

Dear ImGui Giới Thiệu



bạn biết dear imgui chứ



Có, mình biết **Dear ImGui**.

Dear ImGui (viết tắt của *Immediate Mode GUI*) là một thư viện giao diện người dùng (GUI) nhẹ và cực kỳ nhanh, thường được dùng trong các ứng dụng cần giao diện để debug, công cụ phát triển game, và các phần mềm kỹ thuật. Nó được viết bằng **C++** và có thể tích hợp dễ dàng với các engine đồ họa như OpenGL, DirectX, Vulkan...

Đặc điểm nổi bật:

- **Immediate Mode GUI:** Giao diện được vẽ lại hoàn toàn mỗi khung hình, rất phù hợp cho các công cụ realtime.
- **Hiệu suất cao:** Tối ưu cho các công cụ nội bộ, không cần đến animation phức tạp hay hiệu ứng UI như các GUI truyền thống.
- **Dễ tích hợp:** Có thể nhúng vào các engine hiện có, hoặc tự xây dựng hệ thống render tùy ý.
- **Không phụ thuộc vào hệ điều hành:** Bạn cần tự tích hợp đầu vào (mouse, keyboard) và render (OpenGL/DirectX...).

Một ví dụ đơn giản:

```
ImGui::Begin("Hello, world!");
ImGui::Text("This is some useful text.");
ImGui::SliderFloat("float", &myFloat, 0.0f, 1.0f);
if (ImGui::Button("Click me")) {
    // Xử lý khi nút được nhấn
}
```



```
}  
ImGui::End();
```

Bạn đang muốn dùng Dear ImGui cho mục đích gì? Debug game? Làm công cụ riêng? Hay tích hợp vào một engine nào đó (như Defold, Unity, SDL, v.v.)? Mình có thể giúp hướng dẫn chi tiết hơn.



tôi muốn bạn tạo 1 text editor giống hệt notepad, sử dụng imgui. Tôi cần bản chạy trên win 32 directX 11, visual studio 2022. Chỉ cần phần giữa ImGui::Begin và ImGui::End, vì tất cả phần cài đặt tích hợp khác tôi đều làm được rồi



Dưới đây là đoạn mã sử dụng Dear ImGui để tạo một **text editor giống hệt Notepad đơn giản**, chỉ gồm phần trong `ImGui::Begin()` và `ImGui::End()`. Đoạn mã này hỗ trợ:

- Gõ/chỉnh sửa nội dung văn bản
- Lưu trữ nội dung vào một `std::string`
- Cho phép kéo thanh cuộn khi nội dung dài
- Tên cửa sổ có thể là "Untitled - Notepad"



ImGui Text Editor - Đoạn code

```
static std::string textBuffer; // Giữ nội dung văn bản  
static ImGuiInputTextFlags flags = ImGuiInputTextFlags_AllowTabInput;
```



// Bạn có thể mở rộng với tính năng Save/Open nếu muốn

```
ImGui::Begin("Untitled - Notepad");
```

// Kích thước vùng text editor: fill toàn bộ cửa sổ

```
ImVec2 editorSize = ImGui::GetContentRegionAvail();
```

// Dùng InputTextMultiline cho editor

```
ImGui::InputTextMultiline("##TextEditor", &textBuffer, editorSize, flags);
```

```
ImGui::End();
```

✅ Tính năng hiện có:

- Cho phép nhập nhiều dòng văn bản
- Tự động cuộn nếu nội dung vượt quá khung
- Nhận cả phím Tab (indent code)

🔧 Gợi ý mở rộng:

- Thêm menu bar (``File``, ``Edit``, ``Help``) bằng ``ImGui::BeginMenuBar()``
- Hỗ trợ ``ctrl+s``, ``ctrl+o`` bằng cách hook phím
- Highlight cú pháp (nếu cần code editor, bạn nên dùng thêm lib như ImGuiColorTextEdit)

Nếu bạn muốn mình mở rộng thêm (ví dụ: lưu ra file `.txt`, mở file `.txt`, hoặc có chức năng Undo/Redo đơn giản), chỉ cần nói, mình sẽ thêm ngay.



có, thêm các chức năng đó đi



Tuyệt! Dưới đây là phiên bản **ImGui Notepad hoàn chỉnh hơn**, với các chức năng:

✅ Các tính năng mới:

- **Menu File:** New, Open, Save, Save As, Exit
- **Edit:** Undo, Redo, Cut, Copy, Paste, Select All (dựa trên clipboard)
- **Ctrl+S** và các tổ hợp phím tắt cơ bản
- Hỗ trợ load/save từ/to file `.txt`
- Undo/Redo cơ bản (vòng lặp stack đơn giản)

“🔗 Tất cả vẫn chỉ là phần trong `ImGui::Begin()` và `ImGui::End()` như bạn yêu cầu.”

🧠 Giới hạn:

- Undo/Redo cơ bản (stack vòng đơn giản)
- File dialog bạn sẽ cần tự tích hợp (VD: [tinyfiledialogs](#) hoặc Windows API)



Code bên trong `ImGui::Begin()` và `ImGui::End()`:

```
static std::string textBuffer;
static std::string filePath = "Untitled";
static ImGuiInputTextFlags flags = ImGuiInputTextFlags_AllowTabInput;

static std::vector<std::string> undoStack;
static std::vector<std::string> redoStack;
static bool textChanged = false;

// Menu Bar
if (ImGui::BeginMenuBar())
{
    if (ImGui::BeginMenu("File"))
    {
        if (ImGui::MenuItem("New", "Ctrl+N"))
        {
            undoStack.clear(); redoStack.clear();
            textBuffer.clear();
            filePath = "Untitled";
        }
        if (ImGui::MenuItem("Open...", "Ctrl+O"))
        {
            // TODO: Bạn cần tự tích hợp file dialog tại đây
            std::ifstream file("example.txt"); // hardcoded test
            if (file)
            {
                textBuffer.assign((std::istreambuf_iterator<char>(file)),
                                   std::istreambuf_iterator<char>());
                filePath = "example.txt";
                undoStack.clear(); redoStack.clear();
            }
        }
        if (ImGui::MenuItem("Save", "Ctrl+S"))
        {
            std::ofstream file(filePath == "Untitled" ? "example.txt" :
filePath);
            if (file)
            {
                file << textBuffer;
            }
        }
    }
}
```



```

    }
}
if (ImGui::MenuItem("Save As..."))
{
    // TODO: Gọi file dialog và lưu tên vào filePath
    filePath = "example_saved.txt";
    std::ofstream file(filePath);
    if (file)
    {
        file << textBuffer;
    }
}
ImGui::Separator();
if (ImGui::MenuItem("Exit"))
{
    // TODO: Đặt flag để đóng ứng dụng
}
ImGui::EndMenu();
}

if (ImGui::BeginMenu("Edit"))
{
    if (ImGui::MenuItem("Undo", "Ctrl+Z", false, !undoStack.empty()))
    {
        redoStack.push_back(textBuffer);
        textBuffer = undoStack.back();
        undoStack.pop_back();
    }
    if (ImGui::MenuItem("Redo", "Ctrl+Y", false, !redoStack.empty()))
    {
        undoStack.push_back(textBuffer);
        textBuffer = redoStack.back();
        redoStack.pop_back();
    }
    ImGui::Separator();
    if (ImGui::MenuItem("Cut", "Ctrl+X"))
    {
        ImGui::SetClipboardText(textBuffer.c_str());
        undoStack.push_back(textBuffer);
        textBuffer.clear();
    }
}

```

```

    }
    if (ImGui::MenuItem("Copy", "Ctrl+C"))
    {
        ImGui::SetClipboardText(textBuffer.c_str());
    }
    if (ImGui::MenuItem("Paste", "Ctrl+V"))
    {
        undoStack.push_back(textBuffer);
        textBuffer += ImGui::GetClipboardText();
    }
    if (ImGui::MenuItem("Select All", "Ctrl+A"))
    {
        // ImGui::InputTextMultiline không hỗ trợ select all natively
        // Bạn có thể dùng một thư viện editor nếu cần hỗ trợ đầy đủ
    }
    ImGui::EndMenu();
}

ImGui::EndMenuBar();
}

// Shortcut hỗ trợ (không cần ImGui::IsItemFocused)
if (ImGui::IsWindowFocused())
{
    ImGuiIO& io = ImGui::GetIO();
    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_S)) {
        std::ofstream file(filePath == "Untitled" ? "example.txt" : filePath);
        if (file) file << textBuffer;
    }
    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_N)) {
        undoStack.clear(); redoStack.clear();
        textBuffer.clear();
        filePath = "Untitled";
    }
    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_Z) && !undoStack.empty()) {
        redoStack.push_back(textBuffer);
        textBuffer = undoStack.back();
        undoStack.pop_back();
    }
    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_Y) && !redoStack.empty()) {

```

```

        undoStack.push_back(textBuffer);
        textBuffer = redoStack.back();
        redoStack.pop_back();
    }
}

ImVec2 size = ImGui::GetContentRegionAvail();

if (ImGui::InputTextMultiline("##TextEditor", &textBuffer, size, flags))
{
    undoStack.push_back(textBuffer);
    redoStack.clear();
}

ImGui::SetWindowTitle((filePath + " - Notepad").c_str());

```



Ghi chú thêm:

- Nếu bạn **muốn hỗ trợ mở/lưu file chọn bằng dialog**, hãy dùng:
 - ``GetOpenFileNameW()`` (Windows API)
 - ``tinyfiledialogs`` (cross-platform)
- Undo/Redo đơn giản: push vào ``undoStack`` mỗi khi thay đổi; có thể cải tiến bằng so sánh delta hoặc snapshot theo thời gian.

Bạn muốn mình làm sẵn bản `` .cpp `` đầy đủ hoặc cả project Visual Studio với DirectX 11 tích hợp sẵn Dear ImGui? Mình có thể chuẩn bị luôn nếu cần.



bạn hoàn thiện tất cả các phần TODO đi





📌 Đã cập nhật bộ nhớ đã lưu

Dưới đây là phiên bản **hoàn chỉnh** của phần giữa ``ImGui::Begin()`` và ``ImGui::End()`` cho một **Notepad bằng Dear ImGui**, với tất cả phần ``TODO`` đã được xử lý:

- **File dialog**: sử dụng ``GetOpenFileNameW`` và ``GetSaveFileNameW`` từ Windows API
- **Thoát ứng dụng**: dùng một ``bool& shouldExit`` giả định là biến bên ngoài
- **Tự động cập nhật tiêu đề cửa sổ**
- **Shortcut đầy đủ**: Ctrl+S, Ctrl+O, Ctrl+N, Ctrl+Z, Ctrl+Y...

✅ **Đoạn code đầy đủ trong ``ImGui::Begin()`` / ``ImGui::End()``:**

```
#include <windows.h>
#include <commdlg.h>
#include <fstream>
#include <string>
#include <vector>

// Dữ liệu toàn cục cho phần GUI
static std::string textBuffer;
static std::string filePath = "Untitled";
static ImGuiInputTextFlags flags = ImGuiInputTextFlags_AllowTabInput;

static std::vector<std::string> undoStack;
static std::vector<std::string> redoStack;
static bool shouldExit = false;

// Hàm chọn file Open
std::string OpenFileDialog() {
    char filename[MAX_PATH] = "";
    OPENFILENAMEA ofn{};
    ofn.lStructSize = sizeof(ofn);
```

```

    ofn.hwndOwner = nullptr;
    ofn.lpstrFilter = "Text Files\\0*.txt\\0All Files\\0*.*\\0";
    ofn.lpstrFile = filename;
    ofn.nMaxFile = MAX_PATH;
    ofn.Flags = OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;
    ofn.lpstrTitle = "Open Text File";

    if (GetOpenFileNameA(&ofn)) {
        return filename;
    }
    return "";
}

// Hàm chọn file Save
std::string SaveFileDialog() {
    char filename[MAX_PATH] = "";
    OPENFILENAMEA ofn{};
    ofn.lStructSize = sizeof(ofn);
    ofn.hwndOwner = nullptr;
    ofn.lpstrFilter = "Text Files\\0*.txt\\0All Files\\0*.*\\0";
    ofn.lpstrFile = filename;
    ofn.nMaxFile = MAX_PATH;
    ofn.Flags = OFN_OVERWRITEPROMPT;
    ofn.lpstrTitle = "Save Text File";

    if (GetSaveFileNameA(&ofn)) {
        return filename;
    }
    return "";
}

// Giao diện chính
if (ImGui::BeginMenuBar())
{
    if (ImGui::BeginMenu("File"))
    {
        if (ImGui::MenuItem("New", "Ctrl+N")) {
            undoStack.clear(); redoStack.clear();
            textBuffer.clear();
            filePath = "Untitled";
        }
    }
}

```

```

}

if (ImGui::MenuItem("Open...", "Ctrl+O")) {
    std::string path = OpenFileDialog();
    if (!path.empty()) {
        std::ifstream file(path);
        if (file) {
            textBuffer.assign((std::istreambuf_iterator<char>(file)),
                               std::istreambuf_iterator<char>());

            filePath = path;
            undoStack.clear(); redoStack.clear();
        }
    }
}

if (ImGui::MenuItem("Save", "Ctrl+S")) {
    std::string path = (filePath == "Untitled") ? SaveFileDialog() :
filePath;
    if (!path.empty()) {
        std::ofstream file(path);
        if (file) {
            file << textBuffer;
            filePath = path;
        }
    }
}

if (ImGui::MenuItem("Save As...")) {
    std::string path = SaveFileDialog();
    if (!path.empty()) {
        std::ofstream file(path);
        if (file) {
            file << textBuffer;
            filePath = path;
        }
    }
}

ImGui::Separator();
if (ImGui::MenuItem("Exit")) {

```

```

        shouldExit = true;
    }
    ImGui::EndMenu();
}

if (ImGui::BeginMenu("Edit"))
{
    if (ImGui::MenuItem("Undo", "Ctrl+Z", false, !undoStack.empty())) {
        redoStack.push_back(textBuffer);
        textBuffer = undoStack.back();
        undoStack.pop_back();
    }
    if (ImGui::MenuItem("Redo", "Ctrl+Y", false, !redoStack.empty())) {
        undoStack.push_back(textBuffer);
        textBuffer = redoStack.back();
        redoStack.pop_back();
    }

    ImGui::Separator();

    if (ImGui::MenuItem("Cut", "Ctrl+X")) {
        ImGui::SetClipboardText(textBuffer.c_str());
        undoStack.push_back(textBuffer);
        textBuffer.clear();
    }

    if (ImGui::MenuItem("Copy", "Ctrl+C")) {
        ImGui::SetClipboardText(textBuffer.c_str());
    }

    if (ImGui::MenuItem("Paste", "Ctrl+V")) {
        undoStack.push_back(textBuffer);
        textBuffer += ImGui::GetClipboardText();
    }

    if (ImGui::MenuItem("Select All", "Ctrl+A")) {
        // ImGui không có native "select all", cần custom editor để hỗ trợ
    }

    ImGui::EndMenu();
}

```

```

}

ImGui::EndMenuBar();
}

// Xử lý phím tắt
if (ImGui::IsWindowFocused()) {
    ImGuiIO& io = ImGui::GetIO();
    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_S)) {
        std::string path = (filePath == "Untitled") ? SaveFileDialog() :
filePath;
        if (!path.empty()) {
            std::ofstream file(path);
            if (file) {
                file << textBuffer;
                filePath = path;
            }
        }
    }

    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_O)) {
        std::string path = OpenFileDialog();
        if (!path.empty()) {
            std::ifstream file(path);
            if (file) {
                textBuffer.assign((std::istreambuf_iterator<char>(file)),
                                std::istreambuf_iterator<char>());
                filePath = path;
                undoStack.clear(); redoStack.clear();
            }
        }
    }

    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_N)) {
        undoStack.clear(); redoStack.clear();
        textBuffer.clear();
        filePath = "Untitled";
    }

    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_Z) && !undoStack.empty()) {

```

```

        redoStack.push_back(textBuffer);
        textBuffer = undoStack.back();
        undoStack.pop_back();
    }

    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_Y) && !redoStack.empty()) {
        undoStack.push_back(textBuffer);
        textBuffer = redoStack.back();
        redoStack.pop_back();
    }
}

// Editor Text
ImVec2 size = ImGui::GetContentRegionAvail();
if (ImGui::InputTextMultiline("##TextEditor", &textBuffer, size, flags)) {
    undoStack.push_back(textBuffer);
    redoStack.clear();
}

// Tự động cập nhật tiêu đề cửa sổ
std::string windowTitle = filePath + " - Notepad";
SetWindowTextA(FindWindowA(nullptr, "Dear ImGui Demo"), windowTitle.c_str());
// Thay bằng đúng tên window của bạn

ImGui::End();

```

Yêu cầu:

- `#include <windows.h>` và `#include <commdlg.h>` cho file dialog
- `SetWindowTextA` chỉ hoạt động nếu bạn biết tên chính xác cửa sổ Win32. Bạn nên dùng handle thực tế thay vì `"Dear ImGui Demo"`.

Nếu bạn muốn, mình có thể:

- Đóng gói toàn bộ project Visual Studio 2022 sẵn chạy
- Hoặc viết bản `WinMain()` đầy đủ để chạy độc lập với DirectX 11 + ImGui

Bạn muốn tiếp tục theo hướng nào?



bạn thử nghĩ xem có thêm chức năng nào hữu dụng ko



Rất hay! Dưới đây là một số **chức năng hữu dụng và thực tế** mà bạn có thể thêm vào để cải thiện trải nghiệm người dùng và tăng tính chuyên nghiệp cho editor giống Notepad:



GỢI Ý CÁC TÍNH NĂNG HỮU DỤNG



1. Hiển thị số dòng

- Giúp định vị vị trí trong văn bản.
- Có thể hiển thị như cột bên trái: `1\n 2\n 3\n ...`
- Gợi ý: Tách `ImGui::Columns()` thành 2 cột (line numbers + nội dung)



2. Tìm kiếm (Find) và Thay thế (Replace)



- Hộp thoại tìm kiếm chuỗi, bôi đậm kết quả
 - Hỗ trợ "Find Next", "Find Previous", "Replace", "Replace All"
-

3. Ghi nhớ danh sách file gần mở (Recent Files)

- Menu "File > Recent Files" giúp mở nhanh lại những file đã mở gần đây
 - Có thể lưu vào `.ini` hoặc json đơn giản khi thoát
-

4. Tùy chọn theme: Light / Dark

- Cho phép chọn màu nền (giống VS Code)
 - Dùng `ImGui::StyleColorsDark()` / `StyleColorsClassic()`...
-

5. Tình trạng (Status Bar)

- Hiển thị dòng/cột hiện tại, trạng thái file đã chỉnh sửa, tên file
 - Ví dụ: `Ln 10, Col 4 | Modified | example.txt`
-

6. Tự động lưu định kỳ

- Ví dụ: lưu tạm mỗi 60 giây
- Bảo vệ tránh mất dữ liệu (đặc biệt khi app crash)

7. Hỗ trợ mã hóa UTF-8 / UTF-16

- Đảm bảo mở được file có tiếng Việt hoặc Unicode
 - Hiện tại ``std::ifstream`` có thể không đọc đúng UTF-8/UTF-16
-

8. Highlight cú pháp cơ bản (Plain text, INI, Markdown)

- Nếu bạn muốn nâng lên mức "code editor", có thể thêm ``ImGuiColorTextEdit``
 - Hoặc tự làm highlight đơn giản với ``ImDrawList`` vẽ màu theo từ khóa
-

9. Reload file khi có thay đổi từ ngoài

- Dùng ``GetFileTime()`` hoặc ``stat()`` để phát hiện file bị sửa từ chương trình khác
 - Hiển thị thông báo "File has changed externally. Reload?"
-

10. Tùy chọn cài đặt: Font, tab size, word wrap...

- Thêm menu ``Preferences``, có thể:
 - Đổi font chữ (Courier, Consolas...)
 - Chọn wrap word hay không
 - Tab = 2, 4 hoặc 8 spaces
 - Zoom in/out bằng Ctrl + Mouse wheel

Nếu muốn mình có thể:

- Thêm từng tính năng bạn chọn
 - Làm từng phần từng bước để bạn tiện tích hợp
-

Bạn muốn bắt đầu với tính năng nào trước? (Mình gợi ý: **Find/Replace** hoặc **Line Number** vì dễ và rất hữu dụng).



Làm lần lượt từng tính năng đi



 Đã cập nhật bộ nhớ đã lưu

Tuyệt vời! Chúng ta sẽ bắt đầu với tính năng **hiển thị số dòng (Line Number)** trước — đây là một phần cơ bản nhưng rất hữu ích.

TÍNH NĂNG 1: Hiển thị số dòng (Line Numbers)

Mục tiêu:

- Tách cửa sổ thành 2 cột:
 - Bên trái hiển thị số dòng

- Bên phải là `InputTextMultiline` để nhập nội dung
- Tự động cập nhật số dòng theo nội dung

 **Cập nhật đoạn giữa `ImGui::Begin()` và `ImGui::End()` như sau:**

```
ImVec2 editorSize = ImGui::GetContentRegionAvail();
float lineNumberWidth = 40.0f;

// Tách thành 2 cột: line number + editor
ImGui::BeginChild("TextEditor", editorSize, true);
ImGui::Columns(2, "editor_columns");
ImGui::SetColumnWidth(0, lineNumberWidth);

// CỘT 1: Line Numbers
{
    ImGui::BeginChild("LineNumbers", ImVec2(lineNumberWidth, editorSize.y),
false,
                                ImGuiWindowFlags_NoScrollbar |
ImGuiWindowFlags_NoScrollWithMouse);

    int lineCount = 1;
    for (size_t i = 0; i < textBuffer.length(); ++i) {
        if (textBuffer[i] == '\n')
            lineCount++;
    }

    for (int i = 1; i <= lineCount; ++i) {
        ImGui::Text("%d", i);
    }

    ImGui::EndChild();
}
ImGui::NextColumn();
```

```
// CỘT 2: Input Text
{
    ImGui::BeginChild("TextArea", ImVec2(0, editorSize.y), false,
ImGuiWindowFlags_HorizontalScrollbar);

    if (ImGui::InputTextMultiline("##TextBuffer", &textBuffer, ImVec2(-1, -1),
flags)) {
        undoStack.push_back(textBuffer);
        redoStack.clear();
    }

    ImGui::EndChild();
}

ImGui::Columns(1);
ImGui::EndChild();
```

Kết quả:

- Cột trái hiển thị số dòng tương ứng.
- Cột phải là ô nhập nội dung như Notepad.
- Có thể scroll nội dung bình thường, giữ đúng vị trí số dòng.

Bạn có muốn mình thêm luôn khả năng **tìm kiếm (Find/Replace)** tiếp theo không?
Hay bạn muốn chỉnh sửa gì ở phần line number trước?



find replace



Tuyệt, ta sẽ thêm tính năng **Tìm kiếm và Thay thế (Find/Replace)** – một chức năng quan trọng trong bất kỳ text editor nào.

✅ TÍNH NĂNG 2: Find / Replace

🎯 Mục tiêu:

- Một popup khi nhấn `Ctrl+F` hoặc chọn menu `Edit > Find...`
- Nhập chuỗi cần tìm
- Highlight kết quả tìm được (basic)
- Có thể `Replace` hoặc `Replace All`

📦 1. Khai báo biến toàn cục:

```
static bool showFindReplace = false;  
static char findText[256] = "";  
static char replaceText[256] = "";  
static int lastMatchPos = -1;
```

🧠 2. Bổ sung vào menu Edit:

```
if (ImGui::MenuItem("Find / Replace", "Ctrl+F")) {  
    showFindReplace = true;
```

```
}
```

3. Bổ sung phím tắt Ctrl+F:

```
if (ImGui::IsWindowFocused()) {  
    ImGuiIO& io = ImGui::GetIO();  
    if (io.KeyCtrl && ImGui::IsKeyPressed(ImGuiKey_F)) {  
        showFindReplace = true;  
    }  
}
```

4. Popup Find/Replace:

```
if (showFindReplace) {  
    ImGui::OpenPopup("Find / Replace");  
    showFindReplace = false;  
}  
  
if (ImGui::BeginPopupModal("Find / Replace", nullptr,  
ImGuiWindowFlags_AlwaysAutoResize)) {  
    ImGui::InputText("Find", findText, IM_ARRAYSIZE(findText));  
    ImGui::InputText("Replace", replaceText, IM_ARRAYSIZE(replaceText));  
  
    if (ImGui::Button("Find Next")) {  
        size_t pos = textBuffer.find(findText, lastMatchPos + 1);  
        if (pos != std::string::npos) {  
            lastMatchPos = static_cast<int>(pos);  
            // Scroll to position: TODO (có thể dùng ImGui::SetScrollHereY)  
        } else {  
            lastMatchPos = -1;  
        }  
    }  
}
```

```

ImGui::SameLine();
if (ImGui::Button("Replace")) {
    if (lastMatchPos != -1) {
        textBuffer.replace(lastMatchPos, strlen(findText), replaceText);
        lastMatchPos = -1; // Reset sau replace
    }
}

if (ImGui::Button("Replace All")) {
    size_t count = 0;
    size_t pos = 0;
    while ((pos = textBuffer.find(findText, pos)) != std::string::npos) {
        textBuffer.replace(pos, strlen(findText), replaceText);
        pos += strlen(replaceText);
        count++;
    }
    lastMatchPos = -1;
}

ImGui::SameLine();
if (ImGui::Button("Close")) {
    ImGui::CloseCurrentPopup();
    lastMatchPos = -1;
}

ImGui::EndPopup();
}

```

Ghi chú:

- Nếu bạn muốn highlight phần khớp (bôi màu), cần xử lý vẽ text thủ công bằng `ImDrawList` hoặc dùng thư viện editor như `TextEditor` của Balazs Jako.
- Tính năng “scroll to match” có thể dùng thêm `SetScrollHereY()` hoặc điều chỉnh `scrollY` bằng tay (phức tạp hơn).

Bạn có muốn mình giúp thêm phần **highlight kết quả tìm kiếm** hoặc tính năng **scroll tới vị trí khớp** không?



cả 2 đi



Tuyệt! Giờ chúng ta sẽ mở rộng tính năng **Find/Replace** bằng cách:

1. **Highlight kết quả tìm kiếm** (bôi màu các từ khớp).
2. **Tự động scroll tới dòng chứa kết quả** khi ``Find Next``.

✅ TÍNH NĂNG MỞ RỘNG: Highlight và Scroll

🎯 Ý tưởng thực hiện:

- Duyệt ``textBuffer``, tìm các vị trí khớp với ``findText``.
- Khi render nội dung văn bản, tách thành từng dòng và từng phần, chèn màu nền khác cho đoạn trùng khớp.
- Khi ``Find Next`` → tính toán dòng chứa ``match``, rồi ``ScrollTo`` dòng đó.



Cập nhật phần render nội dung text



Thay vì dùng `InputTextMultiline`, ta sẽ **render văn bản thủ công từng dòng** để highlight được.

“Lưu ý: Ta vẫn giữ `textBuffer` là `std::string`, nhưng render bằng `ImGui::TextUnformatted` và highlight bằng `ImDrawList`.”

1. Thay thế phần CỘT 2 (nội dung văn bản):

```
// CỘT 2: Custom render with highlight
{
    ImGui::BeginChild("TextArea", ImVec2(0, editorSize.y), false,
ImGuiWindowFlags_HorizontalScrollbar);
    ImDrawList* drawList = ImGui::GetWindowDrawList();
    ImVec2 startPos = ImGui::GetCursorScreenPos();
    float lineHeight = ImGui::GetTextLineHeightWithSpacing();
    ImVec2 textPos = startPos;

    int lineNum = 0;
    size_t currentPos = 0;
    size_t bufferLen = textBuffer.length();

    while (currentPos < bufferLen) {
        size_t lineEnd = textBuffer.find('\n', currentPos);
        if (lineEnd == std::string::npos) lineEnd = bufferLen;

        std::string line = textBuffer.substr(currentPos, lineEnd - currentPos);

        size_t highlightStart = line.find(findText);
        if (!findText[0]) highlightStart = std::string::npos;

        // Highlight nếu có
        if (highlightStart != std::string::npos) {
            ImVec2 hlStart = ImVec2(textPos.x, textPos.y);
            ImVec2 textOffset = ImGui::CalcTextSize(line.substr(0,
highlightStart).c_str());
            hlStart.x += textOffset.x;
        }
    }
}
```

```

        ImVec2 hlEnd = hlStart;
        hlEnd.x += ImGui::CalcTextSize(findText).x;
        hlEnd.y += lineHeight * 0.75f;

        drawList->AddRectFilled(hlStart, hlEnd, IM_COL32(255, 255, 0,
128)); // Màu vàng
    }

    ImGui::SetCursorScreenPos(textPos);
    ImGui::TextUnformatted(line.c_str());

    currentPos = lineEnd + 1;
    textPos.y += lineHeight;
    lineNum++;
}

// Cho phép scroll bằng chuột
float totalHeight = lineNum * lineHeight;
ImGui::Dummy(ImVec2(0, totalHeight));

ImGui::EndChild();
}

```

2. Tính dòng cần scroll tới khi `Find Next`

Cập nhật nút `Find Next` trong popup:

```

if (ImGui::Button("Find Next")) {
    size_t pos = textBuffer.find(findText, lastMatchPos + 1);
    if (pos != std::string::npos) {
        lastMatchPos = static_cast<int>(pos);

        // Tính dòng
        int line = 0;
        for (int i = 0; i < lastMatchPos && i < textBuffer.size(); ++i) {

```

```
        if (textBuffer[i] == '\n') line++;
    }

    // Scroll tới dòng đó
    float lineY = line * ImGui::GetTextLineHeightWithSpacing();
    ImGui::SetScrollY(lineY);
} else {
    lastMatchPos = -1;
}
}
```

Kết quả đạt được:

- Kết quả tìm kiếm sẽ được **highlight vàng** trong văn bản.
- Khi nhấn "Find Next", sẽ **tự động cuộn tới dòng chứa kết quả**.
- Hỗ trợ tốt cho văn bản nhiều dòng, dài.

Bạn có muốn mình tích hợp luôn phần “**status bar**” ở cuối (hiện vị trí dòng/cột, file đã sửa hay chưa) không? Hoặc thêm “**replace only selected match**” khi highlight hiện tại?



làm tính năng recent file đi



Rất tốt! Tiếp theo, ta sẽ thêm tính năng **Recent Files** – giúp người dùng mở lại các file gần đây một cách nhanh chóng, giống như Notepad, VSCode, v.v.

✓ TÍNH NĂNG 3: Recent Files (File gần đây)

🎯 Mục tiêu:

- Tự động lưu tên các file đã mở gần đây (full path).
- Hiển thị trong menu `File > Recent Files`.
- Lưu & đọc danh sách từ file `recent_files.txt` khi đóng/mở chương trình.

📦 1. Khai báo biến toàn cục:

```
static std::vector<std::string> recentFiles;  
static const int maxRecentFiles = 10;
```

📁 2. Load từ file khi khởi động:

“Thêm vào phần khởi tạo chương trình của bạn (outside ImGui frame):”

```
void LoadRecentFiles() {  
    std::ifstream in("recent_files.txt");  
    std::string line;  
    while (std::getline(in, line)) {  
        if (!line.empty()) {  
            recentFiles.push_back(line);  
        }  
    }  
}
```

```
in.close();  
}
```

3. Lưu lại khi thoát:

“Gọi khi đóng chương trình:”

```
void SaveRecentFiles() {  
    std::ofstream out("recent_files.txt");  
    for (const auto& path : recentFiles) {  
        out << path << "\n";  
    }  
    out.close();  
}
```

+ 4. Hàm thêm file mới vào danh sách:

```
void AddToRecentFiles(const std::string& filepath) {  
    // Xoá nếu đã tồn tại để đẩy lên đầu  
    recentFiles.erase(std::remove(recentFiles.begin(), recentFiles.end(),  
filepath), recentFiles.end());  
    recentFiles.insert(recentFiles.begin(), filepath);  
    if (recentFiles.size() > maxRecentFiles) {  
        recentFiles.pop_back();  
    }  
}
```



5. Thêm vào menu `File > Recent Files`:

```
if (ImGui::BeginMenu("Recent Files")) {
    if (recentFiles.empty()) {
        ImGui::MenuItem("No recent files", nullptr, false, false);
    } else {
        for (const auto& file : recentFiles) {
            if (ImGui::MenuItem(file.c_str())) {
                // Load file
                std::ifstream in(file);
                if (in) {
                    textBuffer = std::string((std::istreambuf_iterator<char>
(in)), std::istreambuf_iterator<char>());
                    currentFilePath = file;
                    AddToRecentFiles(file); // Đưa lên đầu
                    lastMatchPos = -1; // reset find
                }
            }
        }
    }
    ImGui::EndMenu();
}
```



6. Khi mở file (mới):

"Khi bạn có thao tác mở file thành công:"

```
// Sau khi mở file
AddToRecentFiles(currentFilePath);
```



Kết quả:

- File vừa mở sẽ được thêm vào danh sách recent.
 - Khi khởi động chương trình, recent files sẽ được tự động hiển thị.
 - Dễ dàng load lại file chỉ bằng 1 click.
-

Bạn muốn mình thêm tính năng nào tiếp theo?

Gợi ý tiếp theo:

- **Status bar** (hiện dòng/cột, tên file, trạng thái modified)
- **Tùy chọn theme Light/Dark**
- **Tự động lưu định kỳ (auto-save)**
- **Undo/Redo nâng cao**
- Hoặc: Chọn file bằng `ImGuiFileDialog`` để dễ mở hơn?



theme



Tuyệt! Chúng ta sẽ thêm tính năng **Theme Switcher** để cho phép người dùng chuyển giữa **Light** và **Dark mode**, tương tự như các editor hiện đại như VSCode hoặc Notepad++.

✅ TÍNH NĂNG 4: Chuyển Theme (Light/Dark)

🎯 Mục tiêu:

- Cho phép người dùng chuyển giữa 2 theme: `Dark` và `Light`.
- Tùy chọn nằm trong menu `View > Theme`.

📦 1. Khai báo biến lựa chọn theme:

```
enum class Theme { Dark, Light };  
static Theme currentTheme = Theme::Dark;
```

🌈 2. Hàm đổi theme:

```
void ApplyTheme(Theme theme) {  
    if (theme == Theme::Dark) {  
        ImGui::StyleColorsDark();  
    } else if (theme == Theme::Light) {  
        ImGui::StyleColorsLight();  
    }  
}
```

“Gọi `ApplyTheme(currentTheme)`; khi khởi động để áp dụng theme mặc định.”

3. Thêm vào menu `View > Theme`:

```
if (ImGui::BeginMenu("View")) {
    if (ImGui::BeginMenu("Theme")) {
        bool dark = currentTheme == Theme::Dark;
        bool light = currentTheme == Theme::Light;

        if (ImGui::MenuItem("Dark", nullptr, dark)) {
            currentTheme = Theme::Dark;
            ApplyTheme(currentTheme);
        }
        if (ImGui::MenuItem("Light", nullptr, light)) {
            currentTheme = Theme::Light;
            ApplyTheme(currentTheme);
        }

        ImGui::EndMenu();
    }
    ImGui::EndMenu();
}
```

4. Gợi ý mở rộng:

Bạn có thể lưu lựa chọn theme vào file config (`config.txt`) và đọc lại khi mở chương trình:

```
// Khi lưu:
std::ofstream out("config.txt");
out << (currentTheme == Theme::Dark ? "Dark" : "Light") << std::endl;

// Khi load:
std::ifstream in("config.txt");
std::string theme;
if (in >> theme) {
    currentTheme = (theme == "Light") ? Theme::Light : Theme::Dark;
}
```

```
ApplyTheme(currentTheme);  
}
```

✓ Kết quả:

- Người dùng có thể chọn theme từ menu.
- Tự động áp dụng theme ngay lập tức.
- Có thể mở rộng thêm nhiều theme custom khác (Monokai, Dracula, Solarized...)

Bạn có muốn tiếp tục với:

- **Status Bar** (hiển thị dòng/cột, trạng thái file)
- **Auto-Save định kỳ**
- **Mở file bằng ImGuiFileDialog**
- **Font settings (thay đổi font hoặc size)?**

Bạn chọn nhé!



auto save



Ok, ta sẽ làm tính năng **Auto-Save (Tự động lưu)** cho editor nhé.

✅ TÍNH NĂNG 5: Auto-Save định kỳ

🎯 Mục tiêu:

- Tự động lưu nội dung editor ra file hiện tại sau mỗi khoảng thời gian (ví dụ 30s).
- Có thể tắt/bật tính năng này từ menu `File > Auto-Save``.
- Hiển thị trạng thái auto-save trên status bar.

📦 1. Khai báo biến toàn cục:

```
static bool autoSaveEnabled = true;  
static float autoSaveInterval = 30.0f; // giây  
static float autoSaveTimer = 0.0f;  
static bool fileModified = false; // đánh dấu file có thay đổi chưa lưu
```

2. Đánh dấu khi file bị sửa (bạn đã có `textBuffer``, cập nhật chỗ `InputTextMultiline``):

```
if (ImGui::InputTextMultiline("##TextBuffer", &textBuffer, ImVec2(-1, -1),  
flags)) {  
    undoStack.push_back(textBuffer);  
    redoStack.clear();  
    fileModified = true;  
}
```

3. Trong vòng lặp frame chính (cập nhật timer):

```
autoSaveTimer += ImGui::GetIO().DeltaTime;
if (autoSaveEnabled && fileModified && !currentFilePath.empty() &&
    autoSaveTimer >= autoSaveInterval) {
    // Lưu file
    std::ofstream out(currentFilePath);
    if (out) {
        out << textBuffer;
        out.close();
        fileModified = false;
        autoSaveTimer = 0.0f;
    }
}
```

4. Thêm mục bật/tắt Auto-Save trong menu `File`:

```
if (ImGui::MenuItem("Auto-Save", nullptr, autoSaveEnabled)) {
    autoSaveEnabled = !autoSaveEnabled;
}
```

5. (Tùy chọn) Hiển thị trạng thái Auto-Save trong status bar:

Ví dụ đoạn nhỏ:

```
ImGui::BeginChild("StatusBar", ImVec2(0, 20), false);
ImGui::Text("Auto-Save: %s | %s", autoSaveEnabled ? "On" : "Off", fileModified
? "Modified" : "Saved");
ImGui::EndChild();
```

✅ Kết quả:

- Nội dung file sẽ tự động lưu sau mỗi 30 giây nếu có thay đổi.
- Người dùng có thể bật/tắt auto-save.
- Thông tin trạng thái hiển thị trực quan.

Bạn muốn mình làm phần **status bar** chi tiết luôn, hiện dòng/cột và trạng thái file không?



có



Ok, mình sẽ làm phần **Status Bar chi tiết** cho bạn luôn nhé.

✅ TÍNH NĂNG 6: Status Bar (Dòng/Cột, Trạng thái File, Auto-Save)

🎯 Mục tiêu:

- Hiển thị dòng hiện tại và cột con trỏ.
- Hiển thị tên file (hoặc "Untitled" nếu chưa lưu).
- Hiển thị trạng thái file: đã lưu hay chưa (Modified).
- Hiển thị trạng thái Auto-Save (On/Off).

Ý tưởng:

Vì ta đang custom render text (không dùng `ImGui::InputTextMultiline` nữa) để highlight, nên phải track con trỏ thủ công.

Ta sẽ dùng biến `cursorPos` để lưu vị trí con trỏ (tính theo offset trong `textBuffer`).

1. Cập nhật con trỏ (ví dụ dùng `ImGui::IsWindowFocused()` và chuột bàn phím để đặt con trỏ):

Vì `ImGui::InputTextMultiline` không còn, bạn có thể bổ sung thêm xử lý con trỏ (phức tạp, mình có thể giúp nếu cần), hoặc nếu bạn vẫn dùng `ImGui::InputTextMultiline` với highlight đơn giản thì dễ lấy vị trí dòng/cột.

2. Giả sử bạn có biến `cursorPos` (offset trong `textBuffer`), ta tính dòng và cột:

```
int GetLineFromPos(int pos, const std::string& text) {
    int line = 0;
    for (int i = 0; i < pos && i < text.size(); ++i) {
        if (text[i] == '\n') line++;
    }
}
```

```

    }
    return line;
}

int GetColumnFromPos(int pos, const std::string& text) {
    int col = 0;
    for (int i = pos - 1; i >= 0 && i < text.size(); --i) {
        if (text[i] == '\n') break;
        col++;
    }
    return col;
}

```

3. Status Bar code:

```

ImGui::BeginChild("StatusBar", ImVec2(0, 20), false);

std::string fileName = currentFilePath.empty() ? "Untitled" : currentFilePath;
int line = GetLineFromPos(cursorPos, textBuffer) + 1;
int col = GetColumnFromPos(cursorPos, textBuffer) + 1;

ImGui::Text("%s | Ln %d, Col %d | %s | Auto-Save: %s",
            fileName.c_str(),
            line,
            col,
            fileModified ? "Modified" : "Saved",
            autoSaveEnabled ? "On" : "Off");

ImGui::EndChild();

```

4. Cập nhật `cursorPos`

Nếu bạn vẫn dùng `InputTextMultiline` (không render thủ công), bạn có thể lấy vị trí con trỏ qua:

```
ImGuiInputTextCallbackData* callback_data = ...;
int cursorPos = callback_data->CursorPos;
```

Nếu bạn đã custom render, cần thêm xử lý chuột và bàn phím để cập nhật `cursorPos` — nếu muốn, mình giúp thêm phần đó.

✓ Kết quả:

- Thanh status bar nằm dưới cùng cửa sổ.
- Hiển thị đầy đủ: tên file, dòng/cột hiện tại, trạng thái file, trạng thái Auto-Save.
- Giúp người dùng theo dõi trạng thái tập tin dễ dàng hơn.

Bạn muốn mình giúp làm phần **cursorPos** khi custom render để xử lý con trỏ không? Hay cần làm thêm tính năng nào khác?



status bar Hiển thị dòng/cột hiện tại, trạng thái file đã chỉnh sửa, tên file



Dưới đây là đoạn code mẫu cho **status bar** hiển thị đầy đủ:

- Tên file (hoặc "Untitled" nếu chưa lưu)

- Dòng và cột hiện tại của con trỏ text
 - Trạng thái file: "Modified" hoặc "Saved"
-

Giả định:

- `textBuffer` là nội dung văn bản (`std::string`)
 - `currentFilePath` là đường dẫn file hiện tại (`std::string`)
 - `fileModified` là cờ đánh dấu file có chỉnh sửa (`bool`)
 - `cursorPos` là vị trí con trỏ (offset trong `textBuffer`, kiểu `int`)
-

Hàm tính dòng/cột từ `cursorPos`:

```
int GetLineFromPos(int pos, const std::string& text) {
    int line = 0;
    for (int i = 0; i < pos && i < (int)text.size(); ++i) {
        if (text[i] == '\n')
            line++;
    }
    return line;
}

int GetColumnFromPos(int pos, const std::string& text) {
    int col = 0;
    for (int i = pos - 1; i >= 0 && i < (int)text.size(); --i) {
        if (text[i] == '\n')
            break;
        col++;
    }
}
```

```
    return col;
}
```

Code hiển thị Status Bar với ImGui:

```
ImGui::BeginChild("StatusBar", ImVec2(0, 20), false);

std::string fileName = currentFilePath.empty() ? "Untitled" : currentFilePath;
int line = GetLineFromPos(cursorPos, textBuffer) + 1; // dòng bắt đầu từ 1
int col = GetColumnFromPos(cursorPos, textBuffer) + 1; // cột bắt đầu từ 1

ImGui::Text("%s | Ln %d, Col %d | %s",
            fileName.c_str(),
            line,
            col,
            fileModified ? "Modified" : "Saved");

ImGui::EndChild();
```

Lưu ý quan trọng:

- Bạn cần cập nhật giá trị `cursorPos` chính xác theo vị trí con trỏ trong editor.
- Nếu dùng `ImGui::InputTextMultiline`, có thể lấy `cursorPos` trong callback `ImGuiInputTextCallbackData`.
- Nếu custom render text, bạn cần xử lý chuột/bàn phím để thay đổi `cursorPos`.

Nếu bạn cần, mình có thể hỗ trợ phần cập nhật `cursorPos` hoặc thêm tính năng khác nhé!



cursorPos đi



Ok, để lấy được `cursorPos` (vị trí con trỏ) khi dùng **ImGui::InputTextMultiline**, bạn có thể sử dụng callback `ImGuiInputTextCallbackData`. Đây là cách chuẩn và đơn giản nhất để track con trỏ khi người dùng di chuyển hoặc chỉnh sửa text.

Cách làm:

1. Khai báo biến lưu `cursorPos` toàn cục:

```
static int cursorPos = 0;
```

2. Định nghĩa callback để cập nhật `cursorPos`:

```
int TextEditCallback(ImGuiInputTextCallbackData* data) {  
    cursorPos = data->CursorPos;  
    return 0;  
}
```

3. Sử dụng callback khi gọi `InputTextMultiline`:

```
ImGuiInputTextFlags flags = ImGuiInputTextFlags_CallbackAlways;

if (ImGui::InputTextMultiline("##TextBuffer", &textBuffer, ImVec2(-1, -1),
    flags, TextEditCallback)) {
    fileModified = true;
}
```

4. Giải thích:

- `ImGuiInputTextFlags_CallbackAlways` sẽ gọi callback **mọi lúc** con trỏ hoặc nội dung thay đổi.
- Trong callback, bạn cập nhật `cursorPos = data->CursorPos`.
- Nhờ đó, `cursorPos` luôn chính xác vị trí hiện tại của con trỏ trong `textBuffer`.

5. Áp dụng `cursorPos` để tính dòng/cột cho status bar như đoạn trước mình gửi.

Tóm lại:

Bạn chỉ cần thêm callback này và flag khi gọi `InputTextMultiline` là có con trỏ chuẩn để hiện status bar dòng/cột.

Nếu bạn muốn mình hỗ trợ thêm phần xử lý con trỏ khi bạn custom render text (không dùng InputTextMultiline) thì nói mình nhé!