# MÔ PHỎNG MÀN HÌNH RA-ĐA BẰNG PHƯƠNG PHÁP PHỦ ẢNH TRONG GPU

**Nguyễn Trung Kiên[a*], Trương Khánh Nghĩa[a], Nguyễn Thị Lan[a]**

*[a]Học viện Kỹ thuật Quân sự, Hà Nội, Việt Nam*

**Lịch sử bài báo**
Nhận ngày 07 tháng 01 năm 2017 | Chỉnh sửa ngày 10 tháng 04 năm 2017
Chấp nhận đăng ngày 08 tháng 05 năm 2017

**Tóm tắt**

*Ra-đa là một trong những thiết bị được sử dụng rộng rãi và tích hợp trong nhiều loại phương tiện, trang bị như máy bay, tàu thuyền cũng như trong các loại vũ khí, khí tài hiện đại. Vì thế, trong quá trình xây dựng các hệ thống mô phỏng phục vụ huấn luyện sử dụng các trang thiết bị, mô phỏng nội dung hiển thị trên màn hình ra-đa là rất cần thiết. Tuy nhiên, việc mô phỏng màn hình ra-đa thường phức tạp, khối lượng tính toán nhiều. Chính vì vậy, bài báo trình bày cách thức tiếp cận hiệu quả, sử dụng công nghệ lập trình trên card đồ họa để mô phỏng các nội dung, hiệu ứng trong quá trình hoạt động màn hình ra-đa. Phần lớn các tính toán sẽ được thực hiện trên bộ xử lý hiệu năng cao GPU của card đồ họa, dành thời gian xử lý trên CPU cho các tính toán khác do đó đảm bảo việc mô phỏng các màn hình ra-đa đáp ứng được yêu cầu hiển thị thời gian thực và là cơ sở để tích hợp với các nội dung mô phỏng khác trong hệ thống.*

**Từ khóa**: Lập trình GPU; Mô phỏng; Phủ ảnh; Ra-đa; Shader; Texture mapping.

## 1. ĐẶT VẤN ĐỀ

Sự phát triển mạnh mẽ của nền công nghệ điện tử, vi xử lý và công nghệ thông tin đã giúp cho công nghệ mô phỏng ngày càng được ứng dụng rộng rãi trong cuộc sống, đặc biệt trong huấn luyện, đào tạo giúp nâng cao hiệu quả, chất lượng đồng thời tiết kiệm chi phí. Trên các phương tiện như máy bay, tàu biển… cũng như các loại vũ khí, khí tài quân sự thì ra-đa là hệ thống hỗ trợ quan sát từ xa rất hiệu quả. Huấn luyện người học cách thức quan sát, nhận biết các loại thông tin hiển thị trên màn hình ra-đa là một trong các yêu cầu bắt buộc khi muốn thực hành sử dụng phương tiện. Tuy nhiên, trong thực tế thì không thể sử dụng ra-đa thật để đào tạo vì không thể dễ dàng đưa phương tiện ra các địa điểm huấn luyện: Di chuyển tàu thuyền, máy bay…. Trong quân sự, sử dụng đài ra-đa

* Tác giả liên hệ: Email: kiennt.simtech@mta.edu.vn

không những phức tạp, nguy hiểm mà còn ảnh hưởng đến an ninh quốc gia vì các đài phải thực hiện chế độ tắt, mở theo quy định. Không những thế, trong huấn luyện thường ngày thì không có tình huống, mục tiêu để thực hành. Chính vì vậy nhu cầu cần có thiết bị mô phỏng phục vụ huấn luyện sử dụng ra-đa rất cần thiết.

Blasband, Jorch, và Sigda (1998) đã đưa ra phương pháp mô phỏng ra-đa dựa trên các tính toán vật lý bằng cách giải quyết trực tiếp bài toán tính toán mức độ phản xạ điện từ của mục tiêu về ra-đa. Mahafza và Elsherbeni (2004) sử dụng mô phỏng trong MATLAB để hỗ trợ đào tạo học viên hiểu về quá trình thiết kế hệ thống xử lý tín hiệu trong ra-đa. Fabozzi, Holmberg, Duncan, Hancock, và McKay (2010) đưa ra mô hình tính toán hiệu năng cao cho việc xây dựng các ứng dụng mô phỏng ra-đa trong quân sự. Mchale (2015) giới thiệu công cụ mô phỏng ra-đa được phát triển bởi hãng Presagis phục vụ cho việc nghiên cứu và phát triển. Nhìn chung, sản phẩm mô phỏng của các hãng lớn trên thế giới đều được xây dựng công phu, sát thực tiễn nhưng tương ứng là giá thành rất cao và bí mật về công nghệ. Ở trong nước, một số trường đại học được trang bị các hệ thống mô phỏng ra-đa của nước ngoài như Trường Đại học Hàng hải, Học viện Hải quân… Mô phỏng ra-đa được ứng dụng nhiều trong quân sự nhất là trong Quân chủng Phòng không - Không quân, chủ yếu là do trong nước nghiên cứu phát triển (Cao, 2009; Học viện PKKQ, 2012). Tuy nhiên, các hệ thống này tập trung huấn luyện trình tự thao tác của học viên và chỉ mô phỏng về mặt nguyên lý các nội dung hiển thị của ra-đa do tính phức tạp trong việc tính toán thông tin hiển thị.

Trong phạm vi nghiên cứu, bài báo trình bày cách tiếp cận sử dụng công nghệ lập trình trên card đồ họa (GPU) để xử lý bài toán tính toán thông tin hiển thị trên màn hình ra-đa một cách hiệu quả, đáp ứng thời gian thực.

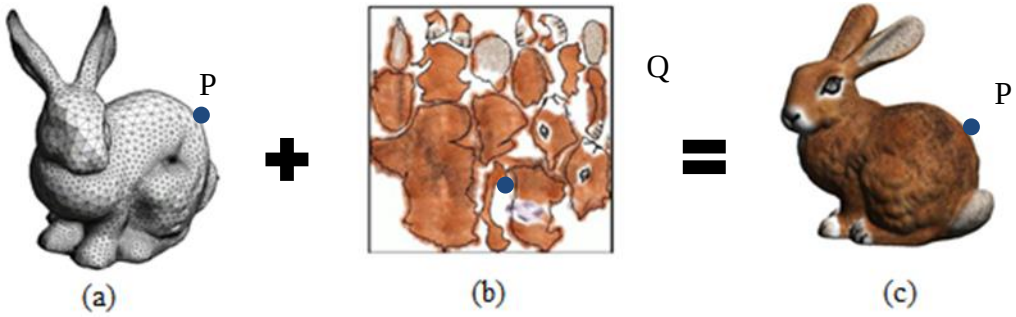
## 2. MÔ PHỎNG NỘI DUNG MÀN HÌNH RA-ĐA

### 2.1. Xử lý đồ họa trong GPU

Các card đồ họa hiện nay hầu hết đều cho phép khả lập trình, trong đó cho phép người dùng can thiệp vào một số bước trong quá trình xử lý thông tin trên card đồ họa. Các chương trình xử lý đó được gọi chung là *shader*, trong đó cơ bản nhất là chương

trình tính toán biến đổi tọa độ đỉnh (*vertex shader*) và chương trình xử lý phân mảnh (*fragment shader – Một* chương trình tính toán màu cho từng điểm thuộc đối tượng đang được vẽ).

Phủ ảnh (*texture mapping*) (Catmull, 1974) là một phương pháp ánh xạ thông tin của một ảnh lên bề mặt các đối tượng hình học ba chiều. Hình 1 mô tả phương pháp phủ ảnh, trong đó, khi hiển thị đối tượng ba chiều (Hình 1a), thì mỗi điểm P trên bề mặt đối tượng sau khi được phân mảnh hóa thì sẽ xác định được một điểm Q tương ứng trên ảnh phủ (Hình 1b) và màu sắc tại Q sẽ được gán cho điểm P (Hình 1c) và hiển thị ra màn hình.

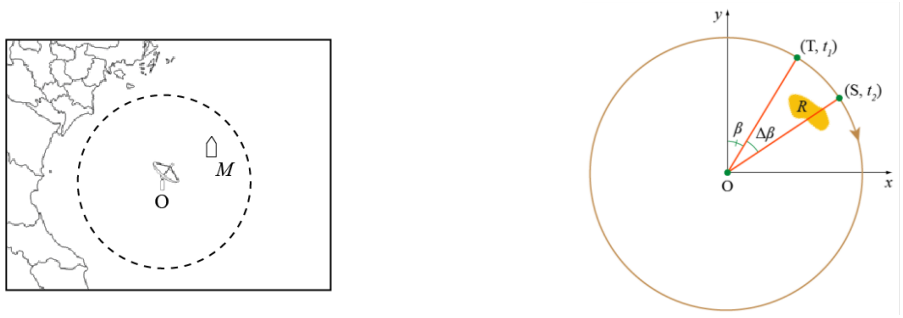



**Hình 1. Phủ ảnh trong GPU**
Ghi chú: (a) Bề mặt ba chiều đối tượng; (b) Ảnh phủ của đối tượng;
(c) Đối tượng được hiển thị trong không gian 3D
Nguồn: Matusik và Durand (2012)

Mục tiếp theo của bài báo sẽ trình bày cách thức sử dụng phương pháp phủ ảnh kết hợp với xử lý trong GPU để mô phỏng các nội dung cần hiển thị trên màn hình ra-đa.

## 2.2. Nội dung hiển thị trên màn hình

Giả sử trong thực tế, ra-đa gắn trên phương tiện đang ở tại vị trí O và một đối tượng M nằm trong phạm vi quét của ra-đa, minh họa trong Hình 2a. Khi đó, thông tin của đối tượng M hiển thị trên màn hình ra-đa không phải là hình dáng vật lý của M mà là một vùng thông tin đặc tả cho sự phản xạ năng lượng điện từ tại bề mặt M về đến ra-đa. Vùng thông tin này được minh họa là vùng *R* trong Hình 2b.




(a) (b)

**Hình 2. Ra-đa với đối tượng trong thực tế và quá trình hiển thị thông tin ra-đa**

Các thông tin quan sát được trên màn hình ra-đa bao gồm nhiễu địa vật, mục tiêu (máy bay, tàu thuyền…), và các loại nhiễu tạp do môi trường xung quanh tạo ra. Ngoài ra trên màn hình còn thể hiện điểm dấu cự ly, phương vị cũng như vị trí hiện thời của tia quét. Ta có thể sử dụng đa giác để đặc tả vùng thông tin quét được $R$ của ra-đa đối với đối tượng $M$. Dữ liệu đa giác và dữ liệu đặc tả hình quạt sẽ được chuyển vào trong GPU để tính toán vùng thông tin của đối tượng M cần được cập nhật tại vị trí hiện thời của tia quét.

## 2.3. Phạm vi cập nhật thông tin

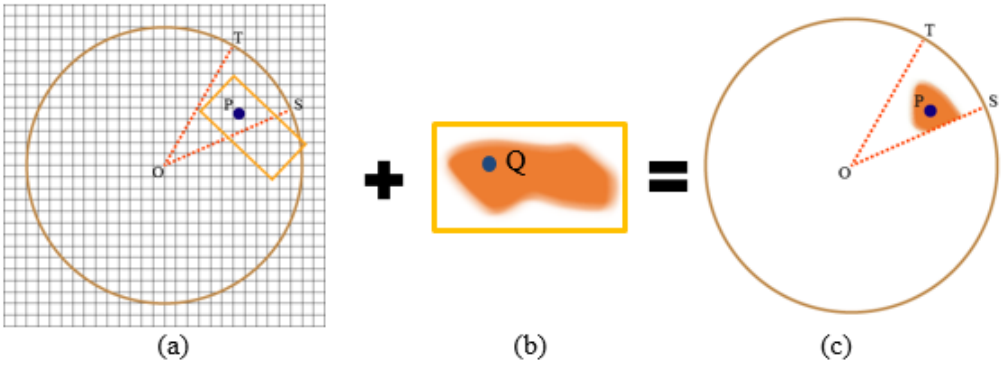
Giả sử tần số quét của ra-đa là $f$, khi đó vận tốc góc của tia quét sẽ là $\varpi = 2\pi f$. Chương trình trong máy tính sẽ được gọi lặp đi lặp lại để mô phỏng quá trình hiển thị thông tin của ra-đa. Gọi thời gian xử lý giữa hai lần gọi hàm hiển thị thông tin là $\Delta t$. Khi đó phạm vi cập nhật thông tin sẽ là hình quạt $TOS$ có góc $\Delta\beta = \varpi\Delta t$, minh họa trong Hình 2b.

Giả sử một phần vùng thông tin hiển thị $R$ của đối tượng $M$ nằm trong phạm vi cập nhật của ra-đa. Khi đó ta cần giải bài toán tìm vùng giao cắt $\Omega = TOS \cap R$. Thông thường, nếu ta mô tả vùng R dưới dạng một hình đa giác bất kỳ thì thuật toán giải bài toán giao cắt sẽ rất phức tạp. Một phương pháp thường dùng là xấp xỉ bằng tập hợp các điểm ngẫu nhiên nằm trong phạm vi $R$. Bài toán sẽ trở nên đơn giản hơn bằng việc quy về kiểm tra lần lượt từng điểm có nằm trong phạm vi hình quạt hay không. Với năng lực tính toán ngày càng cao của các chíp vi xử lý, đặc biệt là các chíp vi xử lý khả lập trình trên GPU, có thể cho phép đưa ra cách giải bài toán hiệu quả hơn.

# 3. XÂY DỰNG THUẬT TOÁN

## 3.1. Hiển thị thông tin đối tượng

Sử dụng thông tin đa giác để đặc tả cho vùng thông tin quét được của đối tượng cho phép hiển thị thông tin của đối tượng tương ứng với từng vị trí tia quét. Tuy nhiên, toàn bộ vùng thông tin hiển thị của đối tượng chỉ được thể hiện bằng một màu được định nghĩa trước. Trong thực tế thì tín hiệu trong vùng hiển thị thông tin này không đồng đều do đặc tính phản xạ năng lượng điện từ trên bề mặt đối tượng không đều. Thay vì số hóa vùng thông tin của đối tượng bằng một đa giác thì ta sẽ xấp xỉ vùng này bằng một hình chữ nhật và sử dụng dữ liệu vùng thông tin làm ảnh phủ cho hình chữ nhật đó. Sau khi đã xác định được điểm P thuộc đối tượng nằm trong phạm vi hình quạt thì từ điểm P sẽ xác định một điểm tương ứng Q trong ảnh phủ. Màu tại điểm Q trong ảnh phủ sẽ được gán cho P để hiển thị lên màn hình. Nguyên tắc xử lý thông tin đối tượng sử dụng ảnh phủ được minh họa trong Hình 3.

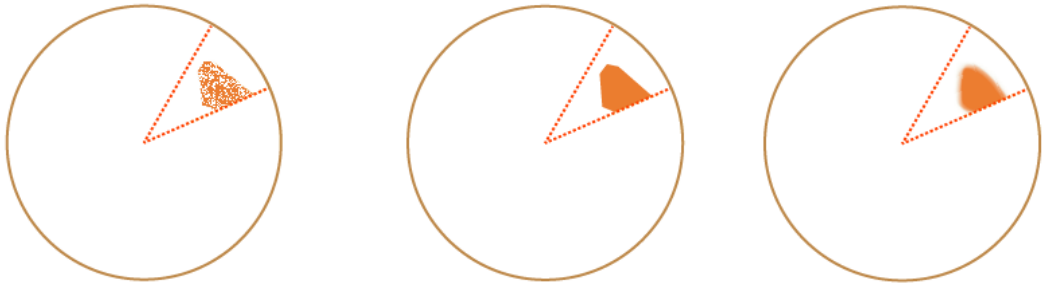



**Hình 3. Nguyên tắc xử lý thông tin đối tượng sử dụng ảnh phủ**

Ghi chú: (a) Ánh xạ thông tin hình bao của đối tượng trên màn hình; (b) Ảnh phủ vùng thông tin của đối tượng; (c) Kết quả hiển thị trên màn hình

Các thuật toán xử lý hoàn toàn trong CPU thường sử dụng một trong hai phương pháp mô tả phạm vi hiển thị của đối tượng đó là dùng tập điểm ngẫu nhiên (Học viện PKKQ, 2012) hoặc dùng đa giác (Cao, 2009). Sử dụng tập điểm ngẫu nhiên cho phép tính toán thuộc tính (màu sắc, độ sáng…) của từng điểm thuộc tập đó. Tuy nhiên, số lượng điểm càng lớn thì đặc tả đối tượng càng sát nhưng khối lượng tính toán càng nhiều không đáp ứng mô phỏng ra-đa trong thời gian thực. Nếu số lượng điểm ít thì việc hiển thị đối tượng lại bị rời rạc. Sử dụng đặc tả bằng đa giác thì sau khi giải bài toán giao cắt, toàn bộ

phần đa giác nằm trong phạm vi hình quạt sẽ có cùng thuộc tính, do đó hình ảnh hiển thị đối tượng sẽ dưới dạng cắt lớp, không sát thực tế. Với phương pháp sử dụng ảnh phủ, ta có thể đặc tả chi tiết thuộc tính của đối tượng (lưu trữ dưới dạng ảnh) và quá trình xử lý trong GPU cho phép tính toán thuộc tính từng điểm của đối tượng do đó kết quả hiển thị gần giống như trong thực tế. Hình 4 mô tả kết quả so sánh giữa các phương pháp.



(a) (b) (c)

**Hình 4. So sánh cách thức xử lý của một số phương pháp hiển thị**

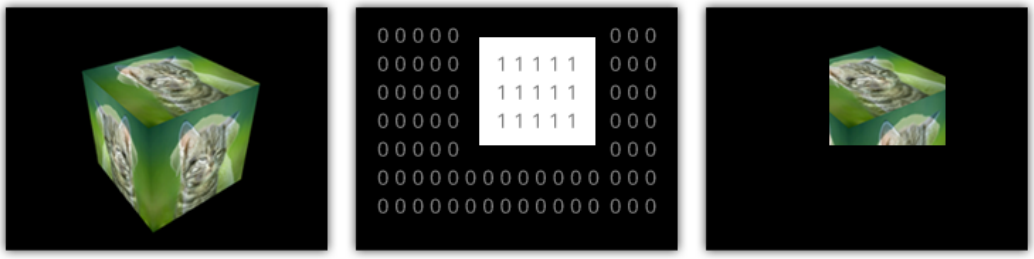
Ghi chú: (a) Tập điểm ngẫu nhiên; (b) Đa giác; (c) Ảnh phủ

### 3.2. Hiển thị dữ liệu nhiễu

Các dữ liệu nhiễu địa vật, nhiễu tạp có tác dụng trên toàn bộ màn hình ra-đa nên ta cũng có thể mô tả từng lớp nhiễu dưới dạng các ảnh phủ, trong đó sử dụng hình chữ nhật tương ứng là hình bao toàn bộ vùng hiển thị của ra-đa. Ta áp dụng phương pháp sử dụng ảnh phủ, mô tả trong Hình 3, để hiển thị dữ liệu nhiễu nằm trong phạm vi hình quạt. Thông tin màu của nhiễu sẽ được tổng hợp cùng thông tin của mục tiêu khi hiển thị ra màn hình.

### 3.3. Đối tượng chuyển động

Khi đối tượng chuyển động thì thông tin của đối tượng sẽ liên tục thay đổi cùng lúc với quá trình ra-đa hoạt động. Tuy nhiên thông tin này chưa được thể hiện lên màn hình ra-đa cho đến khi tia quét đi qua. Chính vì vậy ta phải giữ lại toàn bộ thông tin hiện có trên màn hình (thông tin đã tính toán) và chỉ cập nhật thông tin trong vùng hình quạt hiện thời của tia quét. Để đáp ứng yêu cầu đó ta sử dụng một bộ đệm đặc biệt trong GPU được gọi là *stencil buffer* (Rost và ctg., 2009). Bộ đệm này cho phép ta quyết định việc có hiển thị thông tin của điểm ảnh lên màn hình hay không, minh họa trong Hình 5.




**Hình 5. Mô tả bộ đệm *stencil buffer***
Nguồn: Rufinus và ctg. (2012)

Ta cần phải thiết lập giá trị của bộ đệm *stencil buffer* trước khi hiển thị thông tin mới của đối tượng. Khởi đầu, bộ đệm được gán giá trị không cho phép cập nhật toàn bộ màn hình. Sau đó ta thực hiện lệnh vẽ hình quạt *TOS* để xác định phạm vi cập nhật. Chỉ có các điểm nằm trong phạm vi hình quạt được gán giá trị cho phép cập nhật màn hình. Chính vì vậy, khi thực hiện các thuật toán hiển thị thông tin đối tượng, thông tin nhiễu… sau đó thì chỉ có thông tin tính toán trong phạm vi hình quạt mới được cập nhật và hiển thị lên màn hình.

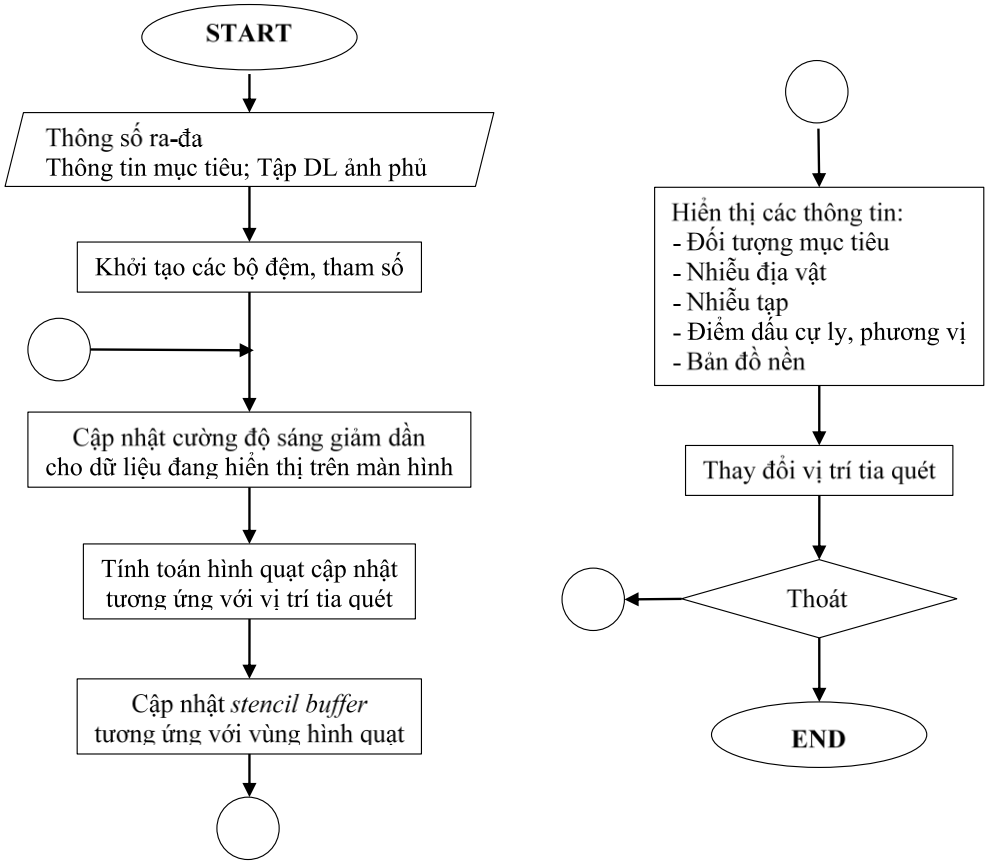
### 3.4. Hiệu ứng mờ dần

Đối với một số loại ra-đa sử dụng màn hình hiển thị tương tự, trên màn hình có phủ lớp bột phát quang. Sau khi xử lý, tùy theo mức độ tín hiệu phản xạ thu được mạnh hay yếu mà chùm tia điện tử bắn vào các điểm trên màn hình cũng có cường độ mạnh yếu khác nhau tạo nên độ sáng tương ứng. Tuy nhiên, các điểm sáng trên màn hình sẽ bị mờ dần do hiệu ứng lưu quang. Để mô phỏng hiệu ứng này, ta sử dụng hàm đặc trưng cho sự suy giảm cường độ sáng theo thời gian được bắt đầu từ khi tia quét đi qua. Gọi $T$ là thời gian lưu quang lớn nhất của mỗi điểm ảnh, $t$ là thời gian tính từ lúc điểm sáng có cường độ lớn nhất (tia quét đi qua): $t \in [0, T]$. Ta có thể xây dựng hàm suy giảm tuyến tính cường độ sáng bằng phương trình $f(t) = max(1 - \min(1, \frac{t}{T}), \varepsilon)$ trong đó, giá trị 1 đặc trưng cho cường độ sáng lớn nhất và giá trị $\varepsilon$ đặc trưng cho cường độ sáng yếu nhất. Để hiện thị hiệu ứng mờ dần thì ta phải áp dụng hàm tính toán suy giảm cường độ sáng này cho tất cả các điểm trên màn hình trước khi cập nhật dữ liệu mới tại phạm vi tia quét. Chính vì vậy ta coi toàn bộ dữ liệu đang hiển thị trên màn hình là một ảnh phủ và xử lý từng điểm

trên đó. Giá trị màu hiện tại của mỗi điểm được nhân tuyến tính với giá trị cường độ sáng tương ứng so với vị trí tia quét mới. Kết thúc quá trình này ta sẽ thu được kết quả là nội dung hiển thị trên màn hình đã bị giảm cường độ sáng đi tương ứng với khoảng thời gian dịch chuyển của tia quét.

### 3.5. Thuật toán

Thuật toán mô phỏng quá trình hiển thị nội dung trên màn hình ra-đa được mô tả trong Hình 6. Dữ liệu đầu vào gồm thông số của đài ra-đa, thông tin các mục tiêu, tập dữ liệu ảnh phủ các vùng thông tin hiển thị của các mục tiêu, tập dữ liệu ảnh nhiễu địa vật, nhiễu tạp…




**Hình 6. Thuật toán mô phỏng nội dung hiển thị trên màn hình ra-đa**

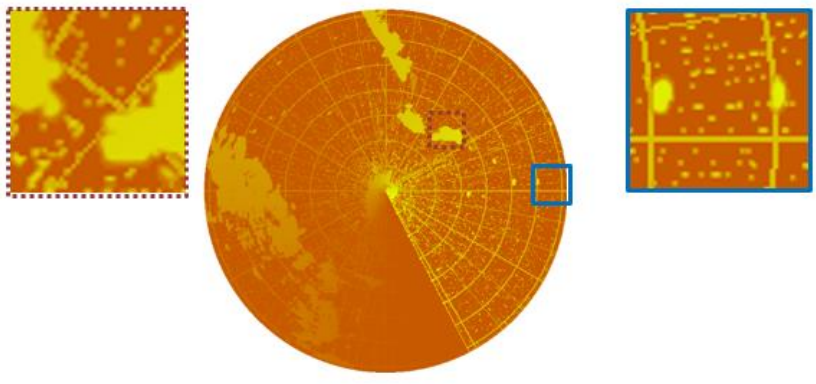
Do hiện tượng lưu quang (nếu có) nên ta phải cập nhật cường độ ánh sáng cho toàn bộ nội dung đang được hiển thị trên màn hình. Sau đó sử dụng *stencil buffer* để giới hạn phạm vi cập nhật thông tin chỉ trong vùng hình quạt của tia quét hiện thời. Sau khi

hiển thị xong các dữ liệu thì tia quét được cập nhật sang vị trí mới và quy trình xử lý được lặp lại.

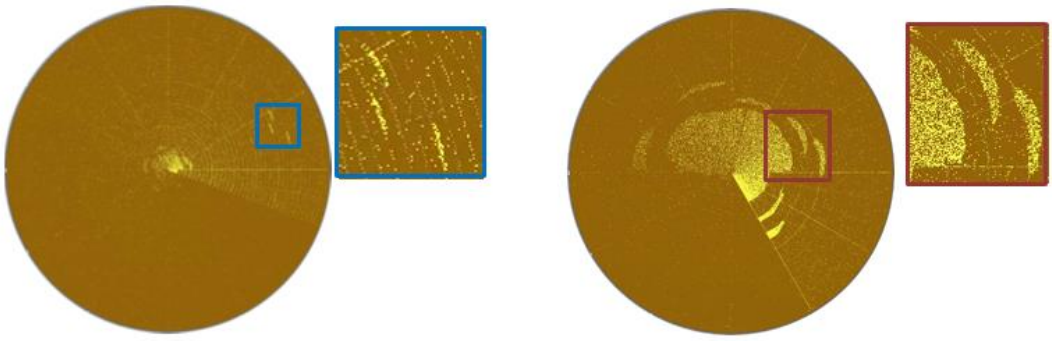
Với các thuật toán truyền thống xử lý tính toán trong CPU thì độ phức tạp của thuật toán phụ thuộc nhiều vào giải thuật giải quyết bài toán tìm giao cắt giữa hình quạt và các đối tượng. Với mỗi đối tượng ta cần xử lý lần lượt các điểm thuộc tập điểm mô tả đối tượng hoặc xử lý lần lượt các đỉnh thuộc đa giác mô tả đối tượng đó. Độ phức tạp của các thuật toán này khi đó là $O(n^2)$. Với thuật toán đề xuất, bài toán được đưa vào xử lý song song trong GPU, chính vì vậy độ phức tạp của thuật toán chỉ còn là O(n).

## 4. KẾT QUẢ MÔ PHỎNG

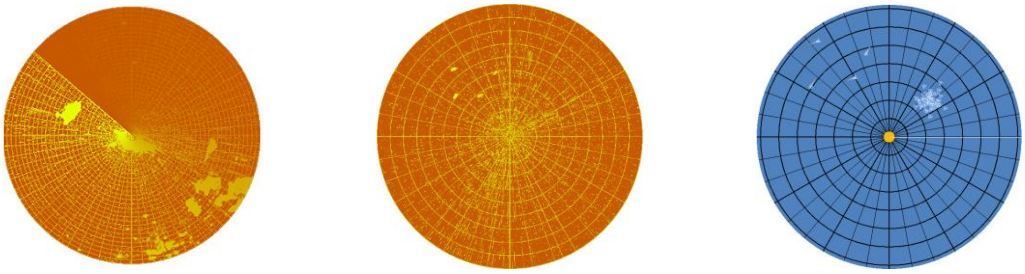
Các thử nghiệm được tiến hành trên máy tính có cấu hình Intel Core i7-6500U CPU 2.50GHz, RAM 4GB, card màn hình NVIDIA Geforce 930M, độ phân giải 1360x768. Hình 7 mô tả kết quả mô phỏng màn hình hiển thị ra-đa với thông tin nhiễu tạp, nhiễu địa vật, các đối tượng mục tiêu và có tích hợp hiệu ứng lưu quang. Hình 8 mô tả kết quả mô phỏng màn hình ra-đa bằng phương pháp xấp xỉ tập điểm đang được ứng dụng trong một số sản phẩm do Học viện PKKQ (2012) xây dựng. Hình 9 là kết quả mô phỏng một số chủng loại ra-đa khác nhau. Khi mô phỏng một loại ra-đa nào đó, dữ liệu đầu vào là tập các đối tượng mục tiêu có chủng loại khác nhau, bố trí ở các vị trí khác nhau, chuyển động theo vận tốc và hướng ngẫu nhiên. Thông tin nhiễu tạp, nhiễu địa vật được trích xuất ngẫu nhiên từ các tập dữ liệu có sẵn ứng với loại ra-đa đó.



**Hình 7. Mô phỏng ra-đa sử dụng phương pháp phủ ảnh, có hiệu ứng lưu quang**



**Hình 8. Mô phỏng ra-đa sử dụng tập điểm: Mục tiêu (trái); Nhiễu địa vật (phải)**



**Hình 9. Kết quả mô phỏng một số loại đài ra-đa khác nhau**

## 5. KẾT LUẬN

Trọng tâm của cách tiếp cận được trình bày trong bài báo này là sử dụng các thông tin, hình ảnh thực tế có được trên màn hình hiển thị các loại ra-đa trong thực tế. Khi thiết kế hệ thống mô phỏng huấn luyện cho loại ra-đa đó thì ta sẽ sử dụng các thông tin này để mô phỏng lại màn hình hiển thị. Với cách tiếp cận này thì ta có thể mô tả thông tin đối tượng đơn giản nhưng hiệu quả hơn. Sử dụng phương pháp phủ ảnh trong GPU ứng dụng trong mô phỏng màn hình ra-đa giúp giảm bớt các tính toán phức tạp liên quan đến xử lý tín hiệu trong ra-đa mà vẫn đảm bảo tính sát thực tế của hệ thống mô phỏng, thời gian tính toán ít hơn.

Trên cơ sở nghiên cứu đạt được, nhóm tác giả đã áp dụng vào mô phỏng một số hoạt động của đài ra-đa như P-18, P37, Kasta2E, 36D6… Sản phẩm đã được ứng dụng vào trong hệ thống mô phỏng xử lý thông tin ra-đa cảnh giới phục vụ cho huấn luyện đào tạo ở một số nhà trường, đơn vị trong quân đội. Hiện nay, nhóm nghiên cứu vẫn đang tiếp tục nghiên cứu, tích hợp bài toán mô phỏng ra-đa vào trong các hệ thống mô phỏng huấn luyện khác như huấn luyện chiến thuật không quân, mô phỏng bắn trên tàu hải quân…

## TÀI LIỆU THAM KHẢO

Blasband, C., Jorch, W., & Sigda, M. (1998). Physics-based radar simulation. *Military Aerospace Electronics*, *9*(11), 250-262.

Cao, H. T. (2009). *Xây dựng cơ sở dữ liệu hỗ trợ thiết kế các hệ thống mô phỏng bán tự nhiên*. Hà Nội, Việt Nam: Học viện Kỹ thuật Quân sự.

Catmull, E. E. (1974). *A subdivision algorithm for computer display of curved surfaces*. (Doctoral Thesis), The University of Utah. Retrieved from https://static1. 1.sqspcdn.com/static/f/552576/6419248/1270507173137/catmull_thesis.pdf

Fabozzi, D. J., Holmberg, C. M., Duncan, B., Hancock, R., & McKay, J. (2010). *High-performance computing interfaces for high-fidelity radar modeling*. Paper presented at the 2010 DoD High Performance Computing Modernization Program Users Group Conference, USA.

Học viện PKKQ. (2012). *Tổng hợp một số hệ thống mô phỏng huấn luyện kíp trắc thủ đài ra-đa P-18, Kasta-2E, 36D6*. Retrieved from http://hocvienpkkq.com

Mahafza, B. R., & Elsherbeni, A. Z. (2004). *MATLAB Simulations for radar systems design*. Florida, USA: Chapman & Hall/CRC Press.

Matusik, W., & Durand, F. (2012). *Texture mapping & sharers* [PowerPoint slides]. Retrieved from https:// ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphicsfall/2012/lecturenotes/MIT6_837F12_Lec16.pdf

Mchale, J. (2015). *Radar simulation tool for training and R&D announced by Presagis*. Retrieved from http://mil-embedded.com/news/radar-simulation-tool-for-training-and-rd-announced-by-presagis/

Rufinus, T., Engeström, E., Andrade, E. S., & Hamilton, A. (2012). *Depth and stencils*. Retrieved from https://open.gl/depthstencils

Rost, R. J., Licea-Kane, B., Ginsburg, D., Kessenich, J. M., Lichtenbelt, B., Malan, H., & Weiblen, M. (2009). *OpenGL Shading Language* (3rd ed.). Boston, USA: Addison-Wesley.

# RADAR SIMULATION USING GPU-BASED TEXTURE MAPPING

**Nguyen Trung Kien[a*], Truong Khanh Nghia[a], Nguyen Thi Lan[a]**

*[a]The Military Technical Academy, Hanoi, Vietnam*
*[*]Corresponding author: Email: kiennt.simtech@mta.edu.vn*

**Article history**
Received: January 07th, 2017 | Received in revised form: April 10th, 2017
Accepted: May 08th, 2017

**Abstract**

*Radar is widely used and integrated in many different kinds of weapons and equipment. During the design and development of a radar-based training system, the simulated operation of radar screen is really important. However, this work is complicated with heavy computation. Thus, this paper proposed a GPU-based method to simulate the contents and effects of a working radar screen. Most of the computation is performed in GPU so that the radar simulation can be run in real time and it can be integrated in a larger simulation system.*

**Keywords**: GPU programming; Radar; Shader; Simulation; Texture mapping.