TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

--------------------------------



BÀI TẬP LỚN MÔN: ĐỒ HỌA MÁY TÍNH

**DUCK SHOOTING**

**BÁO CÁO BÀI TẬP LỚN**

Ngành: Khoa Học Máy Tính

Lớp: DH18CS81

Sinh viên thực hiện:

Phạm Chí Năng – 1851010076

Bùi Phương Tùng – 1851010151

Trương Thiên Phong - 1851010098

Nguyễn Nguyệt Phương Bảo - 1851010013

G.V Hướng dẫn: Ths. Ncs Võ Thị Hồng Tuyết

Thành phố Hồ Chí Minh – Tháng 3 năm 2021

TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

--------------------------------



BÀI TẬP LỚN MÔN: ĐỒ HỌA MÁY TÍNH

**DUCK SHOOTING**

**BÁO CÁO BÀI TẬP LỚN**

Ngành: Khoa Học Máy Tính

Lớp: DH18CS81

Sinh viên thực hiện:

Phạm Chí Năng – 1851010076

Bùi Phương Tùng – 1851010151

Trương Thiên Phong - 1851010098

Nguyễn Nguyệt Phương Bảo - 1851010013

G.V Hướng dẫn: Ths. Ncs Võ Thị Hồng Tuyết

Thành phố Hồ Chí Minh – Tháng 3 năm 2021

**Lời mở đầu**

Đồ họa máy tính là một ngành khoa học Tin học chuyên nghiên cứu về các phương pháp và kỹ thuật có thể mô tả và thao tác trên các đối tượng của thế giới thực bằng máy tính.

Về bản chất: Đó là một quá trình xây dựng và phát triển các công cụ trên cả 2 lĩnh vực phần cứng và phần mềm hỗ trợ cho các lập trình viên thiết kế các chương trình có khả năng đồ họa cao.

Với việc mô tả dữ liệu thông qua các hình ảnh và màu sắc đa dạng của nó, các chương trình đồ họa thường thu hút người sử dụng, … Kích thích khả năng sáng tạo và nâng cao năng suất làm việc. Do vậy, đồ họa máy tính thường ứng dụng trong nhiều lĩnh vực.

Thuộc nhóm học phần chuyên ngành hướng Đồ họa máy tính, môn học cung cấp các kiến thức cơ bản về quy trình xử lý đồ họa trên máy tính và các giải thuật cơ bản trong đồ họa hai chiều và ba chiều.

Để học tập và làm việc được với đồ họa máy tính thì sinh viên bắt buộc phải đảm bảo đã học được các môn sau đây:

* Kỹ thuật lập trình.
* Cấu trúc dữ liệu và giải thuật.
* Lập trình hướng đối tượng.

Xác định được mục tiêu của môn học đối với sinh viên:

* Hiểu được phạm vi ứng dụng của đồ họa máy tính.
* Nắm vững các kiến thức đồ họa cơ bản: các thành phần, quy trình xử lý đồ họa trên máy tính.
* Hiểu và áp dụng được các thuật giải cơ bản của đồ họa 2 chiều và đồ họa 3 chiều.
* Vận dụng các giải thuật đồ họa để phát triển ứng các ứng dụng xử lý đồ họa.

Trong bài báo cáo này nhóm chúng tôi sẽ giới thiệu đến mọi người một trò chơi (game) được viết bằng ngôn ngữ C++ với Visual Studio 2010. Bên cạnh đó sẽ áp dụng đồ họa và mô phỏng vào trong game để cho mọi người thấy được rằng game có những thú vị của nó như thế nào, và lợi ích của việc học tập đối với việc đưa học tập vào bên trong game.

Chúng tôi muốn mang lại cho bạn đọc các cơ sở lý thuyết về những công cụ trên giao diện đồ họa của “graphics.h”. Tài liệu gồm 3 chương, trong đó chương thứ nhất sẽ giúp cho mọi người hiểu được cái nhìn khách quan về game mà chúng tôi xây dựng bởi thư viện “graphics.h”. Tại chương thứ 2, chúng tôi sẽ giới thiệu chi tiết về quy trình xây dựng game cũng như là những thuật toán từ cơ bản đến nâng cao trong việc hình thành nên game. Và chương cuối cùng, chúng ta sẽ thấy được kết quả mà chúng tôi đạt được, cùng với đó là những ưu điểm cùng với những khuyết điểm trong quá trình xây dựng game.

Với bố cục rõ ràng, hình ảnh phong phú, đa dạng. Dù cho bạn chưa từng biết về Đồ Họa Máy Tính thì ít nhất bạn cũng có thể hiểu được một phần nào những thứ mà chúng tôi tạo ra. Trong quá trình biên soạn, mặc dù đã có cố gắng hết sức nhưng vẫn khóc tránh khỏi những sai sót, rất mong quý thầy(cô) và quý bạn đọc có được những đóng góp chân thành để chúng tôi có thể phát triển thêm về sau này.

TRÂN TRỌNG CẢM ƠN

**MỤC LỤC**

[Chương 1. GIỚI THIỆU 1](#_Toc68984746)

[1.1. Giới thiệu đề tài: 1](#_Toc68984747)

[1.2. Mục tiêu đề tài: 2](#_Toc68984748)

[1.3. Giới hạn đề tài: 2](#_Toc68984749)

[1.4. Bố cục báo cáo: 2](#_Toc68984750)

[Chương 2. Game Duck Shooting with C 3](#_Toc68984751)

[2.1. Các kiến thức cơ bản đi kèm 3](#_Toc68984752)

[2.1.1. Thư viện “graphics.h” [1] 3](#_Toc68984753)

[2.1.2. Màn hình đồ họa [2] 3](#_Toc68984754)

[2.1.3. Màu trong “graphics.h” 4](#_Toc68984755)

[2.1.4. Khởi động chế độ đồ họa 4](#_Toc68984756)

[2.1.5. Một số lệnh cơ bản được sử dụng. 5](#_Toc68984757)

[2.1.6. Những hàm sử dụng trong “graphics.h”. [3] 5](#_Toc68984758)

[2.1.7. Thuật toán DDA (Digital Differential Analyzer) [4] 7](#_Toc68984759)

[2.1.8. Thuật toán Bresenham [4] 9](#_Toc68984760)

[2.1.9. Thuật toán MidPoint [4] 10](#_Toc68984761)

[2.1.10. Thuật giải MidPoint vẽ đường tròn [4] 11](#_Toc68984762)

[2.1.11. Giải thuật vẽ đường cong dựa trên thuật toán MidPoint. [4] 13](#_Toc68984763)

[2.1.12. Giải thuật tô màu. [4] 13](#_Toc68984764)

[2.1.12.1. Giới thiệu về tô màu. 13](#_Toc68984765)

[2.1.12.2. Tô màu theo từng điểm 14](#_Toc68984766)

[2.1.12.3. Tô màu theo dòng quét (ScanConvert) 16](#_Toc68984767)

[2.1.12.4. Tô màu theo đường biên (FloodFill) 16](#_Toc68984768)

[2.1.13. Tạo dựng và hiển thị đối tượng đồ họa 2D [5] 17](#_Toc68984769)

[2.1.13.1. Công cụ Turtle graphic 17](#_Toc68984770)

[2.1.13.2. Phép tịnh tiến (Translation) 18](#_Toc68984771)

[2.1.13.3. Phép co dãn/ tỉ lệ (Scale) 18](#_Toc68984772)

[2.1.13.4. Phép xoay (Rotation) 19](#_Toc68984773)

[2.1.13.5. Phép đối xứng 20](#_Toc68984774)

[2.1.14. Xử lý sự kiện chuột và bàn phím với thư viện BGI [6] 21](#_Toc68984775)

[2.1.14.1. Sự kiện chuột 21](#_Toc68984776)

[2.1.14.2. Sự kiện phím 22](#_Toc68984777)

[2.2. Thuật giải đề xuất 23](#_Toc68984778)

[2.2.1. Các ý tưởng đưa ra để giải quyết đề bài 23](#_Toc68984779)

[2.2.2. Giải quyết bài toán 24](#_Toc68984780)

[2.2.2.1. Xây dựng hàm Loading 24](#_Toc68984781)

[2.2.2.2. Xây dựng hàm vẽ các button đầu game 25](#_Toc68984782)

[2.2.2.3. Xử lý nhập tên trong game. 26](#_Toc68984783)

[2.2.2.4. Xử lý vấn đề điểm (High Score) 28](#_Toc68984784)

[2.2.2.5. Xử lý vấn đề vẽ vịt 31](#_Toc68984785)

[2.2.2.6. Xử lý vấn đề bắn vịt 32](#_Toc68984786)

[2.2.2.7. Các hàm hỗ trợ cho game 37](#_Toc68984787)

[2.3. Kết quả 37](#_Toc68984788)

[Chương 3. KẾT LUẬN 39](#_Toc68984789)

[3.1. Những kết quả đạt được 39](#_Toc68984790)

[3.2. Ưu điểm của đề tài 39](#_Toc68984791)

[3.3. Nhược điểm của đề tài 39](#_Toc68984792)

[3.4. Hướng phát triển trong tương lai 40](#_Toc68984793)

[Tài liệu tham khảo 40](#_Toc68984794)

**DANH MỤC HÌNH ẢNH**

[Hình 1 Mô tả quy tắc bàn tay trái 3](#_Toc68984795)

[Hình 2 Bảng màu 4](#_Toc68984796)

[Hình 3 Thuật toán DDA 7](#_Toc68984797)

[Hình 4 Lưu đồ thuật toán DDA 8](#_Toc68984798)

[Hình 5 Thuật toán Bresenham 9](#_Toc68984799)

[Hình 6 Lưu đồ thuật toán Bresenham 9](#_Toc68984800)

[Hình 7 Thuật toán MidPoint 10](#_Toc68984801)

[Hình 8 Thuật toán MidPoint 11](#_Toc68984802)

[Hình 9 Đối xứng các điểm thông qua đường tròn 12](#_Toc68984803)

[Hình 10 Đường tròn với điểm Q và điểm MidPoint 12](#_Toc68984804)

[Hình 11 Lưu đồ giải thuật vẽ đường cong dựa trên thuật toán MidPoint 13](#_Toc68984805)

[Hình 12 Đa giác nội tiếp Hình Chữ Nhật 14](#_Toc68984806)

[Hình 13 Đa giác có 13 đỉnh 15](#_Toc68984807)

[Hình 14 Tô màu theo từng dòng quét 16](#_Toc68984808)

[Hình 15 Lưu đồ tô màu theo từng dòng quét 16](#_Toc68984809)

[Hình 16 Tô màu theo đường biên 17](#_Toc68984810)

[Hình 17 Phép biến đổi tịnh tiến từ P thành Q đối với một điểm 18](#_Toc68984811)

[Hình 18 Phép co dãn P thành Q với một tỉ lệ nhất định 19](#_Toc68984812)

[Hình 19 Phép xoay tại gốc tọa độ 20](#_Toc68984813)

[Hình 20 Phép quay tại một điểm bất kì 20](#_Toc68984814)

[Hình 21 Giao diện Game Loading 24](#_Toc68984815)

[Hình 22 Button Play và Button Quit cho người dùng lựa chọn 25](#_Toc68984816)

[Hình 23 Giao diện sau khi chạy xong Game Loading 26](#_Toc68984817)

[Hình 24 Lưu đồ thể hiện quá trình nhập tên từ bàn phím 28](#_Toc68984818)

[Hình 25 Giao diện của High Score trước khi có điểm 30](#_Toc68984819)

[Hình 26 Giao diện của High Score 31](#_Toc68984820)

[Hình 27 Chú vịt trước và sau khi được xử lý 32](#_Toc68984821)

[Hình 28 Đàn vịt dự kiến sau khi qua xử lý 3D 32](#_Toc68984822)

[Hình 29 Giao diện Round 1 34](#_Toc68984823)

[Hình 30 Giao diện Round 2 35](#_Toc68984824)

[Hình 31 Giao diện Round 3 36](#_Toc68984825)

[Hình 32 Giao diện kết thúc game 36](#_Toc68984826)

**DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| STT | Ký hiệu chữ viết tắt | Chữ viết đầy đủ |
| 1 | DDA | Digital Differential Analyzer |
| 2 | BGI | Borland Graphics Interface |

# GIỚI THIỆU

## Giới thiệu đề tài:

Với đề tài “Xây dựng trò chơi bắn vịt trời. Với các hình vẽ được vẽ từ graphics.h” đã đưa chúng tôi đến với trò chơi Duck Hunt nổi tiếng, và từ đó chúng tôi đã cho ra mắt một sản phẩm mang tên DUCK SHOOTING.

Duck Hunt là một trò chơi điện tử bán súng hạng nhẹ năm 1984 do Nintendo phát triển và xuất bản cho hệ máy trò chơi điện tử Nintendo Entertainment System (NES). Trò chơi được phát hành lần đầu tiên tại Nhật Bản vào tháng 4 năm 1984, sau đó là một cổng trò chơi arcade được phát hành cho Nintendo Vs.

Trong Duck Hunt, người chơi sử dụng NES Zapper kết hợp với TV CRT để bắn những con vịt xuất hiện trên màn hình. Những con vịt xuất hiện một hoặc hai con cùng một lúc, và người chơi được cấp ba phát súng để bắn hạ chúng. Người chơi nhận được điểm khi bắn mỗi con vịt. Nếu người chơi bắn đủ số lượng vịt cần thiết trong một vòng duy nhất, người chơi sẽ tiến vào vòng tiếp theo; nếu không, người chơi sẽ nhận được một trò chơi kết thúc.

Duck Hunt là một trò chơi bắn súng trong đó mục tiêu là bắn các mục tiêu đang di chuyển trên màn hình TV khi đang bay. Trò chơi được chơi từ góc nhìn thứ nhất và yêu cầu khẩu súng hạng nhẹ NES Zapper, thứ mà người chơi nhắm tới và bắn vào màn hình. Mỗi vòng bao gồm tổng cộng mười mục tiêu để bắn. Tùy thuộc vào chế độ trò chơi mà người chơi chọn trước khi bắt đầu chơi, một hoặc hai mục tiêu sẽ xuất hiện trên màn hình vào bất kỳ thời điểm nào và người chơi có ba lần cố gắng bắn trúng chúng trước khi chúng biến mất.

Người chơi được yêu cầu bắn thành công một số lượng mục tiêu tối thiểu để tiến vào vòng tiếp theo; thất bại sẽ dẫn đến một trò chơi kết thúc. Độ khó của trò chơi Duck Hunt tăng lên khi người chơi tiến lên các vòng cao hơn; mục tiêu sẽ di chuyển nhanh hơn và số lượng mục tiêu tối thiểu để bắn sẽ tăng lên. Người chơi nhận được điểm khi bắn một mục tiêu và cũng sẽ nhận được điểm thưởng khi bắn tất cả mười mục tiêu trong một vòng duy nhất.

## Mục tiêu đề tài:

* Viết được game Duck Shooting bằng ngôn ngữ C++ thông qua phần mềm Visual Studio 2010.
* Hiểu được thư viện “graphics.h”.
* Vận dụng các kiến thức về thuật toán có sẵn để vẽ những hình có trong game.
* Giúp cho người học và người tìm hiểu về C++ có thêm nhiều kiến thức về việc lập trình game, học tập và nghiên cứu về Đồ họa Máy Tính nói chung và thư viện “graphics.h” nói riêng.

## Giới hạn đề tài:

Vì đây là một đề tài mở nên hầu hết không bị ràng buộc về mọi thứ, nhưng cũng có những lưu ý về đề tài:

* Đề tài viết bằng ngôn ngữ C++
* Được mở rộng các loại thư viện phục vụ cho đề tài nhưng không thể thiếu thư viện “graphics.h”
* Các hàm vẽ cơ bản như: line, circle, … không được sử dụng trong đề tài vì đó là những hàm.
* Thay vào đó muốn vẽ được hình phục vụ cho đề tài thì phải vẽ bằng những thuật toán đã học. Chẳng hạn như: DDA, MidPoint, …

## Bố cục báo cáo:

Để thực hiện cho báo cáo một cách đầy đủ và hiệu quả thì không thể thiếu được những thứ sau:

* Một văn bản WORD báo cáo chi tiết về nội dung và quy trình làm việc.
* Một bản thuyết trình PowerPoint đề trình bày về nội dung một cách sống động hơn.
* Bên cạnh đó là toàn bộ Project hoàn thiện nhất mà tại đó có add tất cả các file để phục vụ cho đề tài.

# Game Duck Shooting with C

## Các kiến thức cơ bản đi kèm

### Thư viện “graphics.h” [1]

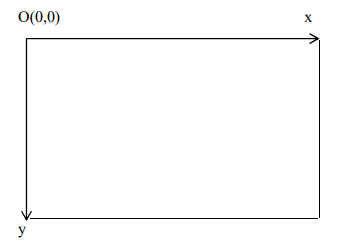
“graphics.h” hay tên chính xác và đầy đủ là Borland Graphics Interface (BGI) là một thư viện đồ họa rất phổ biến trên DOS và các máy tính chạy hệ điều hành Windows thời kỳ đầu như Windows 95, Windows 98, …

Thư viện này cung cấp cho người dùng 2 file: “graphics.h” và “graphics.lib” để có thể sử dụng được với C/C++ cũng như module graph nếu người dùng sử dụng ngôn ngữ Pascal. Bộ thư viện này đi kèm với IDE Borland C++ 3.1 (1992).

Một trong những điểm mạnh của thư viện này là việc khởi tạo cũng như sử dụng rất đơn giản, vì vậy dù ra đời rất lâu nhưng hiện tại vẫn có rất nhiều trường đại học sử dụng cho mục đích giảng dạy nói chung và Trường Đại học Mở Thành phố Hồ Chí Minh nói riêng.

### Màn hình đồ họa [2]

Theo Visual Studio thì màn hình đồ họa có trong thư viện “graphics.h” sẽ được xét theo quy tắc bàn tay trái. Và có dạng như sau:



Hình 1 Mô tả quy tắc bàn tay trái

Màn hình đồ họa trên trong Turbo C thường là (640,480). Như vậy thì x sẽ chạy từ 0 -> 639 và y chạy từ 0 -> 479.

### Màu trong “graphics.h”

Có tất cả là 16 màu:



Hình 2 Bảng màu

Dãy màu bắt đầu từ 0 và các số bên cạnh tương đương với mã màu mà “graphics.h” sử dụng.

### Khởi động chế độ đồ họa

Khai báo thư viện:

#include “graphics.h”

Gọi thư viện để sử dụng:

#pragma comment (lib, “graphics.lib”)

Câu lệnh chúng ta thường dùng trong lập trình:

int gd = DETECT, gm;

void initgraph(&gd, &gm, “c:\\tc\\bgi”);

Trong đó thì:

* gd viết tắt của graphdriver.
* gm viết tắt của graphmode.

Tùy vào graphdriver mà sẽ có graphmode khác nhau.

|  |  |  |
| --- | --- | --- |
| graphdriver | graphmode | Độ phân giải |
| VGA | VGALO | 640 x 200 pixel |
| VGAHI |  | 640 x 480 pixel |
| DETECT | Tự xác định chế độ đồ họa | Độ phân giải cao nhất để gán cho graphmode |

### Một số lệnh cơ bản được sử dụng.

Sau đây là một số lệnh cơ bản luôn được sử dụng trong chế độ đồ họa “graphics.h”

|  |  |
| --- | --- |
| Dừng chế độ đồ họa | void closegraph(); |
| Trả về mã lỗi khi khởi động chế độ đồ họa | int graphresult(); |
| Tô điểm có tọa độ x,y với màu “color” tùy chỉnh theo người dùng | putpixel(int x, int y, int color); |

### Những hàm sử dụng trong “graphics.h”. [3]

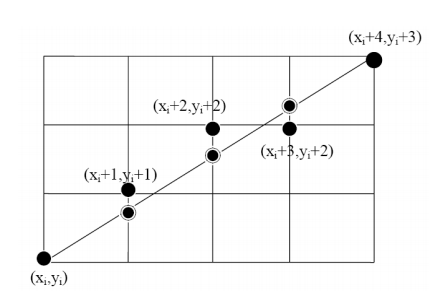
Đây là bảng thể hiện các hàm:

|  |  |
| --- | --- |
| Tên hàm | Định nghĩa/ Giải thích |
| getmaxx() | Tọa độ x lớn nhất |
| getmaxy() | Tọa độ y lớn nhất |
| getx() | Vị trí x hiện tại của con trỏ |
| gety() | Vị trí y hiện tại của con trỏ |
| setbkcolor(màu nền ) | Đặt màu nền: 0 - 15 |
| setcolor(màu vẽ) | Đặt màu vẽ: 0 - 15 |
| getbkcolor() | Màu nền hiện tại |
| getmaxcolor() | Số màu tối đa của màn hình |
| getcolor() | Màu vẽ hiện tại của người dùng |
| putpixel(x, y, c) | Vẽ điểm có tọa độ tại (x, y) và màu cố định c |
| getpixel(x, y) | Trả lại màu tại điểm (x, y) |
| line(x, y, x1, y1) | Vẽ đoạn thẳng đi qua (x, y) và (x1, y1) |
| lineto(x, y) | Vẽ đoạn thẳng đi qua vị trí hiện tại của con trỏ tới điểm (x, y) |
| rectangle(x1, y2, x2, y2) | Vẽ hình chữ nhật rỗng |
| bar(x1, y2, x2, y2) | Vẽ hình chữ nhật đặc Vẽ hình chữ nhật đặc |
| setlinestyle(kiểu đường, mẫu tô, độ đậm) | + Kiểu đường từ 0 -> 4  0: đường đặc  1: đường chấm  2: đường gạch  3: đường gạch dài  4: đường tự tạo  + Mẫu tô: Chỉ có tác dụng khi kiểu đường là 4, ta dùng 2 byte để định nghĩa  + Độ đậm là 1 hoặc 3 1: nét vẽ bình thường 3: nét vẽ đậm |
| setfillstyle(mẫu tô, màu tô) | + Mẫu tô: từ 0 -> 12  + Màu tô: từ 0 -> 15 |
| drawpoly(mảng số nguyên chứa tọa độ các điểm,số cặp điểm) | Vẽ đa giác rỗng |
| fillpoly(mảng số nguyên chứa tọa độ các điểm,số cặp điểm) | Vẽ đa giác đặc |
| arc(x, y, góc đầu, góc cuối, bán kính) | Vẽ cung tròn có tâm (x, y) với các góc và bán kính tương ứng. |
| circle(x, y, bán kính) | Vẽ đường tròn có tâm tại (x, y) |
| ellipse(x, y, góc đầu, góc cuối, a, b) | Vẽ cung elip với tâm, các góc và các bán kính theo hoành độ và tung độ tương ứng |
| fillellipse(x, y, a, b) | Vẽ hình elip đặc |
| floodfill(x, y, c) | Tô màu một hình kín chứa điểm x, y và màu c, màu c phải trùng với setfillstyle(mẫu tô, c); |
| outtextxy(x, y, s) | viết văn bản tại vị trí (x, y) |

Đó là một số hàm cơ bản có thể sử dụng trong “graphics.h”. Bên cạnh đó thì vẫn còn nhiều hàm để xử lý âm thanh, màu sắc, nét vẽ, …

### Thuật toán DDA (Digital Differential Analyzer) [4]

DDA (hay còn gọi là thuật toán số gia) là một thuật toán vẽ đoạn thẳng xác định các điểm dựa vào hệ số góc của phương trình đường thẳng . Trong đó .

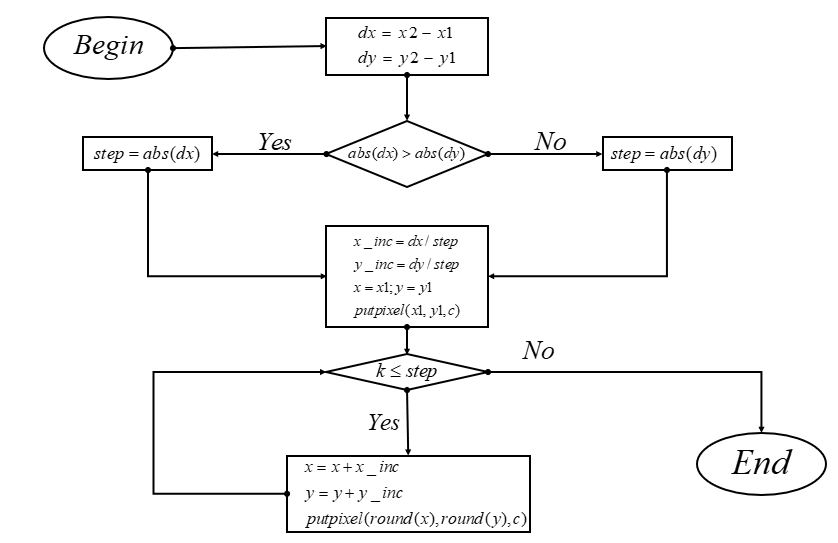


Hình 3 Thuật toán DDA

Nhận thấy trong hình vẽ 3 thì tọa độ điểm  sẽ tăng 1 đơn vị trên mỗi điểm vẽ, còn việc quyết định chọn là  hay  sẽ phụ thuộc vào giá trị sau khi làm tròn tung độ y. Ta có công thức tổng quát:

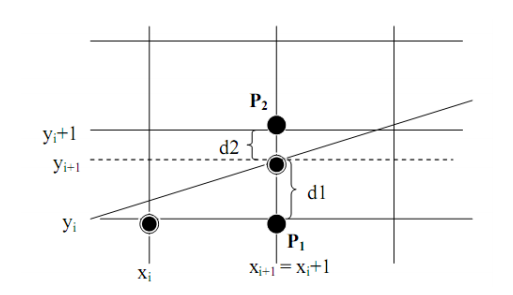


Ta có lưu đồ sau:



Hình 4 Lưu đồ thuật toán DDA

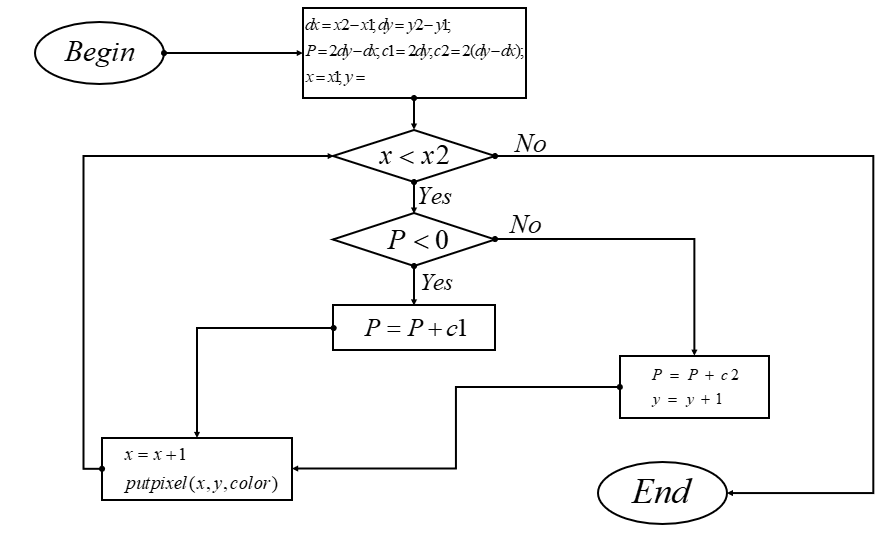
### Thuật toán Bresenham [4]



Hình 5 Thuật toán Bresenham

Với ý tưởng:  sẽ không giống với DDA, mà tối ưu tốc độ bằng cách hạn chế các phép toán trên số thực.

Ta có lưu đồ sau:



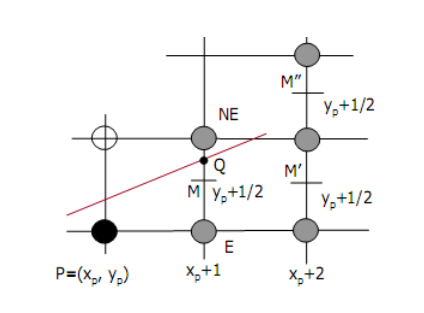
Hình 6 Lưu đồ thuật toán Bresenham

Nhận xét chung:

* Thuật toán Bresenham chỉ thao tác trên số nguyên và chỉ tính toán trên phép cộng và phép nhân 2. Điều này là một cải tiến làm tăng tốc độ đáng kể so với thuật toán DDA.
* Ý tưởng chính của thuật toán này là ở chỗ xét dấu Pi để quyết định điểm kế tiếp, và sử dụng công thức truy hồi Pi +1 - Pi để tính Pi bằng các phép toán đơn giản trên số nguyên.
* Tuy nhiên, việc xây dựng trường hợp tổng quát cho thuật toán Bresenham có phức tạp hơn thuật toán DDA.

### Thuật toán MidPoint [4]

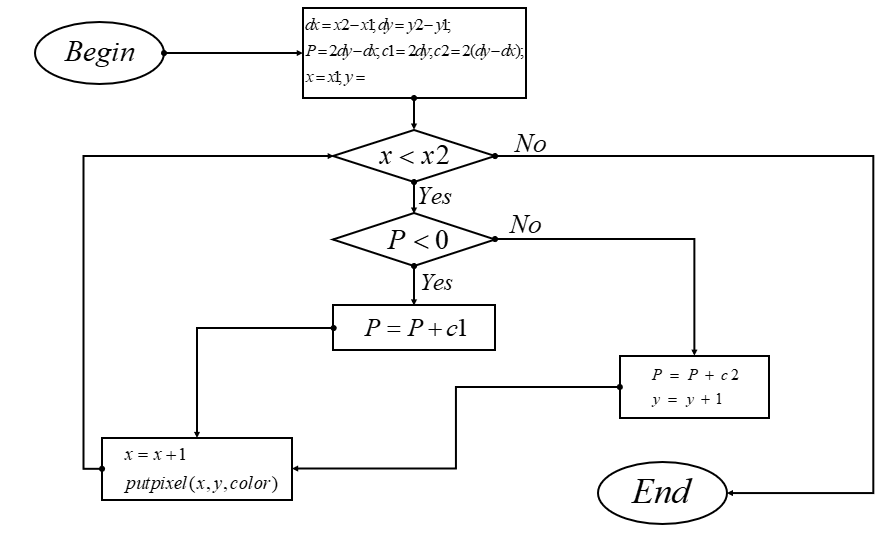
Pitteway công bố thuật toán MidPoint vào 1967, Van Aken cải tiến 1984. Xét hệ số góc thuộc [0, 1]. Giả thiết rằng đã chọn P để vẽ, xác định pixel tiếp theo sẽ là tại N hay NE Giao của đường thẳng với Xp+1 tại Q, M là trung điểm của NE và E.



Hình 7 Thuật toán MidPoint

Ý tưởng của thuật toán MidPoint là xét điểm M xem nằm phía nào của đường thẳng, nếu M nằm phía trên đường thẳng thì chọn E (tức là đường thẳng gần với E hơn NE), ngược lại chọn NE. Vì vậy, ta cần xác định vị trí tương đối của M so với đường thẳng chứa đoạn thẳng cần vẽ.

Ta có lưu đồ tương tự như của Bresenham:



Hình 8 Thuật toán MidPoint

### Thuật giải MidPoint vẽ đường tròn [4]

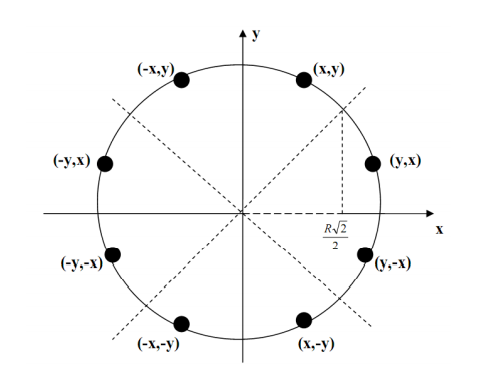
Trong hệ tọa độ Descartes, phương trình đường tròn bán kính R có dạng:

* Với tâm : 
* Với tâm : 

Trong hệ tọa độ cực

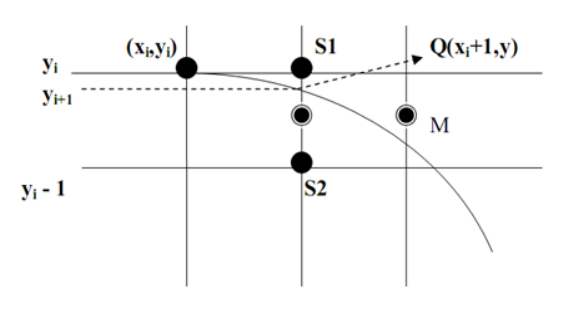
* 
* 

Với 



Hình 9 Đối xứng các điểm thông qua đường tròn

Do tính đối xứng của đường tròn C nên ta chỉ cần vẽ 1/8 cung tròn, sau đó lấy đối xứng qua 2 trục tọa độ và 2 đường phân giác thì ta vẽ được cả đường tròn.



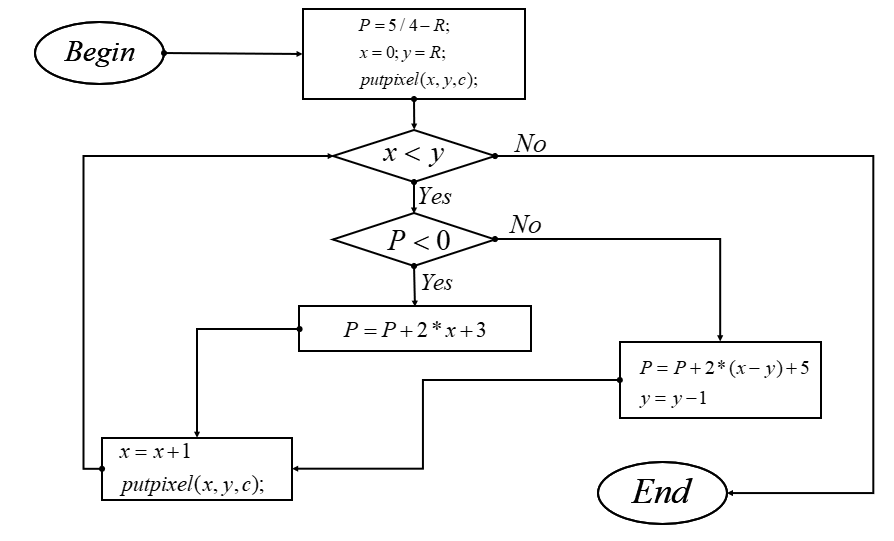
Hình 10 Đường tròn với điểm Q và điểm MidPoint

### Giải thuật vẽ đường cong dựa trên thuật toán MidPoint. [4]

Để vẽ đường được đường cong dựa trên thuật toán MidPoint, ta có các bước sau đây:

* Tạo phương trình đường cong (dựa vào tính đối xứng, đối xứng, hàm chẵn, hàm lẻ, …).
* Tính đạo hàm để phân từng vùng.
* Tính  dựa trên xét dấu.
* Thực hiện tính toán.

Ta có lưu đồ của giải thuật như sau:



Hình 11 Lưu đồ giải thuật vẽ đường cong dựa trên thuật toán MidPoint

### Giải thuật tô màu. [4]

#### Giới thiệu về tô màu.

Tô màu một vùng là thay đổi màu sắc của các điểm vẽ nằm trong vùng cần tô. Một vùng tô thường được xác định bởi một đường khép kín nào đó gọi là đường biên. Dạng đường biên đơn giản thường gặp là đa giác.

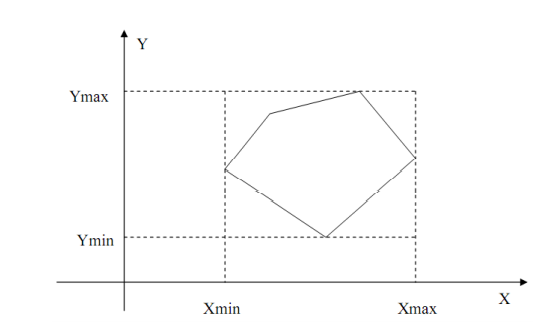
Việc tô màu thường chia làm 2 công đoạn:

* + Xác định vị trí các điểm cần tô màu.
  + Quyết định tô các điểm trên bằng màu nào. Công đoạn này sẽ trở nên phức tạp khi ta cần tô theo một mẫu tô nào đó chứ không phải tô thuần một màu. Giáo trình giới thiệu 3 cách tiếp cận chính để tô màu:
    - Tô màu theo từng điểm (có thể gọi là tô màu đơn giản).
    - Tô màu theo dòng quét (ScanConvert).
    - Tô màu dựa theo đường biên (FloodFill).

#### Tô màu theo từng điểm

Thuật toán này bắt đầu từ việc xác định một điểm có thuộc vùng cần tô hay không, nếu đúng thì sẽ tô với màu muốn tô.

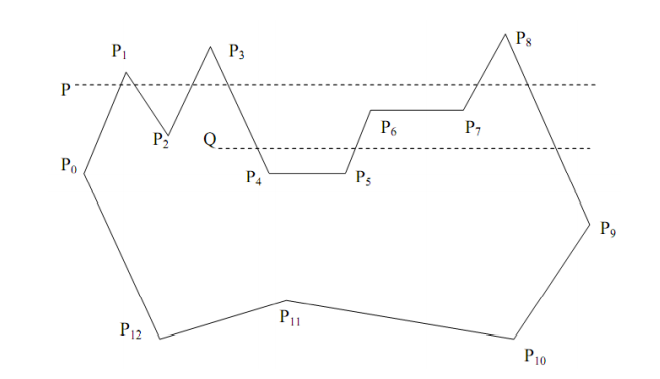
Không mất tính tổng quát, ta xét tô màu một đa giác bất kỳ. Đầu tiên ta tìm hình chữ nhật nhỏ nhất có các cạnh song song với hai trục tọa độ chứa đa giác cần tô dựa vào 2 tọa độ .



Hình 12 Đa giác nội tiếp Hình Chữ Nhật

Nguyên tắc xác định một điểm nằm trong đa giác:

* Dựa theo định lý nửa đường thẳng của Jordan.
* Một điểm nằm trong đa giác thì số giao điểm từ một tia bất kỳ xuất phát từ điểm đó cắt biên của đa giác phải là một số lẻ. Tia xuất phát có thể sang phải hay sang trái. Đặc biệt, tại các đỉnh cực trị thì một giao điểm phải được tính 2 lần. (Xem hình 13).



Hình 13 Đa giác có 13 đỉnh

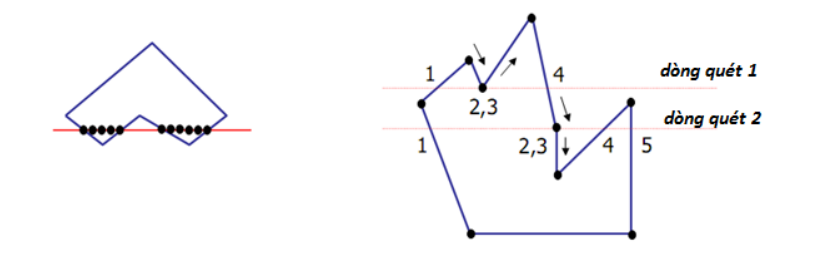
Thuật toán xác định điểm nằm trong đa giác:

* Với mỗi đỉnh của đa giác ta đánh dấu là 0 hay 1 theo quy ước như sau: nếu là đỉnh cực trị hay đoạn cực trị thì đánh số 0. Nếu là đỉnh đơn điệu hay đoạn đơn điệu thì đánh dấu 1.
* Xét số giao điểm của tia nửa đường thẳng từ P là điểm cần xét với biên của đa giác. Nếu số giao điểm là chẵn thì kết luận điểm không thuộc đa giác. Ngược lại, số giao điểm là lẻ thì điểm thuộc đa giác.

Nhận xét: Thuật toán tô đơn giản có ưu điểm là tô rất mịn và có thể sử dụng được cho đa giác lồi hay đa giác lõm, hoặc đa giác tự cắt, đường tròn, ellipse. Tuy nhiên, giải thuật này sẽ trở nên chậm khi ta phải gọi hàm PointInpoly nhiều lần. Để khắc phục nhược điểm này người ta đưa ra thuật toán tô màu theo dòng quét.

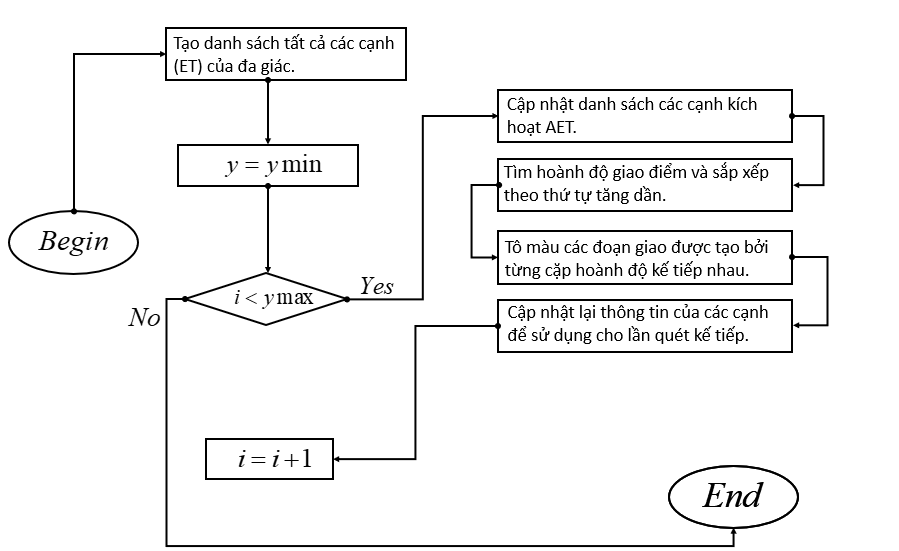
#### Tô màu theo dòng quét (ScanConvert)

Ý tưởng: Sử dụng giao điểm giữa các biên đa giác và đường quét để xác định các điểm nằm trong đa giác.



Hình 14 Tô màu theo từng dòng quét

Ta có lưu đồ thể hiện việc tô màu theo từng dòng quét:



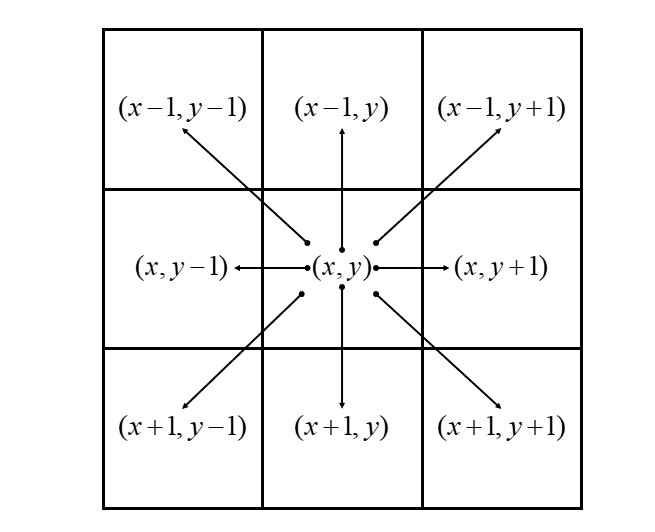
Hình 15 Lưu đồ tô màu theo từng dòng quét

#### Tô màu theo đường biên (FloodFill)

Ý tưởng:

* Thuật toán nhằm tô màu vùng kín, giới hạn bởi màu Bcolor, màu sử dụng để tô là Fcolor với điểm (x, y) nằm trong vùng tô màu.
* Thuật sử dụng phép gọi đệ quy, ban đầu (x, y) được kiểm tra màu, nếu màu của nó là Fcolor hoặc Bcolor thì tiến trình kết thúc. Trong trường hợp ngược lại, điểm (x, y) được tô với màu Fcolor và quá trình gọi đệ quy với các điểm lân cận của (x, y). Các điểm lân cận được sử dụng là bốn lân cận.

Tô màu theo đường biên được thể hiện như sau:



Hình 16 Tô màu theo đường biên

### Tạo dựng và hiển thị đối tượng đồ họa 2D [5]

#### Công cụ Turtle graphic

Là đồ họa vector sử dụng con trỏ tương đối.

Gồm 3 thuộc tính:

* Vị trí (location)
* Hướng (direction)
* Bút vẽ (pen):
  + Màu sắc (color)
  + Độ rộng (width)
  + Trạng thái (on/off state)

Với các đặc tính:

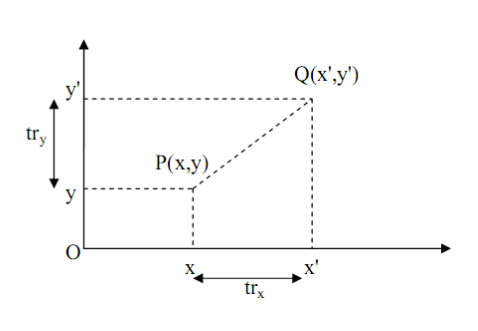
* Đi tới
* Đi lui
* Quẹo trái
* Quẹo phải

#### Phép tịnh tiến (Translation)

Có hai quan điểm về phép biến đổi hình học, đó là:

* Biến đổi đối tượng: thay đổi tọa độ của các điểm mô tả đối tượng theo một quy tắc nào đó.
* Biến đổi hệ tọa độ: Tạo ra một hệ tọa độ mới và tất cả các điểm mô tả đối tượng sẽ được chuyển về hệ tọa độ mới.

Nếu  là vector tịnh tiến thì: 

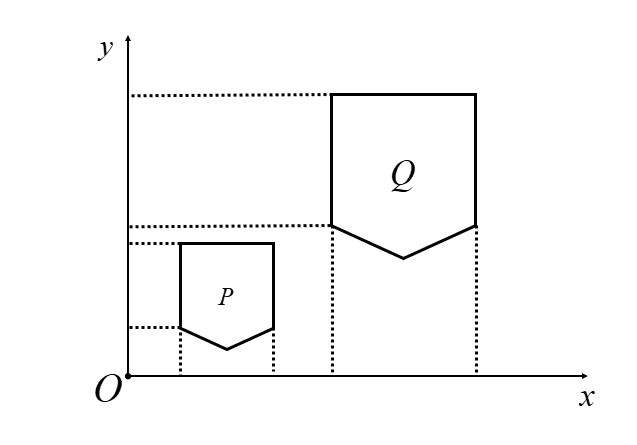


Hình 17 Phép biến đổi tịnh tiến từ P thành Q đối với một điểm

#### Phép co dãn/ tỉ lệ (Scale)

Phép biến đổi tỉ lệ làm thay đổi kích thước đối tượng. Để co hay giãn tọa độ của một điểm P (x, y) theo trục hoành và trục tung lần lượt là Sx và Sy (gọi là các hệ số tỉ lệ), ta nhân Sx và Sy lần lượt cho các tọa độ của P.



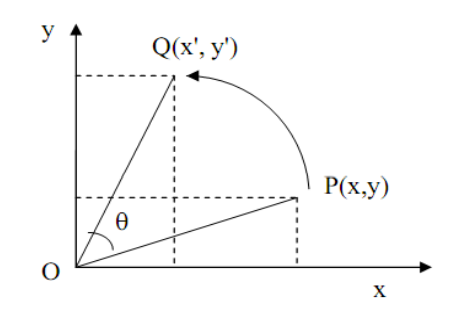


Hình 18 Phép co dãn P thành Q với một tỉ lệ nhất định

#### Phép xoay (Rotation)

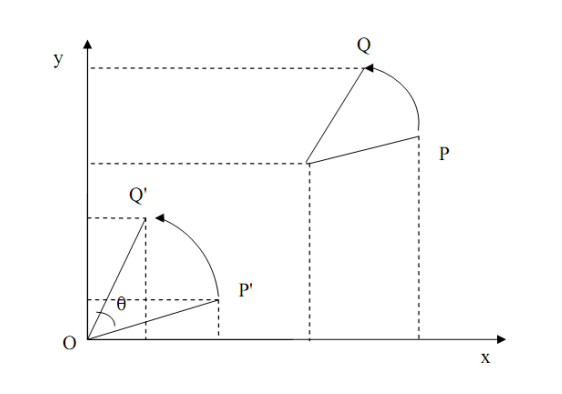
Ta có công thức biến đổi của phép quay điểm P (x, y) quanh gốc tọa độ góc θ:





Hình 19 Phép xoay tại gốc tọa độ

Tương tự cho phép xoay tại một điểm bất kì



Hình 20 Phép quay tại một điểm bất kì

#### Phép đối xứng

Phép đối xứng trục có thể xem là phép quay quanh trục đối xứng một góc . Với phương trình ban đầu:



### Xử lý sự kiện chuột và bàn phím với thư viện BGI [6]

Giới thiệu chung: Đối với các ứng dụng đồ họa như: vẽ, xử lý văn bản, xử lý bảng tính, … thì nhận, xử lý các sự kiện chuột và bàn phím là rất cần thiết. Nếu ứng dụng xử lý tốt các sự kiện này, giúp cho người dùng sử dụng hết các tính năng mà ứng dụng cung cấp cũng như tạo sự tiện dụng, nhanh chóng, tăng năng suất khi làm việc.

#### Sự kiện chuột

Phương thức đơn giản nhất trong sự kiện này:

int mousex()

int mousey()

Cả hai phương thức này trả về tọa độ của chuột trên khung cửa sổ ứng dụng tại thời điểm hàm này được gọi. Khi vị trí của chuột nằm ngoài khung ứng dụng thì việc gọi hàm này sẽ xảy ra 2 trường hợp:

Nếu chuột chưa từng di chuyển vào phạm vi cửa sổ ứng dụng - giá trị trả về là.

Nếu chuột đã từng ở trong phạm vi cửa sổ ứng dụng và sau đó di chuyển ra ngoài giá trị trả về tọa độ lần cuối cùng của chuột trong cửa sổ ứng dụng.

Thư viện BGI nhận các sự kiện chuột:

* WM\_MOUSEMOVE
* WM\_LBUTTONDBLCLK
* WM\_LBUTTONDOWN
* WM\_LBUTTONUP
* WM\_MBUTTONDBLCLK
* WM\_MBUTTONDOWN
* WM\_MBUTTONUP
* WM\_RBUTTONDBLCLK
* WM\_RBUTTONDOWN
* WM\_RBUTTONUP

Trong đó RBUTTON là chuột phải, LBUTTON là chuột trái và MBUTTON là chuột giữa. Các loại sự kiện chuột ở trên sẽ giúp ta dễ dàng thao tác với những hàm sau:

ismouseclick

getmouseclick

clearmouseclick

registermousehandler

#### Sự kiện phím

Tương tự như với chuột, thư viện BGI cũng cung cấp các hàm sau đây để xử lý sự kiện phím:

getch

* + KEY\_HOME
  + KEY\_UP
  + KEY\_PGUP
  + KEY\_LEFT
  + KEY\_CENTER
  + KEY\_RIGHT
  + KEY\_END
  + KEY\_DOWN
  + KEY\_PGDN
  + KEY\_INSERT
  + KEY\_DELETE
  + KEY\_F1
  + KEY\_F2
  + KEY\_F3
  + KEY\_F4
  + KEY\_F5
  + KEY\_F6
  + KEY\_F7
  + KEY\_F8
  + KEY\_F9

kbhit

## Thuật giải đề xuất

### Các ý tưởng đưa ra để giải quyết đề bài

Xây dựng hàm Loading. Tạo giao diện như một game chính thức, khi khởi động game thì mình có thể trải nghiệm cảm giác chân thật nhất có thể. Từ đó giúp cho người chơi (người sử dụng) hứng thú với game hơn.

Xây dựng hàm vẽ DDA, MidPoint, Elip.

Vẽ background cho game.

Xây dựng các hàm vẽ button. Xây dựng các button để giúp cho người chơi game dễ dàng tiếp cận với game hơn.

Sau đó xây dựng hàm cho phép người dùng nhập tên. Để lưu lại tên cùng như điểm số để cho người chơi phía sau có thể tra cứu được thông tin của người chơi trước để có thể cố gắng trong game tiếp theo.

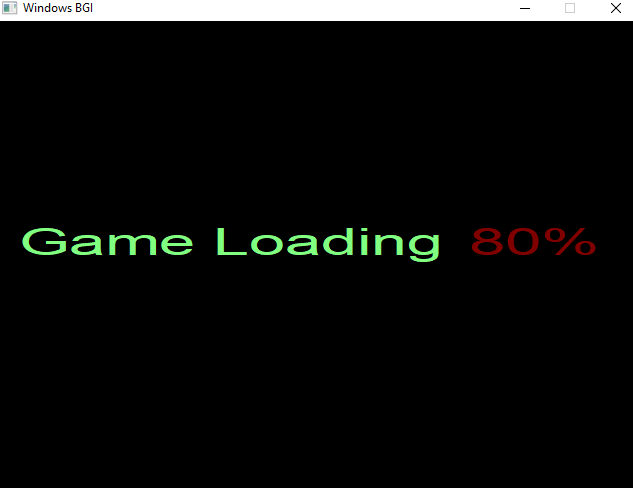
Vấn đề tiếp theo của bài toán đó chính là vẽ “vịt”.

Giải quyết vấn đề bắt chuột và bàn phím. Vì nó giúp cho người dùng chọn được các button, và có thể nhập tên của người dùng vào game.

### Giải quyết bài toán

#### Xây dựng hàm Loading

Nhằm tạo sự hứng thú cho người dùng, tạo cho người dùng có cảm giác đang trải nghiệm một game chân thật, từ đó mà nhóm đã viết ra hàm này để phục vụ điều đó.



Hình 21 Giao diện Game Loading

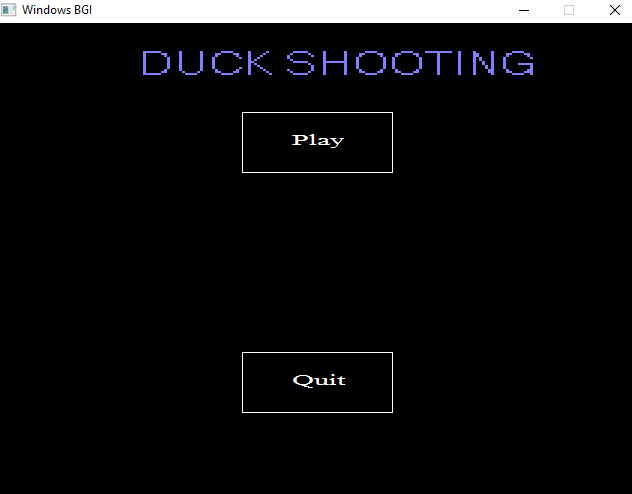
Để xây dựng hàm thì cần những việc như sau:

Sử dụng outtextxy: Để đưa ra giao diện chữ trên màn hình BGI (Game Loading), và số phần trăm.

Để cho giao diện phần trăm chạy từ 0 đến 100 thì sử dụng một vòng lặp WHILE và sau khi chạy thì thực hiện xóa màn hình để không lặp lại và chồng lên nhau.

#### Xây dựng hàm vẽ các button đầu game

Vấn đề đặt ra là không phải khi chạy xong hàm Loading thì tự vào game luôn. Yêu cầu cần phải có các nút để cho người dùng nhấn để cải thiện game có thể đẹp hơn, dễ sử dụng hơn nữa.



Hình 22 Button Play và Button Quit cho người dùng lựa chọn

Để có được giao diện như trên chúng tôi đã thực hiện như sau:

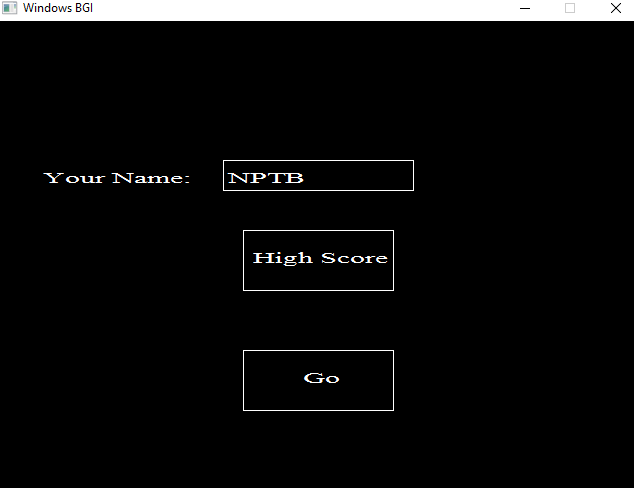
Xác định tọa độ giữa màn hình để vẽ 2 button này vào giữa. Bằng cách sử dụng getmaxx và getmaxy.

Sử dụng hàm lineDDA để nối các tọa độ đã có và cộng trừ một cách phù hợp để vẽ ra được button trên.

Sau đó sử dụng các hàm outtextxy để xuất ra Play và Quit.

Giải quyết việc click chuột thì chuyển sang giao diện khác. (Được giải thích ở [2.1.14](#_Xử_lý_sự))

#### Xử lý nhập tên trong game.



Hình 23 Giao diện sau khi chạy xong Game Loading

Các vấn đề được đưa ra như sau:

Tạo button.

Xuất chữ: Your Name, High Score, Go.

Giải quyết vấn đề nhập tên từ bàn phím. (Được tìm hiểu tại [2.1.14](#_Xử_lý_sự))

Cách giải quyết vấn đề nhập tên từ bàn phím:

Yêu cầu đặt ra là chỉ cho phép người dùng nhập từ bàn phím các phím từ a đến z, A đến Z, và các số từ 0 đến 9. Do yêu cầu đặt ra là tê người chơi chỉ cho phép nhập chữ và chữ số. Và độ dài tên không quá 15 ký tự.

Cách xử lý:

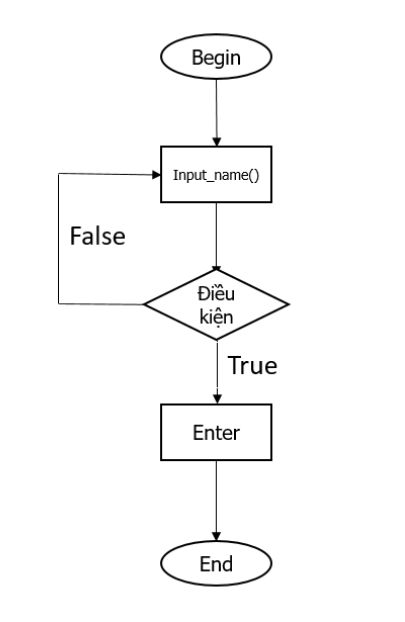
Viết hàm vẽ các button để hiển thị cho người dùng sử dụng. Như sau: “button\_go”, “name\_BG”.

Cho phép người dùng nhập từ bàn phím từ a đến z, từ A đến Z và số từ 0 đến 9.

Khi nhập thì các ký tự được nhận dưới dạng mã ASCII. Sau đó chuyển từ ASCII sang dạng chuỗi (char) để hiện thị ra bên ngoài màn hình, bên cạnh đó sử dụng “strcat” để nối chuỗi thành tên của người chơi.

Chạy vòng lặp đến khi nhấn phím Enter thì lưu tên vào bảng điểm Highscore và kết thúc hàm việc nhập tên.

Nếu người dùng nhập dài hơn 15 ký tự hoặc những ký tự không cho phép thì chương trình sẽ xóa đi dữ liệu đã nhập và cho người dùng nhập lại.



Hình 24 Lưu đồ thể hiện quá trình nhập tên từ bàn phím

#### Xử lý vấn đề điểm (High Score)

Trong quá trình chơi game chúng ta sẽ ghi nhận lại điểm số của người chơi nếu người chơi bắn trúng (click) vịt thì sẽ tính điểm cho mỗi lần bắn trúng là 50 điểm. Và sẽ có 3 lần bắn.

Yêu cầu đặt ra:

Chia làm 3 cột. Mỗi cột một nhiệm vụ của nó tiêu biểu như: STT, Name, Point.

Hiển thị số thứ tự.

Hiển thị tên người chơi.

Hiển thị điểm người chơi.

Sắp xếp những người chơi có điểm cao nhất lên dòng đầu tiên theo thứ tự giảm dần từ trên xuống dưới.

Tạo button Go tại đây để cho người chơi có thể bắt đầu game tại chính màn hình luôn, khỏi phải trở về lại màn hình chính rồi chơi game.

Cách xử lý trong giao diện High Score:

Sau khi nhập tên người chơi thông qua đầu game thì tên của người chơi sẽ được lưu dưới dạng file “kq.txt”. Đồng thời điểm của người chơi cũng sẽ được lưu tại file này.

Để hiển thị được tên và điểm số của người dùng lên thì sẽ sử dụng 2 hàm trong đó có:

* + “lineCount”: Đếm số dòng trong file “kq.txt” để tiện cho việc xét duyệt.
  + “readPoint”: Hàm này sẽ thực hiện bằng cách đếm số dòng trong file thông qua hàm trên rồi sau đó sẽ đọc giá trị điểm của từng người chơi thích hợp. Nếu mà đọc file không được sẽ hiển thị thông báo qua màn hình Console: “Open file error!”

Để xếp hạng được người chơi: Cách giải quyết đặt ra là sẽ chạy các vòng lặp xét điểm từ trên xuống rồi sau đó sẽ hoán vị thay đổi vị trí của các hàng. Sau đó chuyển điểm của người chơi cao nhất lên trên cùng. Thông qua các hàm “xephang”, “Point\_Player” và “high\_Score”.

Giao diện của High Score:



Hình 25 Giao diện của High Score trước khi có điểm



Hình 26 Giao diện của High Score

#### Xử lý vấn đề vẽ vịt

Yêu cầu đặt ra:

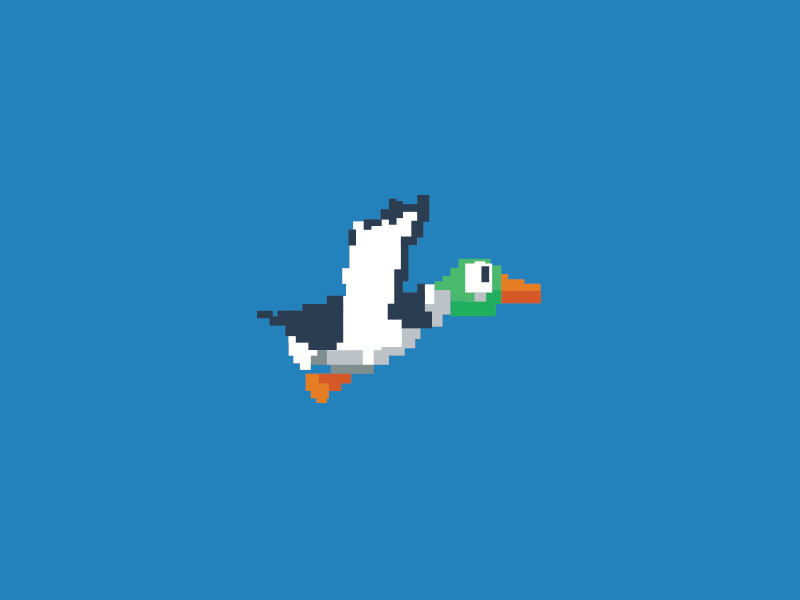
Nhân vật vịt trong game phải được vẽ bằng các hàm tự định nghĩa.

Không được sử dụng các API có sẵn như: line, circle, elip, …

Xử lý bài toán:

Sử dụng hàm lineDDA để vẽ các đường thẳng tạo thành các chi tiết của vịt.

Từ đó xây dựng được hàm như: “drawLipDuck\_Left”, “drawRearDuck\_Left”, “drawHeadDuck”, “drawHandDuckTop”, “drawHandDuckDown”, “drawDuck\_Left”.

Hình 27 Chú vịt trước và sau khi được xử lý

Đó không phải là sự khác biệt quá lớn giữa 2 chú vịt mà thư viện “graphics.h” mang lại. Đây mới là sự khác biệt sau khi demo dự kiến thông qua bài học về đồ họa 3D.



Hình 28 Đàn vịt dự kiến sau khi qua xử lý 3D

#### Xử lý vấn đề bắn vịt

Yêu cầu đặt ra:

Chia ra làm 3 round. Mỗi round tốc độ vịt bay sẽ khác nhau.

Vịt bay từ bên trái qua phải, từ bên phải sang bên trái.

Mỗi lần bắn trúng vịt người chơi sẽ được cộng thêm 50 điểm.

Nếu bắn trúng thì một viên đạn, còn bắn hụt thì sẽ bị trừ đi 1 viên đạn.

Vịt sẽ luôn hiển thị tại vị trí ban đầu nếu người chơi bắn trúng hoặc hụt vịt.

Nếu điểm đạt được 100 điểm thì sẽ sang round 2.

Nếu đạt được 300 điểm thì sẽ sang round 3.

Sau khi chơi xong game thì hiển thị lại điểm của người chơi

Xử lý yêu cầu cho vịt xuất hiện:

Sử dụng các hàm vẽ vịt đã viết ở trên.

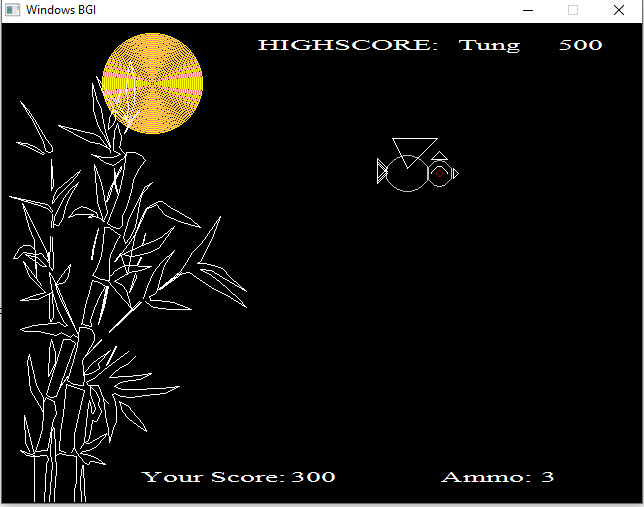
Sử dụng vòng lặp để thay đổi từng vị trí của các vịt.

Ở mỗi round thì tốc độ của vịt sẽ khác nhau vì tốc độ vẽ background sẽ khác nhau. Số lượng ảnh background cần vẽ giảm dần và vịt sẽ được vẽ nhanh hơn khi tiến vào các ground tiếp theo.

Viết các hàm kiểm tra (check) khi bắn trúng vịt, khi không bắn hụt vịt.

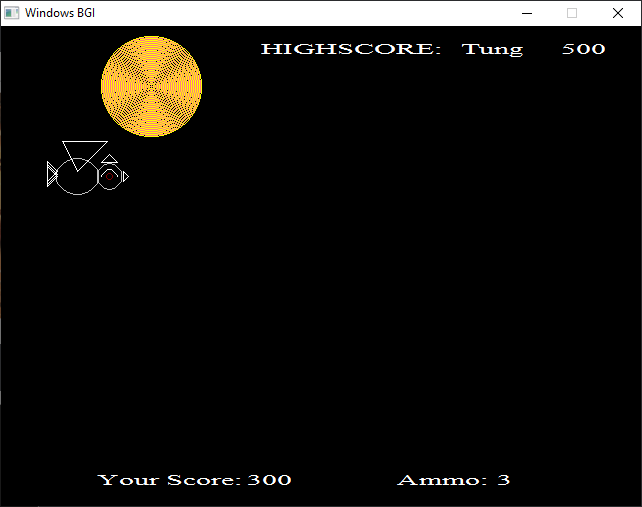
Nếu tọa độ khi click chuột trùng với tọa độ của vịt thì hàm “checkDuckDie” sẽ kiểm tra điều đó đúng hay không. Nếu đúng thì vịt mất, tức là sẽ xóa màn hình và cho chạy lại vòng lặp và được cộng điểm. Khi vịt không chết cũng sẽ chạy lại vòng lặp nhưng không được cộng điểm.

Sau mỗi round sẽ có những sự thay đổi.

Round 1: có thêm hàm vẽ “drawSun” và “draw\_BamBoo”. 

Hình 29 Giao diện Round 1

* + Round 2: Chỉ còn lại hàm “drawSun”. Với tốc độ vẽ nhanh hơn round 1.

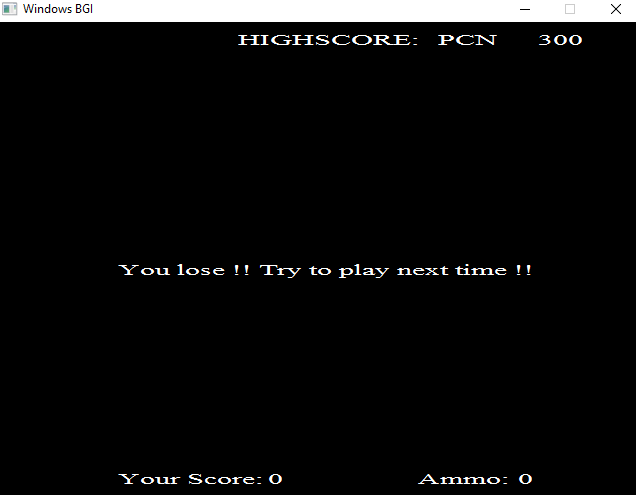


Hình 30 Giao diện Round 2

* + Round 3: Không còn sử dụng hàm “drawSun”. Nhưng ở đây thì tốc độ bay của vịt được cải thiện một cách đáng kể. Sẽ nhanh hơn, khiến cho người chơi khó có thể bắn trúng được vịt.



Hình 31 Giao diện Round 3



Hình 32 Giao diện kết thúc game

#### Các hàm hỗ trợ cho game

Sau đây là các hàm hỗ trợ game

Hàm “plot” hỗ trợ việc vẽ các elip tạo nên hình dạng của chú vịt.

Hàm “button\_Play\_Quit” hỗ trợ hiển thị 2 button đầu game.

Hàm “button\_Go” hỗ trợ người dùng chơi tại màn hình Hight Score.

Hàm “buttonCheckDie” hỗ trợ việc click chuột có đúng vào trong tọa độ của các vật thể hay không. Chẳng hạn như các button Play, Quit, Go, …

Hàm “drawSun” hỗ trợ vẽ mặt trời cho background.

Hàm “checkDuckDie” hỗ trợ kiểm tra chuột có click trúng các tọa độ của vịt.

## Kết quả

Bảng phân công công việc:

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | Công việc được phân công |
| 1 | Trương Thiên Phong | * Code chính |
| 2 | Bùi Phương Tùng | * Code chính |
| 3 | Phạm Chí Năng | * Vẽ background game * Viết báo cáo Word * Hỗ trợ sửa bug. |
| 4 | Nguyễn Nguyệt Phương Bảo | * Viết các hàm xử lý vẽ vịt và high score. * Soạn bài thuyết trình PowerPoint. * Hỗ trợ sửa bug. |

Bảng liệt kê:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Nhiệm vụ | Kết quả (Pass/Fail) | Ghi chú |
| 1 | Tạo hàm Loading | Pass |  |
| 2 | Tạo các button (Play, Quit, Go, InputName, ...) | Pass |  |
| 3 | Vẽ background cho game | Pass | Quá trình load background gây ra hiệu ứng chớp màn hình. Dẫn đến trải nghiệm game không như mong muốn |
| 4 | Xử lý vịt bay | Pass | * Vịt chỉ bay từ bên trái qua * Có hàm dành cho vịt bên phải. |
| 5 | Nhập tên người chơi | Pass | Không cho phép nhập các ký tự đặc biệt. Và độ dài có giới hạn |
| 6 | Nhập điểm người chơi | Pass |  |
| 7 | Xếp hạng người chơi | Pass |  |
| 8 | Xử lý bắn vịt | Pass |  |
|  |  |  |  |

# KẾT LUẬN

## Những kết quả đạt được

Phát triển được một game từ nền tảng ngôn ngữ C++ thông qua sự hỗ trợ của thư viện “graphics.h”

Hiểu được cách vẽ các đối tượng ngoài đời vào trong ngôn ngữ lập trình nói chung và ngôn ngữ C++ nói riêng.

Giúp tăng khả năng sáng tạo cho đề tài.

Cải thiện được vấn đề làm việc nhóm, hỗ trợ lẫn nhau trong việc code và phát hiện các lỗi (bug) trong đề tài.

## Ưu điểm của đề tài

Giao diện vào game đầu tiên khiến người chơi hứng thú muốn trải nghiệm.

Cho phép người chơi có thể nhập tên để dễ dàng quan sát.

Có bảng điểm “High Score” giúp người chơi dễ dàng thấy được ai đang là người giữ kỷ luật của game.

Game không quá khó, nên việc bắn vịt điểm cao sẽ rất dễ dàng.

Điều nổi bật ở đây là nếu trong game bạn bắn trúng vịt thì sẽ được tặng thêm 1 viên đạn, khác hẳn với những game bắn vịt trước đây.

Bắn vịt trong game dễ dàng khi có sử dụng bắt chuột.

Giao diện với background cây tre và mặt trời, là những hình ảnh đặc trưng của đất nước Việt Nam của chúng ta, nên việc có những hình ảnh này trong game giúp người chơi nhớ lại nguồn cội của mình.

## Nhược điểm của đề tài

Đầu chương trình có thực hiện việc Loading đẹp nhưng khi vào bên trong thì hơi nhạt nhẽo, có thể khiến người chơi tụt hứng. Vì chỉ có 2 button Play và Quit.

Giao diện nhập tên chỉ cho phép nhập những ký tự được cho phép

Độ dài của tên bị giới hạn.

Không có nhiều chế độ cho game.

Giao diện chơi game không bắt mắt.

## Hướng phát triển trong tương lai

Phát triển về mặt hình ảnh trong game.

Phát triển game có nhiều level hơn.

Phát triển từ game 2D lên game 3D.

Sửa chữa những khuyết điểm mà ở bản hiện tại đang có.

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | Sử dụng "graphics.h" để xây dựng ứng dụng đồ họa, "www.stdio.vn," 06 07 2015. [Online]. Available: "https://www.stdio.vn/modern-cpp/su-dung-graphics-h-de-xay-dung-ung-dung-do-hoa-M1L2BL". |
| [2] | BGI. [Online]. Available: http://winbgim.codecutter.org. |
| [3] | B. G. I. (. f. Windows, "https://home.cs.colorado.edu/," [Online]. Available: https://home.cs.colorado.edu/~main/cs1300/doc/bgi/index.html. |
| [4] | T. V. P. B. -. T. đ. h. Đ. L. K. C. N. T. Tin, "Giáo Trình ĐỒ HỌA MÁY TÍNH". |
| [5] | T. N. V. T. H. Tuyết, "Giáo án bài giảng," in *Môn ĐỒ HỌA MÁY TÍNH - Trường Đại Học Mở TPHCM*, 2020. |
| [6] | X. l. s. k. c. v. b. p. đ. v. t. v. BGI, "www.stdio.vn," 2017. [Online]. Available: https://www.stdio.vn/modern-cpp/xu-ly-su-kien-chuot-va-phim-voi-borland-graphics-interface-bgi-q2BL2. |