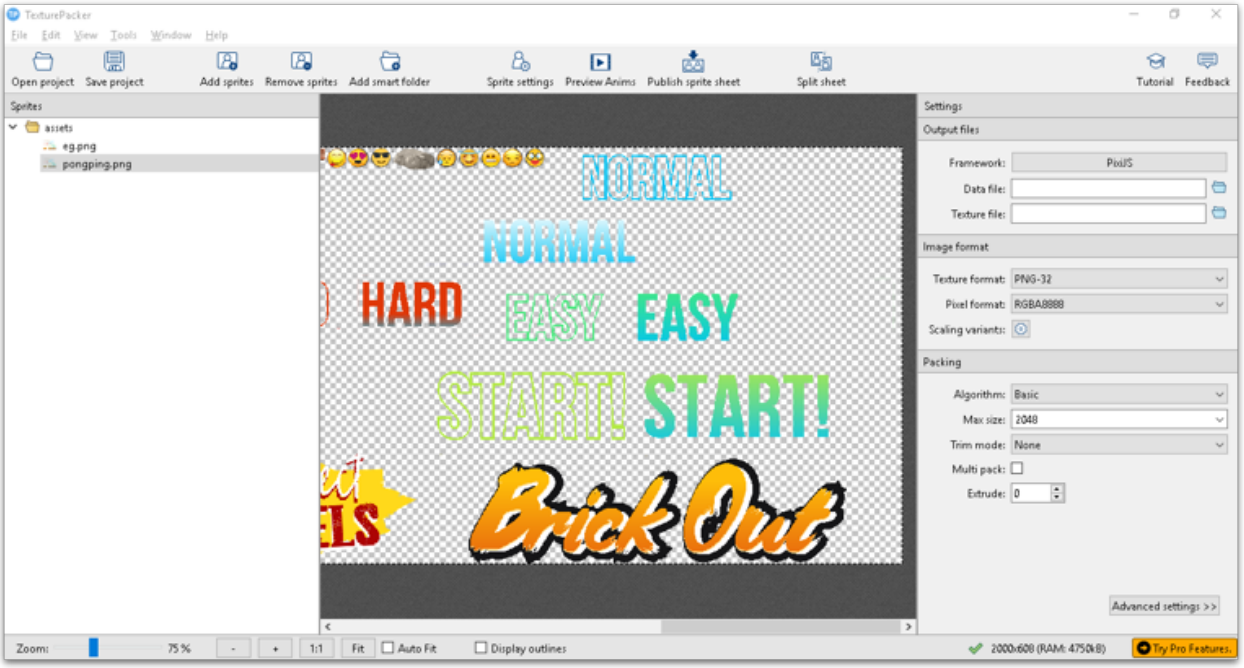
File ảnh các spirte



```
"normal-default.png":
{
    "frame": {"x":482,"y":0,"w":350,"h":90},
    "rotated": false,
    "trimmed": false,
    "spriteSourceSize": {"x":0,"y":0,"w":350,"h":90},
    "sourceSize": {"w":350,"h":90}
},
"normal-hover.png":
{
    "frame": {"x":350,"y":90,"w":350,"h":90},
    "rotated": false,
    "trimmed": false,
    "spriteSourceSize": {"x":0,"y":0,"w":350,"h":90},
    "sourceSize": {"w":350,"h":90}
},
"play-clicked.png":
{
```

File json chứa thông tin các spirte

- Có thể dùng phần mền **TexturePacker** để ghép nhiều ảnh 2D thành spirtesheet và kết xuất thông tin tọa độ ra file **json**.



2) Thư viện bump

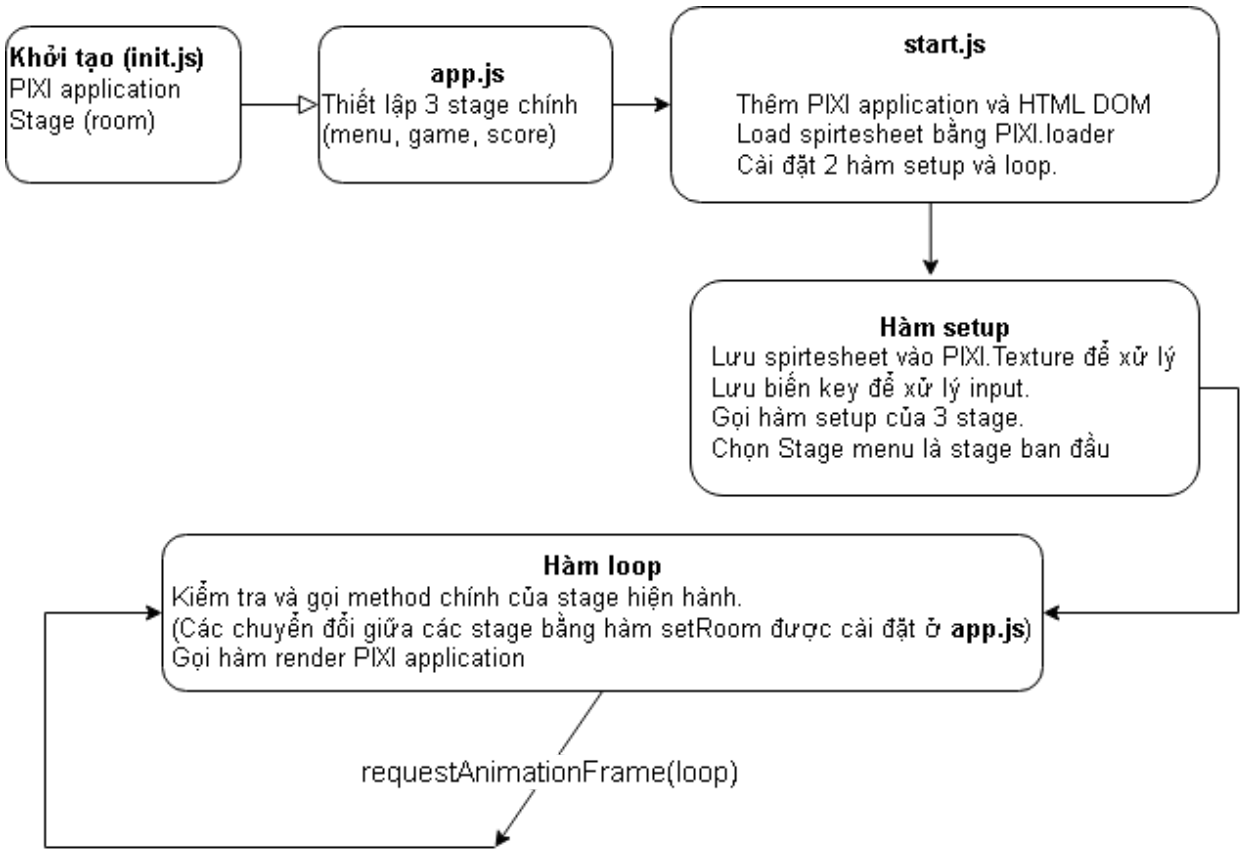
- Trong demo của game Brick Out có sử dụng một thư viện thứ 3 là Bump.
- Bump là thư viện hỗ trợ xử lý collision 2D cho PixiJS rendering engine.
- . Một số nhóm hàm trong Bump bao gồm:
  + Nhóm hàm **Hit**: kiểm tra các đối tượng có va chạm với nhau hay không
  + Nhóm hàm **Collision**: tính toán và tạo chuyển động nảy khi hai đối tượng va chạm.

3) Cấu trúc source code game.

Script	Mô tả thành phần chính
init.js	Biến **app:** dùng class **PIXI.Application** và thiết lập các thông số ban đầu**.** - **PIXI.Application** giúp khởi tạo ứng dụng Pixi ban đầu. - **PIXI.Application** tự động tạo các thành phần cơ bản như renderer, ticker và root container.
	Biến **room:** chứa thông tin và các hàm phụ trợ cho stage trong game. Mỗi stage sẽ được định nghĩa trong script **app.js**.
	Biến **style:** thiết lập style (font, font size, color, …) của text trong game, dùng class **PIXI.TextStyle.** - Một đối tượng **PIXI.TextStyle** chứa thông tin biểu diễn của đối tượng **PIXI.Text**. - Một đối tượng **PIXI.TextStyle** có thể được dùng chung cho nhiều đối tượng **PIXI.Text**.
	Hàm **setRoom:** - Để dịch chuyển giữa 3 stage.

	- Khi chuyển đổi giữa các stage trong game phải bật flag reset là true.
util.js	Class **Mathf:** class hỗ trợ tính toán ở game chính, gồm các hàm: - rad: chuyển đổi degree sang radian - deg: chuyển đổi radian sang degree - clamp: chuẩn hóa một value trong khoảng [min, max] - next: hỗ trợ chuẩn hóa giá trị tiếp theo khi nảy ra (bounce). - clamp360: chuẩn hóa một góc trong khoảng [0, 360],
	Class **Key:** class gồm các hàm xử lý input từ người dùng, dùng chủ yếu các thành phần trong **HTML DOM Events** và **EvenTarget interface** của Javasrcipt như - keyup - keydown - event.preventDefault - window.addEventListener
app.js	Ba stage chính của game được định nghĩa từ biến **room:** - Stage **room.menu:** menu của game, chọn độ khó. - Stage **room.game:** phần game chính - Stage **room.score:** hiển thị score sau game chính
start.js	- Thêm Pixi application vào HTML DOM. - Dùng **PIXI.loader** để tải trước các tài nguyên (spirtesheet) và gọi hàm setup.
	Hàm **setup:** - Lưu spirtesheet bằng đối tượng **PIXI.Texture** để ta có thể truy xuất các spirte trong quá trình xử lý. - Biến **key** để xử lý input của bàn phím, gồm 4 thành phần: a (left), d (right), shift (giảm tốc độ player), enter (chọn trong game). - Gọi hàm setup để cài đặt ba stage chính (menu, game, score). - Thiết lập stage hiện tại (ban đầu) là menu. - Chạy hàm **loop**.
	Hàm **loop:** - Kiểm tra và gọi method xử lý chính của stage hiện tại. - Render Pixi application đã được khởi tạo ở init.js. - Update frame bằng phương thức **requestAnimationFrame** gọi đệ quy lại hàm loop.

4) Diagram game Brick Out



5) Hình ảnh demo game Brick Out

- Menu (có thể dùng thao tác bằng chuột hoặc bàn phím)

- Game (**a** dịch trái, **d** dịch phải, **shift** giảm tốc độ player)

- Bảng điểm

III. Tài liệu tham khảo

1) PixiJS guide
2) PixiJS Github
3) Source code game tham khảo.
4) PIXI.Application
5) PIXI.TextStyle
6) HTML DOM Events
7) How to clamp a number.

_____***_____