|  |  |
| --- | --- |
| **logo moi** | BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM**  Khoa Công Nghệ Thông Tin |

**ĐỒ ÁN MÔN HỌC**

**ĐỒ HỌA MÁY TÍNH**

**THIẾT KẾ GAME 2D**

**“Rắn Săn Mồi”**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Giảng viên hướng dẫn: Thầy Nguyễn Đông Phương

Sinh viên thực hiện: Nguyễn Tiến Thịnh

MSSV: 1711061728 - Lớp: 17DTHA6

Trần Như Tín

MSSV: 1711061743 - Lớp: 17DTHA6

Thành phố Hồ Chí Minh, 2020

**MỤC LỤC**

# **DANH MỤC HÌNH VẼ**

[Hình 2.1. Khai báo chương trình 2](#_Toc18950)

[Hình 2.2. Định dạng font chữ 2](#_Toc5157)

[Hình 2.3. Xử lý văn bản 2](#_Toc24164)

[Hình 2.4. Khởi tạo tọa độ cây 3](#_Toc26175)

[Hình 2.5. Tọa độ của bầu trời 3](#_Toc14743)

[Hình 2.6. Tọa độ đồi 1 4](#_Toc25286)

[Hình 2.7. Tọa độ đồi 2, 3, 4 5](#_Toc4546)

[Hình 2.8. Thiết kế thân máy bay 7](#_Toc2209)

[Hình 2.9. Thiết kế đuôi máy bay 8](#_Toc20031)

[Hình 2.10. Thiết kế cánh máy bay 9](#_Toc7588)

[Hình 2.11. Thiết kế lầu 1 10](#_Toc12776)

[Hình 2.12. Thiết kế tầng 2 10](#_Toc1075)

[Hình 2.13. Thiết kế 2 trệt và 4 cửa sổ 12](#_Toc23740)

[Hình 2.14. Thiết kế mũi tên bên trái 13](#_Toc11988)

[Hình 2.15. Thiết kế mũi tên bên phải 14](#_Toc1955)

[Hình 2.16. Thiết kế mũi tên lên, xuống 14](#_Toc13150)

[Hình 2.17. Thiết kế renderBitmapString 15](#_Toc11447)

[Hình 2.18. Tọa độ Background 16](#_Toc3043)

[Hình 2.19. Thiết kế vạch thứ nhất 17](#_Toc6437)

[Hình 2.20. Thiết kế vạch thứ hai 18](#_Toc3002)

[Hình 2.21. Thiết kế vạch thứ ba 19](#_Toc21721)

[Hình 2.22. Thiết kế máy bay người chơi 20](#_Toc1810)

[Hình 2.23. Thiết kế máy bay vật cản sau cùng 21](#_Toc14104)

[Hình 2.24. Thiết kế máy bay vật cản ở giữa 22](#_Toc17571)

[Hình 2.24. Thiết kế máy bay vật cản đầu tiên 24](#_Toc11801)

[Hình 2.24. Thiết kế mây 25](#_Toc15012)

[Hình 2.25. Thiết kế núi 26](#_Toc31914)

[Hình 2.26. Xuất điểm, tốc độ và level 27](#_Toc30806)

[Hình 2.27. Tính điểm và thông báo Game Over 28](#_Toc5643)

# **LỜI CÁM ƠN**

Em xin cảm ơn thầy Nguyễn Đông Phương - giảng viên bộ môn “ Đồ họa máy tính” đã giảng dạy cho chúng em những kiến thức về đồ họa máy tính thật bổ ích, qua đó em cũng đã tự làm và hoàn thành game 2D nhỏ “Rắn Săn Mồi”.

Tuy nhiên trong quá trình nghiên cứu và thực hiện đồ án, do kiến thức chuyên ngành còn nhiều hạn chế nên chúng em vẫn còn nhiều thiếu sót khi thực hiện. Nhóm em mong thầy đọc qua báo cáo và góp ý những sai sót trong đồ án để nhóm em có thêm kinh nghiệm và có thể hoàn chỉnh đồ án hơn nữa.

Một lần nữa em xin chân thành cám ơn những ý kiến, những kiến thức và chỉ bảo của thầy trong suốt thời gian qua.

**Sinh viên thực hiện**

Nguyễn Tiến Thịnh

Trần Như Tín

# **PHẦN MỞ ĐẦU**

## **Mục tiêu của đồ án**

* Trang bị cho sinh viên kiến thức chung về OpenGL, ứng dụng công nghệ thông tin trong OpenGL.
* Tăng khả năng tư duy, thiết kế, quan sát để có thể tiếp cận tốt hơn với những công nghệ mới, tạo ra những sản phẩm mang tính thực tế cao.
* Chương trình Game được viết bằng Visual Studio 2019.

## **Lý do chọn đề tài**

* Vì mô hình Game 2D “Airplane Flight Game” là một thể loại game thể thao tốc độ, kích thích tư duy logic, óc quan sát, khả năng kiên nhẫn và giảm stress sau những giờ học tập, làm việc căng thẳng.
* Người chơi né các chướng ngại vật bằng cách bấm phím mũi tên di chuyển lên, xuống từ bàn phím để tránh chuyển động của máy bay vật cản tiến tới.

## **Nguyên lý hoạt động của mô hình**

* Game 2D “Airplane Flight Game” hoạt động dựa trên sự chuyển động của những chiếc máy bay di chuyển lên, xuống.
* Game có lối chơi đơn giản. Như mọi game lái máy bay khác, bạn là người cầm lái và đưa máy bay mình phóng nhanh nhất, né tránh các máy bay vật cản khác. Điều khiển máy bay khá dễ dàng. Người chơi tăng tốc bằng cách nhấn phím mũi tên Left và giảm tốc bằng cách nhấn phím mũi tên Right trên bàn phím.
* Trong game có các máy bay vật cản cản đường. Vì vậy người chơi cần khéo léo trong việc điều khiển máy bay sao cho tránh được các chướng ngại vật bằng cách nhấn phím mũi tên Up để máy bay di chuyển lên và nhấn phím mũi tên Down để máy bay di chuyển xuống.
* Sau khi hoàn thành tốc độ nhất định, người chơi sẽ lên level mới, xuất hiện các chướng ngại vật khó hơn, hấp dẫn và kịch tính hơn.

# 

# **TRIỂN KHAI ỨNG DỤNG**

## **Giới thiệu thư viện**

Theo định nghĩa tin học, OpenGL (Open Graphics Library) là một tiêu chuẩn kỹ thuật [đồ họa](https://quantrimang.com/do-hoa-may-tinh) được hình thành với mục đích định ra một [giao diện lập trình ứng dụng (API)](https://quantrimang.com/giao-dien-lap-trinh-ung-dung-api-la-gi-148288) theo không gian 3 chiều.OpenGL cũng có thể được dùng trong các ứng dụng đồ họa 2 chiều. Giao diện lập trình này chứa khoảng 250 hàm để vẽ các cảnh phức tạp từ những hàm đơn giản. Nó được dùng rộng rãi trong các trò chơi điện tử. Ngoài ra nó còn dùng trong các ứng dụng CAD, thực tế ảo, mô phỏng khoa học, mô phỏng thông tin, phát triển trò chơi.

Thư viện hỗ trợ OpenGL

* GLUT - Thư viện công cụ cho OpenGL
* GLU - Thư viện hàm cho các ứng dụng OpenGL

## **Triển khai, thiết kế**

### ***Cấu trúc chương trình***

Bước 1: Khi chạy vào chương trình

Bước 2: Nhấn phím “SPACE” để bắt đầu

Bước 3: Nhấn phím mũi tên Up, Down để máy bay di chuyển lên, xuống và nhấn phím Right, Left để giảm tốc, tăng tốc độ.

Bước 4: Nhấn phím “ESC” để thoát

### ***Thiết kế Game “Airplane Flight Game”***

### Hàm khai báo chương trình

*Hình 2.1. Khai báo chương trình*

### Hàm định dạnh font chữ

*Hình 2.2. Định dạng font chữ*

### Hàm xử lý văn bản

*Hình 2.3. Xử lý văn bản*

**Chức năng của hàm là:** Xây dựng hàm renderBitmapString với 4 giá trị truyền vào. Đối số thứ 1, 2 là tọa độ mà chuỗi chuyền vào sẽ hiển thị. Đối với đối số thứ 3 mặc định sẽ là: GLUT\_BITMAP\_HELVETICA\_10. Đối với tham số cuối cùng đó chính là chuỗi mà chúng ta chuyền vào.

* Ta có thể hiển thị văn bản ở bất kỳ vị trí raster nào của cửa sổ. Vị trí raster có thể được đặt bằng cách sử dụng nhóm hàm glRasterPos từ thư viện OpenGL. Cú pháp của các hàm là: **glRasterPos2f (x, y);**
* Chức năng hiển thị ký tự bitmap bằng OpenGL là glutBitmapCharacter. Cú pháp là **glutBitmapCharacter (void \* font, int character);**

### Hàm khởi tạo cây

*Hình 2.4. Khởi tạo tọa độ cây*

**Chức năng của hàm:**

* Ta khởi tạo thân và đỉnh cây bằng hình tam giác. Câu lệnh tạo: **glBegin(GL\_TRIANGLES);**
* Tham số truyền vào hàm bao gồm tọa độ (x, y) của hình tam giác.

### Hàm vẽ first Design

### *Tọa độ bầu trời*

*Hình 2.5. Tọa độ của bầu trời*

Kết quả chạy chương trình:

### *Tọa độ đồi*

*Hình 2.6. Tọa độ đồi 1*

**Chức năng của hàm:** Ta xây dựng đồi bằng một hình tam giác có tọa độ (x, y) màu xanh nhạt.

Kết quả chạy chương trình: Đồi thứ nhất xuất hiện tại tọa độ V1(15, 30); V2(27,20); V3(0,20).

Ta xác định tọa độ cho các đồi còn lại:

*Hình 2.7. Tọa độ đồi 2, 3, 4*

*Kết quả ta có:*

### *Vị trí đặt cây*

**Chức năng của hàm:** Do chúng ta đã xây dựng một hàm thiết kế cây. Để sử dụng hàm này, chúng ta sẽ gọi hàm và truyền tọa độ (x, y) tại các vị trí khác nhau. Cụ thể ở đây chúng ta khởi tạo 5 vị trí tọa độ (x, y) cho 5 cây.

Kết quả chạy chương trình:

### *Thiết kế máy bay*

* **Thiết kế thân máy bay**

*Hình 2.8. Thiết kế thân máy bay*

**Chức năng của hàm:**Xây dựng thân máy bay bằng câu lệnh vẽ hình đa giác. Tham số truyền vào hàm bao gồm tọa độ (x, y).

Kết quả ta được:

* **Thiết kế đuôi máy bay**

*Hình 2.9. Thiết kế đuôi máy bay*

Thiết kế đuôi máy bay bằng câu lệnh vẽ hình đa giác: **glBegin(GL\_POLYGON)**. Truyền tọa độ (x, y) để cho ra 2 hình chữ nhật kết hợp với nhau.

Kết quả ta được:

* **Thiết kế cánh máy bay**

*Hình 2.10. Thiết kế cánh máy bay*

Thiết kế cánh máy bay tương tự đuôi máy bay. Sử dụng câu lệnh vẽ hình đa giác: **glBegin(GL\_POLYGON)**. Truyền tọa độ (x, y) để cho ra 2 hình chữ nhật kết hợp với nhau.

Kết quả chạy chương trình:

### *Tọa độ nhà tầng*

* **Thiết kế tầng 1**

*Hình 2.11. Thiết kế lầu 1*

**Chức năng của hàm:**Xây dựng tầng 1 bằng câu lệnh vẽ hình đa giác. Tham số truyền vào hàm bao gồm tọa độ (x, y). Truyền tạo độ để cho ra hình chữ nhật.

Kết quả ta được:

* **Thiết kế tầng 2**

*Hình 2.12. Thiết kế tầng 2*

Tương tự việc thiết kế tầng 1, sử dụng câu lệnh vẽ hình đa giác và truyền tọa độ (x, y) để cho ra hình chữ nhật.

Kết quả ta được:

* **Thiết kế 2 trệt phía bên hông tầng 1 và 4 cửa sổ nhỏ tầng 1**

*Hình 2.13. Thiết kế 2 trệt và 4 cửa sổ*

Kết quả chạy chương trình:

### *Thiết kế các mũi tên di chuyển*

*Hình 2.14. Thiết kế mũi tên bên trái*

**Chức năng của hàm:** Xây dựng thân mũi tên bằng câu lệnh vẽ hình đa giác. Và đỉnh nhọn mũi tên bằng câu lệnh vẽ hình tam giác. Tham số truyền vào là tọa độ (x, y).

Kết quả chạy: Xuất hiện mũi tên bên trái tại vị trí như trên màn hình.

Tương tự, ta xác định các vị trí tọa độ cho các mũi tên bên phải:

*Hình 2.15. Thiết kế mũi tên bên phải*

Xác định tọa độ cho mũi tên lên, xuống:

*Hình 2.16. Thiết kế mũi tên lên, xuống*

Kết quả ta được:

### *Thiết kế renderBitmapString*

*Hình 2.17. Thiết kế renderBitmapString*

**Chức năng của hàm:**  Xây dựng hàm renderBitmapString có tham số truyền vào là tọa độ (x, y), phông chữ ta đã định dạnh sẵn và văn bản cần hiển thị.

Kết quả chương trình chạy:

### Hàm bắt đầu Game

### *Xây dựng tọa độ background*

*Hình 2.18. Tọa độ Background*

Kết quả chạy chương trình:

### *Thiết kế vạch tính điểm*

* *Vạch thứ nhất có tọa độ x từ 80 -> 100 (vạch sau cùng)*

Xác định tọa độ (x, y) để vẽ vạch thứ nhất bằng câu lệnh vẽ hình đa giác.

*Hình 2.19. Thiết kế vạch thứ nhất*

Ta có roadDivTop giảm 1 đơn vị. Nếu roadDivTop < - 100, tức là ra ngoài khoảng Map (Vì Map có độ rộng là 100). Khi đó gán cho roadDivTop = 20, tức là tạo ra một hình chữ nhật mới có tọa độ x là 100 -> 120 và cứ liên tục như vậy. Đồng thời tăng score lên.

* *Vạch thứ hai có tọa độ x từ 40 -> 60 (vạch ở giữa)*

Xác định tọa độ (x, y) để vẽ vạch thứ hai bằng câu lệnh vẽ hình đa giác.

*Hình 2.20. Thiết kế vạch thứ hai*

* *Vạch thứ ba có tọa độ x từ 0 -> 20 (vạch đầu tiên)*

Xác định tọa độ (x, y) để vẽ vạch thứ ba bằng câu lệnh vẽ hình đa giác.

*Hình 2.21. Thiết kế vạch thứ ba*

### *Thiết kế tọa độ máy bay*

* *Tọa độ của máy bay người chơi*

*Hình 2.22. Thiết kế máy bay người chơi*

**Chức năng của hàm:**

* Thiết kế thân máy bay người chơi bằng câu lệnh vẽ hình đa giác có 2 màu xanh lá cây đậm và xanh nước biển đậm kết hợp.
* Thiết kế đuôi máy bay và cánh máy bay bằng câu lệnh vẽ hình đa giác: **glBegin(GL\_POLYGON)**. Truyền tọa độ (x, y) để cho ra 2 hình chữ nhật kết hợp với nhau.

Kết quả ta có:

* *Tọa độ của máy bay vật cản sau cùng*

*Hình 2.23. Thiết kế máy bay vật cản sau cùng*

Kết quả chạy chương trình:

Ta có plane1 giảm 1 đơn vị. Nếu plane 1< - 100, tức là ra ngoài khoảng Map (Vì Map có độ rộng là 100). Khi đó gán cho plane 1= 0, tức là setup về vị trí ban đầu và máy bay sẽ xuất hiện với tọa độ giống với máy bay của người chơi.

Kiểm tra nếu theo trục Oy giá trị tuyệt đối y của máy bay vật cản - giá trị y của máy bay người chơi < 8 thì hai máy bay sẽ chạm nhau. Đồng thời start = 0 và gv = 1 in ra màn hình văn bản “GAME OVER” và thông báo điểm đến người chơi.

* *Tọa độ của máy bay vật cản ở giữa*

Thiết kế tương tự máy bay vật cản sau cùng. Truyền vào tọa độ (x, y).

*Hình 2.24. Thiết kế máy bay vật cản ở giữa*

Kết quả:

Ta có plane2 giảm 1 đơn vị. Nếu plane2 < - 100, tức là ra ngoài khoảng Map (Vì Map có độ rộng là 100). Khi đó gán cho plane2 = 0, tức là setup về vị trí ban đầu và máy bay sẽ xuất hiện với tọa độ giống với máy bay của người chơi.

Kiểm tra nếu theo trục Oy giá trị tuyệt đối y của máy bay vật cản - giá trị y của máy bay người chơi < 8 thì hai máy bay sẽ chạm nhau, theo trục 0x cũng tương tự. Đồng thời start = 0 và gv = 1 in ra màn hình văn bản “GAME OVER” và thông báo điểm đến người chơi.

* *Tọa độ của máy bay vật cản đầu tiên*

Thiết kế tương tự máy bay vật cản sau cùng và máy bay vật cản ở giữa. Truyền vào tọa độ (x, y).

*Hình 2.24. Thiết kế máy bay vật cản đầu tiên*

Kết quả:

### *Thiết kế tọa độ mây*

*Hình 2.24. Thiết kế mây*

**Chức năng của hàm:**

* Thiết kế mây bằng câu lệnh vẽ hình đa giác: **glBegin(GL\_POLYGON)**
* Truyền tham số tọa độ (x, y).

Kết quả:

Ta xét may1 sẽ bằng máy bay vật cản sau cùng, may3 bằng máy bay vật cản ở giữa. Và viết hàm cố định mây không cho di chuyển theo máy bay của người chơi.

### *Thiết kế tọa độ núi*

*Hình 2.25. Thiết kế núi*

Kết quả chạy chương trình:

### *Xuất score, speed và level*

*Hình 2.26. Xuất điểm, tốc độ và level*

**Chức năng của hàm:**

* Xuất tốc độ: Viết hàm kiểm tra khi điểm > 40 thì level = score/ 40 và FPS = FPS + 2.

Kết quả chạy chương trình:

### Hàm tính điểm và thông báo “GAME OVER”

*Hình 2.27. Tính điểm và thông báo Game Over*

**Chức năng của hàm:** Nếu gv = 1 thì chương trình sẽ in ra màn hình văn bản “GAME OVER” tại vị trí (75, 38) . Đồng thời thông báo điểm đến người chơi tại vị trí x = 75 và y = 42.

Kết quả chạy:

### Hàm Main

Chức năng của hàm là gọi các hàm có trong chương trình và đây là nơi mà chương trình sẽ khởi chạy đầu tiên.

# **KẾT LUẬN**

Qua việc nghiên cứu, tìm hiểu và thực hiện nghiên cứu đề tài “Thiết kế Game hiển thị ở dạng 2D hoặc 3D”. Nhóm chúng em đã đã xây dựng thành công “Game 2D Airplane Flight Game” .

Về mặt lý thuyết, đồ án trình bày một số nghiên cứu về OpenGL. Qua việc nghiên cứu đồ án này, chúng em đã phần nào củng cố được kiến thức về OpenGL và nắm được cách làm thể nào để xây dựng, thiết kế Game 2D hoặc 3D.