**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

A blue and white logo

AI-generated content may be incorrect.

**BÁO CÁO BÀI TẬP LỚN**

**MÔN NHẬP MÔN LẬP TRÌNH**

**ĐỀ TÀI: MÔ PHỎNG CƠ CẤU CƠ KHÍ ĐẢO CHIỀU CHUYỂN ĐỘNG THANH RĂNG BẰNG HỆ THỐNG BÁNH RĂNG**

**Họ và tên: Nguyễn Ngọc Phú**

**Mã sinh viên: 24021181**

**Giảng Viên: TS.Trần Thanh Hải**

**Lớp: K69M** - **MT3**

**Mã học phần: INT1008**-**2**

**Ngành: Công nghệ kỹ thuật Cơ - Điện tử**

**HÀ NỘI – 2025**

**Mục Lục**

[**Tóm tắt** 3](#_Toc198894190)

[**Lời cảm ơn** 4](#_Toc198894191)

[**Phần 1: Mở đầu** 5](#_Toc198894192)

[**1.1. Giới thiệu về Code::Blocks** 5](#_Toc198894193)

[**1.1.1. Giới thiệu Code::Blocks.** 5](#_Toc198894194)

[**1.1.2. Lý do lựa chọn Code::Blocks** 6](#_Toc198894195)

[**1.2. Giới thiệu về OpenGL** 6](#_Toc198894196)

[**1.2.1. Giới thiệu OpenGL** 6](#_Toc198894197)

[**1.2.2 Lý do lưa chọn thư viện đồ họa OpenGL** 7](#_Toc198894198)

[**1.3. Lý do lựa chọn đề tài.** 7](#_Toc198894199)

[**Phần 2: Phân tích thiết kế chương trình** 8](#_Toc198894200)

[**2.1. Sơ đồ khối** 8](#_Toc198894201)

[**2.3. Hàm sử dụng trong chương trình** 13](#_Toc198894202)

[**2.3.1. Hàm có sẵn trong thư viện hệ thống** 13](#_Toc198894203)

[**2.3.2. Hàm tự định nghĩa trong chương trình** 17](#_Toc198894204)

[**Phần 3: Hướng dẫn** 18](#_Toc198894205)

[**3.1. Hướng dẫn cài đặt Code::Blocks** 18](#_Toc198894206)

[**3.2. Hướng dẫn cài đặt OpenGL cho Code::Blocks** 21](#_Toc198894207)

[**3.3. Hướng dẫn sử dụng chương trình** 25](#_Toc198894208)

[**Phần 4: Kết luận và phương hướng phát triển** 31](#_Toc198894209)

[**4.1. Kết luận chung** 31](#_Toc198894210)

[**4.2. Ý nghĩa** 31](#_Toc198894211)

[**4.3. Tính năng nổi bật** 32](#_Toc198894212)

[**4.4. Một số hạn chế** 32](#_Toc198894213)

[**4.5. Kiểm nghiệm** 33](#_Toc198894214)

[**4.6. Phương hướng phát triển** 34](#_Toc198894215)

# **Tóm tắt**

Dự án sử dụng phần mềm Code::Blocks và thư viện OpenGL để hỗ trợ hiển thị rõ ràng mô phỏng thành cơ khí bao gồm bánh răng và thanh răng trong không gian ba chiều. Dự án đã triển khai logic để mô phỏng chuyển động quay của bánh răng dẫn động và sự tương tác của nó với thanh răng, tạo ra chuyển động tịnh tiến. Mô hình đã thể hiện khả năng đảo chiều chuyển động của thanh răng thông qua việc thay đổi chiều quay của bánh răng dẫn động. Ngoài ra, dự án còn có bảng điều khiển có thể đóng mở và còn sử dụng các nút bấm, thanh trượt để tinh chỉnh các thông số kỹ thuật, màu sắc của mô hình. Không gian 3D để hỗ trợ hiển thị, quan sát từng góc, quan sát kĩ các chuyển động, va chạm của các bánh răng và thanh răng. Kết quả mô phỏng định tính cho thấy sự phù hợp với nguyên lý hoạt động cơ bản của cơ cấu đảo chiều thanh răng.

# **Lời cảm ơn**

Để hoàn thành báo cáo “ Mô phỏng cơ cấu cơ khí đảo chiều thanh răng bằng hệ thống bánh răng” em xin được bày tỏ lòng biết ơn sâu sắc đến giảng viên thầy Trần Thanh Hải đã truyền đạt cho em những kiến thức, kỹ năng cần thiết để hoàn thành bài báo cáo một cách trọn vẹn. Em xin gửi những lời chúc tốt đẹp nhất tới người thầy - người dẫn dắt em. Cảm ơn thầy vì đã kiên nhẫn, đôn đốc, đề ra những phương hướng đóng góp cho cá nhân em cũng như góp một phần công sức lớn cho quá trình hoàn thiện bài dự án. Nhờ sự hướng dẫn tận tâm của thầy, em đã có cơ hội tiếp cận và ứng dụng các kiến thức về mô phỏng cơ cấu cơ khí vào thực tiễn, từ đó nâng cao kỹ năng lập trình và tư duy hệ thống. Sự giúp đỡ của thầy chính là đòn bẩy đưa em phát triển tiến tới một bài dự án hoàn chỉnh như bây giờ. Vì vậy mà em rất mong sẽ nhận được những lời nhận xét, góp ý chân thành từ phía thầy để hoàn thành bài báo cáo một cách tốt hơn. Một lần nữa, chúng em xin gửi lời cảm ơn chân thành nhất đến thầy và mong muốn sẽ tiếp tục nhận được sự hỗ trợ, chỉ dẫn quý báu từ thầy trong những chặng đường học tập và nghiên cứu tiếp theo.

## 

## **Phần 1: Mở đầu**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

## **1.1. Giới thiệu về Code::Blocks**

### **1.1.1. Giới thiệu Code::Blocks.**

Code::Blocks là một môi trường phát triển tích hợp (IDE) miễn phí, mã nguồn mở và đa nền tảng. Nó được thiết kế để đáp ứng nhu cầu khắt khe nhất của người dùng khi lập trình bằng các ngôn ngữ C, C++ và Fortran.

Một số đặc điểm nổi bật của Code::Blocks:

* Miễn phí và mã nguồn mở: Bạn có thể tải xuống, sử dụng và thậm chí sửa đổi mã nguồn của Code::Blocks hoàn toàn miễn phí.
* Đa nền tảng: Code::Blocks hoạt động trên các hệ điều hành Windows, macOS và Linux, cho phép bạn phát triển ứng dụng trên nhiều môi trường khác nhau.
* Hỗ trợ nhiều trình biên dịch: Code::Blocks hỗ trợ nhiều trình biên dịch khác nhau như GCC, Clang và Visual C++, mang lại sự linh hoạt cho người dùng.
* Hệ thống xây dựng tùy chỉnh: Code::Blocks có một hệ thống xây dựng tùy chỉnh rất nhanh, không yêu cầu tệp tin makefile. Nó cũng hỗ trợ xây dựng song song để tận dụng tối đa sức mạnh của bộ vi xử lý đa nhân.
* Hỗ trợ gỡ lỗi mạnh mẽ: Code::Blocks cung cấp các công cụ gỡ lỗi mạnh mẽ với hỗ trợ đầy đủ cho các điểm dừng (breakpoints), theo dõi biến (watch variables), và dấu vết ngăn xếp (call stack).
* Giao diện tùy biến cao: Bạn có thể tùy chỉnh giao diện của Code::Blocks để phù hợp với sở thích và quy trình làm việc của mình.
* Hỗ trợ plugin: Code::Blocks được xây dựng dựa trên kiến trúc plugin, cho phép bạn mở rộng chức năng của nó bằng cách cài đặt hoặc tự phát triển các plugin.
* Tích hợp với hệ thống quản lý phiên bản: Code::Blocks tích hợp tốt với các hệ thống quản lý phiên bản phổ biến như Git và Subversion.
* Hỗ trợ cú pháp nổi bật và tự động hoàn thành mã: Giúp bạn viết mã nhanh hơn và giảm thiểu lỗi chính tả.
* Trình duyệt lớp (Class Browser): Giúp bạn dễ dàng duyệt và quản lý các lớp và thành viên trong dự án của mình.

### **1.1.2. Lý do lựa chọn Code::Blocks**

Code::Blocks là một môi trường phát triển tích hợp (IDE) mạnh mẽ, miễn phí, linh hoạt, đa nền tảng và mã nguồn mở, hỗ trợ rất tốt cho ngôn ngữ lập trình C/C++. Code::Blocks có giao diện trực quan, dễ sử dụng, phù hợp với cả người mới học và lập trình viên có kinh nghiệm. Công cụ này hỗ trợ nhiều trình biên dịch như GCC, MinGW và Visual C++, giúp tôi dễ dàng cấu hình và biên dịch chương trình trên nhiều nền tảng. Ngoài ra, Code::Blocks còn hỗ trợ quản lý dự án tốt, có tính năng kiểm tra cú pháp, giúp quá trình lập trình hiệu quả hơn. Đặc biệt, khi làm việc với đồ họa hoặc lập trình hệ thống như OpenGL, SDL, hay thư viện graphics.h, Code::Blocks cho phép dễ dàng tích hợp thư viện ngoài và quản lý chúng trong cùng một môi trường.

## **1.2. Giới thiệu về OpenGL**

### **1.2.1. Giới thiệu OpenGL**

OpenGL (Open Graphics Library) là một thư viện đồ họa tiêu chuẩn, được sử dụng rộng rãi để xây dựng các ứng dụng đồ họa 2D và 3D. Với khả năng hoạt động đa nền tảng và hiệu suất cao nhờ tận dụng GPU, OpenGL cho phép lập trình viên mô phỏng các đối tượng, hiệu ứng ánh sáng, chuyển động và môi trường 3D một cách chân thực. Đây là nền tảng được nhiều nhà phát triển game, ứng dụng mô phỏng kỹ thuật, và phần mềm đồ họa lựa chọn nhờ tính linh hoạt và hỗ trợ mạnh mẽ từ cộng đồng.

Một số đặc điểm nổi bật của thư viện đồ họa OpenGL:

* Đa nền tảng: Chạy được trên nhiều hệ điều hành như Windows, Linux, macOS mà không cần thay đổi mã nguồn nhiều.
* Độc lập với phần cứng: OpenGL trừu tượng hóa phần cứng đồ họa, giúp lập trình viên không cần quan tâm đến chi tiết kỹ thuật của từng loại GPU.
* Hiệu suất cao: Tận dụng GPU để xử lý đồ họa song song, tăng tốc độ hiển thị hình ảnh và hoạt ảnh 3D.
* Hỗ trợ đồ họa 2D và 3D: Từ vẽ hình cơ bản đến mô hình 3D phức tạp, ánh sáng, bóng đổ, hiệu ứng vật liệu...
* Tương thích với nhiều thư viện bổ trợ:
* Dễ dàng tích hợp với GLUT, GLFW, SDL, GLM, Assimp, stb\_image... để phát triển ứng dụng phức tạp.

### **1.2.2 Lý do lưa chọn thư viện đồ họa OpenGL**

Việc lựa chọn thư viện đồ họa OpenGL kết hợp với ngôn ngữ lập trình C++ trong môi trường phát triển Code::Blocks vì OpenGL là một tiêu chuẩn đồ họa đa nền tảng, cung cấp khả năng truy cập trực tiếp vào phần cứng đồ họa, từ đó cho phép xây dựng các ứng dụng mô phỏng và hiển thị hình ảnh 2D, 3D với hiệu suất cao và độ chính xác lớn. Ngôn ngữ C++ được sử dụng rộng rãi trong các lĩnh vực yêu cầu hiệu năng, đặc biệt là trong các hệ thống mô phỏng và đồ họa máy tính, nhờ khả năng quản lý bộ nhớ hiệu quả và hỗ trợ lập trình hướng đối tượng. Bên cạnh đó, Code::Blocks là một môi trường phát triển tích hợp (IDE) mã nguồn mở, hỗ trợ tốt việc biên dịch và tổ chức mã nguồn C++, đồng thời có khả năng tích hợp thư viện bên ngoài một cách linh hoạt. Sự kết hợp này không chỉ đáp ứng yêu cầu kỹ thuật của dự án mà còn tạo điều kiện thuận lợi trong quá trình phát triển, kiểm thử và mở rộng phần mềm.

## **1.3. Lý do lựa chọn đề tài.**

Trong ngành Cơ-Điện tử, đặc biệt là lĩnh vực cơ khí chế tạo máy, các cơ cấu truyền động như bánh răng và thanh răng đóng vai trò then chốt trong việc biến đổi chuyển động, truyền lực và điều khiển phương – chiều của lực một cách linh hoạt, chính xác. Một cơ cấu tiêu biểu được ứng dụng rộng rãi trong các máy công cụ như máy tiện, máy khoan, máy CNC hay các hệ thống truyền động tuyến tính là cơ cấu đảo chiều chuyển động của thanh răng bằng hệ thống bánh răng.

Cơ cấu này hoạt động dựa trên sự ăn khớp giữa nhiều bánh răng với thanh răng, trong đó các bánh răng trung gian được bố trí theo nguyên lý cho phép đảo ngược chiều chuyển động tịnh tiến của thanh răng một cách chính xác và hiệu quả. Tuy nhiên, việc hiểu rõ nguyên lý hoạt động của cơ cấu này thường gặp khó khăn nếu chỉ tiếp cận qua lý thuyết hoặc bản vẽ kỹ thuật 2D phẳng. Nhận thấy sự thực tế khi quan sát các cơ cấu rất cần thiết trong học tập, em lựa chọn thực hiện đề tài mô phỏng cơ cấu đảo chiều của thanh răng bằng hệ thống bánh răng trong OpenGL C++ nhằm tạo ra một mô hình 3D sinh động, hỗ trợ học tập và nghiên cứu kỹ thuật cơ khí.

Thông qua đề tài này, em không chỉ mong muốn nâng cao kiến thức chuyên môn về cơ cấu truyền động trong cơ khí mà còn phát triển tư duy mô phỏng kỹ thuật số, kết nối lý thuyết cơ học với công nghệ mô phỏng hiện đại, góp phần cho việc xây dựng các ứng dụng kỹ thuật có tính trực quan và tương tác cao trong tương lai.

# **Phần 2: Phân tích thiết kế chương trình**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## **2.1. Sơ đồ khối**

Dưới đây là một sơ đồ khối mô tả một thuật toán tương tác, liên quan đến việc điều khiển hệ thống bánh răng thông qua cả bàn phím và chuột để khởi động mô phỏng cơ cấu cơ khí đảo chiều chuyển động thanh răng.

**2.1.1. Sơ đồ khối tổng quát**

|  |
| --- |
|  |

**2.1.2. Sơ đồ khối xử lí bàn phím**

|  |
| --- |
| **A diagram of a flowchart  AI-generated content may be incorrect.** |

**2.1.3. Sơ đồ khối xử lí chuột.**

|  |
| --- |
| **A diagram of a flowchart  AI-generated content may be incorrect.** |

**2.2. Thư viện sử dụng trong chương trình**

* GL/glut.h: Đây là thư viện tiêu đề cho OpenGL Utility Toolkit (GLUT). GLUT là một bộ công cụ cửa sổ hệ thống độc lập, giúp bạn dễ dàng tạo và quản lý cửa sổ, xử lý sự kiện (như bàn phím và chuột), và vẽ đồ họa OpenGL một cách đơn giản. Nó thường được sử dụng cho việc học tập và tạo mẫu nhanh các ứng dụng OpenGL.
* GL/gl.h: Đây là thư viện tiêu đề cốt lõi của OpenGL. Nó chứa các khai báo cho các hàm OpenGL, các hằng số và các kiểu dữ liệu cần thiết để vẽ đồ họa 2D và 3D. Bất kỳ chương trình OpenGL nào cũng cần thư viện này.
* cmath: Đây là thư viện tiêu đề chuẩn của C++ cung cấp các hàm toán học cơ bản như hàm lượng giác (sin, cos, tan), hàm mũ (pow, exp), hàm logarit (log), hàm làm tròn (floor, ceil) và nhiều hàm khác.
* vector: Đây là một phần của Standard Template Library (STL) trong C++. Nó định nghĩa một lớp template std::vector, đại diện cho một mảng động có thể thay đổi kích thước. vector rất hữu ích khi bạn cần một cấu trúc dữ liệu có thể chứa một dãy các phần tử và có thể dễ dàng thêm hoặc xóa các phần tử.
* cstdio: Đây là thư viện tiêu đề chuẩn của C (và cũng có trong C++) cung cấp các hàm liên quan đến nhập/xuất dữ liệu theo kiểu C, chẳng hạn như printf, scanf, fopen, fclose, fprintf, fscanf.
* string: Đây là một phần của STL trong C++. Nó định nghĩa lớp std::string, cung cấp một cách mạnh mẽ và tiện lợi để làm việc với chuỗi ký tự so với kiểu chuỗi ký tự kiểu C (char[]). Lớp string hỗ trợ nhiều thao tác như nối chuỗi, tìm kiếm, thay thế.
* cstring: Đây là thư viện tiêu đề chuẩn của C (và cũng có trong C++) cung cấp các hàm để thao tác với các chuỗi ký tự kiểu C (null-terminated byte strings), chẳng hạn như strcpy, strcat, strlen, strcmp.
* windows.h: Đây là một thư viện tiêu đề dành riêng cho hệ điều hành Windows. Nó chứa các khai báo cho hầu hết các hàm, kiểu dữ liệu, cấu trúc và hằng số cần thiết để tương tác với Windows API (Application Programming Interface). Nó cho phép bạn thực hiện các tác vụ liên quan đến hệ điều hành như tạo cửa sổ, quản lý tiến trình, làm việc với hệ thống tệp, đồ họa (bao gồm cả một số API đồ họa cũ hơn).
* mmsystem.h: Đây cũng là một thư viện tiêu đề dành riêng cho Windows. Nó chứa các khai báo cho các hàm liên quan đến đa phương tiện (multimedia) trong Windows, chẳng hạn như phát và ghi âm thanh, làm việc với các thiết bị MIDI.

## **2.3. Hàm sử dụng trong chương trình**

### **2.3.1. Hàm có sẵn trong thư viện hệ thống**

**2.3.1.1. Trong thư viện GL/glut.h**

* Hàm khởi tạo và cửa sổ:
* glutInit(): Khởi tạo GLUT
* glutInitDisplayMode(): Thiết lập chế độ hiển thị (double buffer, RGB, depth)
* glutInitWindowSize(): Đặt kích thước cửa sổ ban đầu
* glutInitWindowPosition(): Đặt vị trí cửa sổ ban đầu
* glutCreateWindow(): Tạo cửa sổ với tiêu đề cho trước
* glutFullScreen(): Chuyển sang chế độ toàn màn hình
* glutReshapeWindow(): Thay đổi kích thước cửa sổ\
* Hàm callback
* glutDisplayFunc(): Đặt hàm callback để vẽ cảnh
* glutReshapeFunc(): Đặt hàm callback khi thay đổi kích thước cửa sổ
* glutMouseFunc(): Đặt hàm callback xử lý sự kiện chuột
* glutMotionFunc(): Đặt hàm callback khi di chuyển chuột với nút nhấn
* glutPassiveMotionFunc(): Đặt hàm callback khi di chuyển chuột không nhấn nút
* glutKeyboardFunc(): Đặt hàm callback xử lý bàn phím
* glutTimerFunc(): Đặt hàm callback gọi hàm sau một thời gian
* Hàm vẽ và quản lý bộ nhớ đệm
* glutSwapBuffers(): Hoán đổi bộ nhớ đệm (cho double buffering)
* glutPostRedisplay(): Yêu cầu vẽ lại cảnh
* Hàm vẽ hình cơ bản
* glutSolidSphere(): Vẽ hình cầu đặc
* glutBitmapCharacter(): Vẽ ký tự bitmap
* Hàm từ GLU (OpenGL Utility)
* gluPerspective(): Thiết lập phép chiếu phối cảnh
* gluLookAt(): Thiết lập vị trí camera
* gluOrtho2D(): Thiết lập phép chiếu trực giao 2D
* gluNewQuadric(): Tạo đối tượng quadric mới
* gluCylinder(): Vẽ hình trụ/cone
* gluDisk(): Vẽ đĩa (hình tròn)
* gluDeleteQuadric(): Xóa đối tượng quadric
* gluQuadricNormals(): Thiết lập các vector pháp tuyến cho quadric
* Hàm quản lý trạng thái
* glutMainLoop(): Bắt đầu vòng lặp chính của GLUT

**2.3.1.2. Trong thư viện GL/gl.h**

* Hàm Quản Lý Trạng Thái & Khởi Tạo
* glClearColor(): Đặt màu xóa màn hình (background)
* glEnable() / glDisable(): Bật/tắt các tính năng OpenGL (GL\_DEPTH\_TEST, GL\_LIGHTING, GL\_LIGHT0, ...)
* glShadeModel(): Chọn chế độ tô bóng (GL\_SMOOTH hoặc GL\_FLAT)
* glIsEnabled(): Kiểm tra xem một tính năng có đang bật không (vd: glIsEnabled(GL\_LIGHTING))
* Hàm Vẽ Hình Học Cơ Bản
* glBegin()/glEnd(): Bắt đầu/kết thúc một khối lệnh vẽ cơ bản (GL\_QUADS, GL\_TRIANGLES, GL\_LINES,…)
* glVertex2f() / glVertex3f(): Định nghĩa tọa độ điểm (2D/3D)
* glNormal3f(): Đặt vector pháp tuyến (cho lighting)
* glColor3f() / glColor4f(): Đặt màu sắc (RGB/RGBA)
* glLineWidth(): Đặt độ dày nét vẽ
* Hàm Chiếu & Ma Trận
* glMatrixMode(): Chọn ma trận hiện tại (GL\_PROJECTION, GL\_MODELVIEW)
* glLoadIdentity(): Đặt ma trận hiện tại thành ma trận đơn vị
* glPushMatrix() / glPopMatrix(): Lưu/khôi phục ma trận hiện tại
* glRotatef(): Xoay đối tượng theo một trục
* glTranslatef(): Dịch chuyển đối tượng
* glViewport(): Đặt vùng hiển thị (viewport)
* Hàm thiết lập ánh sáng và vật liệu
* glLightfv(): Đặt thông số ánh sáng (vị trí, màu sắc)
* glMaterialfv(): Đặt thuộc tính vật liệu (ambient, diffuse, specular, shininess)
* glNormal3f(): Đặt vector pháp tuyến (cho tính toán lighting)
* Hàm bộ đệm & Hiển Thị
* glClear(): Xóa bộ đệm (GL\_COLOR\_BUFFER\_BIT, GL\_DEPTH\_BUFFER\_BIT)
* glFlush(): Buộc OpenGL thực thi các lệnh đang chờ
* glReadPixels(): Đọc dữ liệu pixel từ bộ đệm hiển thị
* Hàm vẽ 2D
* glRasterPos2f(): Đặt vị trí vẽ raster (cho text/bitmap)
* glBitmapCharacter(): Vẽ ký tự bitmap (thường dùng với glutBitmapCharacter)
* Hàm kiểm tra độ sâu và pha trộn màu
* glDepthFunc(): Đặt hàm kiểm tra độ sâu (mặc định là GL\_LESS)
* glEnable(GL\_DEPTH\_TEST): Bật kiểm tra độ sâu (Z-buffer)
* glEnable(GL\_BLEND): Bật blending (pha trộn màu)

**2.3.1.3. Trong thư viện cmath**

* Các hàm lượng giác: sin(), cos(),…
* Các hàm lũy thừa và căn bậc 2: sqrt(), pow(),…
* Hàm giá trị tuyệt đối: fabs()
* Hàm làm tròn và giới hạn giá trị: std::max() / std::min() (từ <algorithm>, nhưng thường dùng cùng <cmath>)

**2.3.1.4. Trong thư viện vector**

* Hàm push\_back(): thêm một phần tử mới vào cuối của container
* Hàm size(): Lấy số lượng phần tử trong vector
* Hàm clear(): Xóa tất cả phần tử trong vector

**2.3.1.5. Trong thư viện cstdio**

* Hàm sprintf: sử dụng để định dạng chuỗi và lưu vào một mảng ký tự
* Hàm std::string (từ C++ string, không phải từ cstdio nhưng thường dùng cùng): sử dụng rộng rãi để xử lý chuỗi
* Hàm strcmp(): sử dụng để so sánh chuỗi
* Hàm strstr(): sử dụng để tìm chuỗi con trong chuỗi

**2.3.1.6. Trong thư viện string**

🡪 Chỉ dùng string để khai báo các chuỗi thông tin ( std::string …)

**2.3.1.7. Trong thư viện cstring**

* Hàm strlen(): sử dụng để tính độ dài của một chuỗi ký tự. Độ dài này được xác định bằng số lượng ký tự có trong chuỗi trước ký tự null kết thúc chuỗi ('\0').
* Hàm sprintf(): sử dụng để định dạng chuỗi và lưu vào một mảng ký tự

**2.3.1.8. Trong thư viện windows.h kết hợp mmsystem.h**

🡪 Mục đích để xử lí âm thanh của chương trình

Ví dụ:

Phát âm thanh lặp lại (loop): PlaySound(TEXT("gears-18005.wav"), NULL, SND\_FILENAME | SND\_ASYNC | SND\_LOOP):

Dừng âm thanh: PlaySound(NULL, NULL, 0):

\*) Giải thích các tham số

* SND\_FILENAME: Âm thanh từ file.
* SND\_ASYNC: Phát không đồng bộ (không chờ âm thanh kết thúc).
* SND\_LOOP: Lặp lại âm thanh.

### **2.3.2. Hàm tự định nghĩa trong chương trình**

* Hàm init(): Khởi tạo các tham số đồ họa, ánh sáng, vật liệu và vị trí ban đầu của các bánh răng.
* Hàm display(): Hàm render chính, vẽ toàn bộ cảnh 3D và giao diện người dùng.
* Hàm reshape(): Xử lý thay đổi kích thước cửa sổ, điều chỉnh tỷ lệ khung hình.
* Hàm mouse(): Xử lý sự kiện chuột (click, di chuyển).
* Hàm motion(): Theo dõi di chuyển chuột khi nhấn giữ (xoay camera, zoom).
* Hàm passiveMotion(): Theo dõi di chuột không nhấn (hiển thị tooltip).
* Hàm keyboard(): Xử lý sự kiện bàn phím (start/stop, reset, fullscreen...).
* Hàm update(): Hàm timer, cập nhật hoạt ảnh và trạng thái hệ thống mỗi khung hình.
* Hàm resetSimulation(): Đặt lại tất cả thông số về trạng thái ban đầu.
* Hàm controlSound(): Điều khiển âm thanh động cơ bánh răng.
* Hàm playClickSound(): Phát âm thanh khi click chuột.
* Hàm drawGear(): Vẽ một bánh răng với các thông số tùy chỉnh.
* Hàm drawBox(): Vẽ hình hộp (phần thân cố định mô hình).
* Hàm drawGround(): Vẽ mặt đất với lưới tọa độ.
* Hàm drawCylinderEndBox():  Vẽ khối hộp chữ nhật nối với các bánh răng.
* Hàm drawCylinder():Vẽ hình trụ (phần trục quay).
* Hàm drawCircle():Vẽ hình tròn (phần trục quay).
* Hàm drawSideCylinder(): Vẽ hình trụ nằm ngang (dùng cho trục quay hoặc bộ phận kết nối).
* Hàm drawRack(): Vẽ thanh răng với các răng hình thang.
* Hàm drawRackBevel(): Vẽ phần vát mép (bevel) của thanh răng để tăng tính thực tế.
* Hàm drawTrapezoidalTooth(): Vẽ một răng hình thang của thanh răng (rack) với đầy đủ chi tiết 3D.
* Hàm drawRotatingComponents(): Vẽ tổ hợp các bộ phận chuyển động.
* Hàm drawUI(): Vẽ bảng điều khiển chính.
* Hàm drawHelp(): Hiển thị hướng dẫn các phím tắt.
* Hàm drawSlider(): Vẽ các thanh trượt điều chỉnh thông số.
* Hàm drawColorSlider(): Vẽ các thanh trượt màu sắc.
* Hàm drawArrow(): Vẽ mũi tên trên bảng điều khiển.
* Hàm drawPowerButton(): Vẽ nút nguồn có hiệu ứng ấn.
* Hàm drawSoundButton(): Vẽ nút âm thanh có hiệu ứng ấn.
* Hàm drawGearTooltip(): Hiển thị chú thích khi di chuột qua bánh răng.
* float lerp(): Hàm nội suy tuyến tính cho animation mượt mà.
* bool isPointInRect(): Kiểm tra điểm có trong hình chữ nhật không.
* Hàm calculateNormal(): Tính vector pháp tuyến cho lighting.
* Hàm drawBeveledEdge(): Vẽ cạnh vát cho chi tiết 3D.
* Hàm drawTooth(): Vẽ một răng bánh răng với hiệu ứng 3D.
* Hàm updateRotationSpeed(): Điều chỉnh tốc độ quay mượt mà.
* Hàm updateRackPosition(): Cập nhật vị trí thanh răng dựa trên tốc độ quay.

# **Phần 3: Hướng dẫn**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## **3.1. Hướng dẫn cài đặt Code::Blocks**

Bước 1: Tải file codeblocks-25.03mingw-setup.exe tại [link tải Code::Blocks](https://www.codeblocks.org/downloads/binaries/)

|  |  |
| --- | --- |
| File này 🡪 |  |

Bước 2: Sau khi đã tải xong thì mở file setup.

|  |
| --- |
|  |
| Ấn Next |

|  |
| --- |
| Ấn I Agree |

|  |
| --- |
| Ấn Next |

Bước 3: Chọn vị trí muốn lưu thư mục Code::Blocks

|  |
| --- |
| Ấn Install  Nếu muốn thay đổi vị trí lưu trữ thì Ấn Browse 🡪 Chọn vị trí muốn lưu 🡪 Ấn Install |

## **3.2. Hướng dẫn cài đặt OpenGL cho Code::Blocks**

Sau khi đã cài đặt được Code::Blocks thì ta tiến hành cài đặt thư viện OpenGL:

Bước 1: tiến hành tải các thư viện và dự án từ github ([Link tải](https://github.com/Novphu/CodeBlocks_OpenGL_Gear_and_Rack))

Sau khi đã ấn vào link github

**A screenshot of a computer

AI-generated content may be incorrect.**

Sau đó chọn vào dòng code:

|  |  |  |
| --- | --- | --- |
|  | **🡪** |  |

🡪 Sau đó Nhấn Download ZIP để tải về các thư viện và dự án.

Bước 2: Giải nén file ZIP vừa tải.

Bước 3: Mở Code::Blocks

**A computer screen shot of a computer

AI-generated content may be incorrect.**

Bước 5: Chọn Settings

**A screenshot of a computer

AI-generated content may be incorrect.**

🡪 Chọn compiler

**A screenshot of a computer

AI-generated content may be incorrect.**

Bước 6: Chọn Linker settings

**A screenshot of a computer

AI-generated content may be incorrect.**

Bước 7: Thêm các thư viện

|  |  |  |
| --- | --- | --- |
| * libopengl32.a | * libglut32.a | * winmm |
| * libglu32.a | * libfreeglut.a |  |

**A screenshot of a computer

AI-generated content may be incorrect.**

🡪 Add các thư viện (trừ thư viện winmm ) từ …\CodeBlock\_OpenGL\_Gear\_and\_Rack\msys64\ucrt64\lib

Bước 8: Mở file dự án ( BAI\_TAP\_LON.cbp )

**A group of colorful cubes

AI-generated content may be incorrect.**

## **3.3. Hướng dẫn sử dụng chương trình**

**3.3.1. Thoát chương trình**

****

Ấn nút trên hoặc nút escape trên bàn phím.

**3.3.2. Tắt/Bật âm thanh chương trình**

Dùng chuột nhấn vào nút âm thanh hoặc phím M/m trên bàn phím.

|  |  |
| --- | --- |
| A yellow circle with white sound icon  AI-generated content may be incorrect. |  |
| Âm thanh đang bật. | Âm thanh đang tắt. |

**3.3.3. Đóng/Mở bảng điều khiển**

Dùng chuột nhấn vào mũi tên trên bảng điều khiển hoặc dùng phím U/u trên bàn phím.

|  |  |
| --- | --- |
|  |  |
| Đóng bảng điều khiển. | Mở bảng điều khiển. |

**3.3.4. Bật/Tắt chế độ toàn màn hình**

Dùng phím F/f trên bàn phím để bật/tắt chế độ toàn màn hình

|  |  |  |
| --- | --- | --- |
|  | Nhấn F/f 🡪 |  |

**3.3.5. Bật/Tắt chế độ Tooltip của chương trình.**

Dùng phím T/t trên bàn phím để bật/tắt chế độ tooltip.

|  |  |  |
| --- | --- | --- |
|  | Nhấn T/t 🡪 |  |

**3.3.6. Khởi động và dừng mô hình.**

* Dùng phím space trên bàn phím.
* Dùng chuột nhấp vào nút Start/Stop trên bảng điều khiển.

|  |  |
| --- | --- |
|  |  |
| Nhấn chuột để khởi động. | Nhấn chuột để dừng lại. |

* Dùng chuột nhấp vào các bánh răng của mô hình.

|  |  |  |
| --- | --- | --- |
|  | Sau khi nhấp mô hình sẽ quay 🡪 |  |

**3.3.7. Khôi phục các thông số ban đầu của mô hình.**

**A blue rectangle with white text

AI-generated content may be incorrect.**

Dùng chuột nhấp vào nút reset hoặc ấn phím R/r từ trên bàn phím.

**3.3.8. Đóng/Mở bảng hướng dẫn của chương trình.**

Dùng chuột nhấp vào nút Hide/Show Hints trên bảng điều khiển.

|  |  |
| --- | --- |
| **A screenshot of a computer game  AI-generated content may be incorrect.** |  |
| Đang mở | Đang đóng |

**3.3.9. Đóng/Mở bảng cài đặt các thông số của bánh răng.**

Dùng chuột nhấp vào các nút Gear 1(2,3,4) trên bảng điều khiển để mở bảng cài đặt.

|  |  |  |
| --- | --- | --- |
|  | Sau khi nhấp 1 lần 🡪 |  |

Dùng chuột nhấp vào các nút Gear 1(2,3,4) trên bảng điều khiển một lần nữa để đóng bảng cài đặt.

|  |  |  |
| --- | --- | --- |
|  | Sau khi nhấp lần 2 🡪 |  |

**3.3.10. Đóng/Mở bảng cài đặt các thông số của thanh răng.**

Dùng chuột nhấp vào các nút Rack Controls trên bảng điều khiển để mở bảng cài đặt.

|  |  |  |
| --- | --- | --- |
|  | Sau khi nhấp 1 lần 🡪 |  |
|  |  |  |

Dùng chuột nhấp vào các nút Rack Controls trên bảng điều khiển một lần nữa để đóng bảng cài đặt.

|  |  |  |
| --- | --- | --- |
|  | Sau khi nhấp lần 2 🡪 |  |

**3.3.11. Điều chỉnh các thông số trong các bảng cài đặt.**

Dùng chuột nhấp vào các thanh trượt thông số để điều chỉnh thông số của các bánh răng và thanh răng.

|  |  |  |
| --- | --- | --- |
|  | Sau khi nhấp thay đổi thanh trượt Teeth🡪 |  |

**3.3.12. Di chuyển hệ bánh răng và đảo chiều.**

* Dùng 2 phím A/D để di chuyển hệ bánh răng sang trái hoặc phái.
* Di chuyển sang phải:

|  |  |  |
| --- | --- | --- |
|  | Nhấn phím D 🡪 |  |
|  |  |  |

* Di chuyển sang Trái:

|  |  |  |
| --- | --- | --- |
|  | Nhấn phím A 🡪 |  |

* Dùng chuột giữa để di chuyển hệ bánh răng sang trái hoặc phải.
* Dùng chuột giữa giữ và kéo sang phải:

|  |  |  |
| --- | --- | --- |
|  | **🡪** |  |

* Dùng chuột giữa giữ và kéo sang trái:

|  |  |  |
| --- | --- | --- |
|  | **🡪** |  |

**3.3.13. Thay đổi góc nhìn mô hình**

Dùng chuột trái giữ và kéo theo chiều muốn nhìn mô hình:

|  |  |  |
| --- | --- | --- |
|  | **🡪** |  |

**3.3.14. Phóng to/Thu nhỏ mô hình.**

Dùng chuột phải giữ và kéo xuống để phóng to mô hình:

|  |  |  |
| --- | --- | --- |
|  | **🡪** |  |

Dùng chuột phải giữ và kéo lên để thu nhỏ mô hình:

|  |  |  |
| --- | --- | --- |
|  | **🡪** |  |

# **Phần 4: Kết luận và phương hướng phát triển**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sau khi hoàn thành xong “Mô phỏng cơ cấu cơ khi đảo chiều chuyển động thanh răng bằng hệ thống bánh răng” , bản thân em đã rút ra những kết luận như sau:

## **4.1. Kết luận chung**

Dự án đã đạt được các mục tiêu đề ra ban đầu bao gồm việc xây dưng hệ thống bánh răng, thanh răng, giao diện người dùng hài hòa và điều khiển chương trình bằng chuột và bàn phím. Quá trình phát triển đã củng cố kiến thức về các khái niệm cơ bản của OpenGL như: các hình cơ bản, các thuật toán để vẽ, các phép biến đổi hình học. Hệ thống hoạt động ổn định và chính xác, đáp ứng yêu cầu mô phỏng cơ bản với hiệu suất tốt. Ngoài ra, dự án còn làm rõ hơn về mối quan hệ toán học giữa chuyển động quay và chuyển động tịnh tiến trong cơ cấu thanh răng - bánh răng và cách áp dụng chúng vào mô phỏng.

## **4.2. Ý nghĩa**

Dự án mô phỏng cơ cấu đảo chiều thanh răng với hệ thống bánh răng bằng OpenGL C++ không chỉ là một bài tập kỹ thuật mà còn mang nhiều ý nghĩa thiết thực. Dự án đã cung cấp một công cụ trực quan mạnh mẽ để hiểu rõ nguyên lý hoạt động của một cơ cấu cơ khí quan trọng. Hơn nữa, dự án đã rèn luyện kỹ năng lập trình đồ họa và tư duy mô phỏng máy tính. Mô hình này có thể đóng góp vào việc phát triển các hệ thống cơ khí hiệu quả và trực quan hơn trong tương lai.

* Trực quan hóa các nguyên lý cơ học: Mô hình 3D sống động giúp người tìm hiểu dễ dàng hình dung và hiểu rõ nguyên lý hoạt động của cơ cấu đảo chiều thanh răng và hệ thống bánh răng, một khái niệm trừu tượng trong cơ học kỹ thuật.
* Mô phỏng tương tác người dùng: Dự án có thể phát triển thành một ứng dụng tương tác, cho phép người dùng thay đổi các thông số thiết kế (ví dụ: kích thước bánh răng, số răng) và quan sát trực tiếp ảnh hưởng của chúng đến chuyển động của cơ cấu.
* Nghiên cứu và phát triển các kỹ thuật mô phỏng: Dự án có thể là nền tảng để nghiên cứu và thử nghiệm các kỹ thuật mô phỏng tiên tiến hơn, chẳng hạn như tích hợp các hiệu ứng vật lý phức tạp (ma sát, va chạm), tối ưu hóa hiệu suất hiển thị cho các hệ thống lớn.

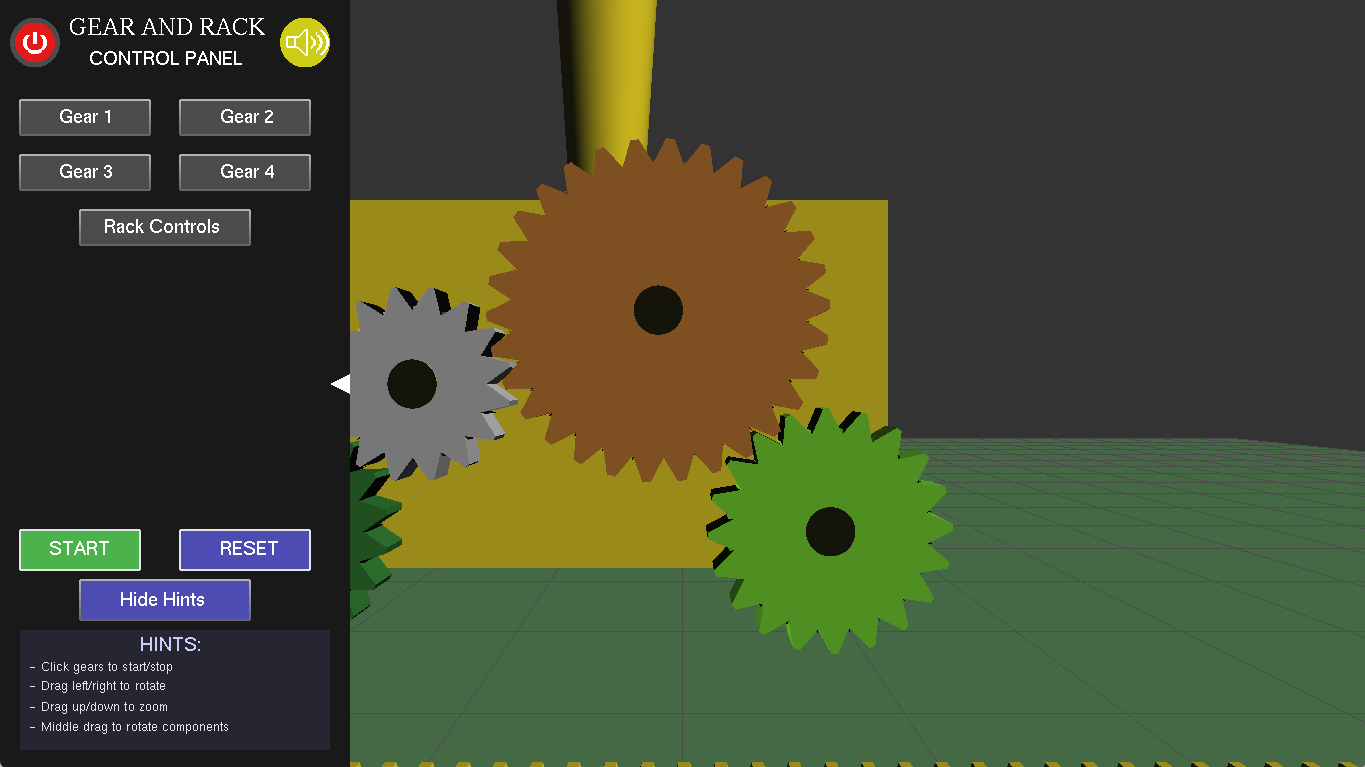
## **4.3. Tính năng nổi bật**

* Mô phỏng đa chiều
* Hệ thống 4 bánh răng với khả năng:
* Ăn khớp và truyền động theo tỉ số răng.
* Thay đổi số răng, bán kinh, độ dày.
* Tùy chỉnh màu sắc để dễ phân biệt.
* Thanh răng di chuyển tịnh tiến khi bánh răng ăn khớp cùng thanh răng.
* Các thành phần cơ khí phụ trợ như: trục, khung đỡ.
* Giao diện trực quan
* Bảng điều khiển (UI Panel) cho phép:
* Lựa chọn từng bánh răng để điều chỉnh các thông số kỹ thuật.
* Thay đổi tốc độ quay bằng thanh trượt.
* Khôi phục lại các thông số và trạng thái ban đầu.
* Chế độ xem 3D linh hoạt:
* Xoay camera bằng chuột trái.
* Phóng to/thu nhỏ bằng chuột phải.
* Xoay thành phần cơ khí bằng chuột giữa.
* Hiệu ứng âm thanh và hình ảnh:
* Chế độ chiếu sáng và vật liệu giúp mô hình trở lên sống động, thực tế hơn.
* Âm thanh mô phỏng tiếng bánh răng khi hoạt động.
* Âm thanh mô phỏng tiếng nhấn nút.
* Hiệu ứng chuyển động mượt mà.

## **4.4. Một số hạn chế**

* Âm thanh: Phụ thuộc vào PlaySound của Windows: Hàm PlaySound chỉ hoạt động trên Windows, không đa nền tảng, không tương thích trên Linux/macOS.
* Thiếu bóng đổ (shadow): Các đối tượng không có bóng đổ, làm giảm độ sâu trường ảnh.
* Phụ thuộc vào GLUT/GLU cũ: Một số hàm như gluCylinder đã lỗi thời

## **4.5. Kiểm nghiệm**

****

**A video game of a machine

AI-generated content may be incorrect.**

Kết quả mô phỏng cho thấy mô phỏng chính xác nguyên lý truyền động, sự ăn khớp giữa bánh răng dẫn động và thanh răng diễn ra tương đối mượt mà, giao diện trực quan, chi tiết, nhiều chức năng, dễ sử dụng, bố cục hợp lý, hài hòa với giao diện của chương trình, phản hồi tương tác nhanh, ít có dấu hiệu trễ, hiệu năng ổn định, mượt mà, đầy đủ chức năng điều chỉnh thông số, hiệu ứng âm thanh, nút bấm đặc sắc, hiệu ứng chuyển động thực tế. Hệ thống đạt được hiệu suất ổn định, độ trễ tương tác thấp, mức tiêu thụ tài nguyên thấp, thời gian cập nhật trạng thái nhanh đảm bảo cho việc mô hình phản ứng mượt mà với thời gian hoặc tương tác của người dùng, các thao tác điều khiển qua bàn phím được xử lý ngay lập tức , không có độ trễ đáng kể, thao tác chuột nhạy, tốc độ xử lí cao, tốc độ phản hồi nhanh.

## **4.6. Phương hướng phát triển**

Trong thời gian tới, em sẽ học tập, nghiên cứu và phát triển đề tài :

* Nâng cấp đồ họa: Sử dung GLFW + Glad ( Các thư viện tối ưu, hiện đại hơn) thay vì Glut.
* Tối ưu đồ họa, ánh sáng, phản xạ.
* Mở rộng tính năng: Cho phép người dùng thêm/xóa bánh răng động, ghi lại quá trình thành video.
* Tối ưu các hàm vật lí, xử lí chuyển động, thêm các điều kiện chuyển động phù hợp và thực tế hơn.
* Cải thiện giao diện trực quan: chuyển sang sử dụng thư viện ImGui, thêm tính năng giữ, kéo và thả để sắp xếp bánh răng.