TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Quang Thông
Nguyễn Anh Vũ
Võ Nhật Phước
Hoàng Trung Nguyên

XÂY DỰNG GAME CỜ CARO

ĐỒ ÁN MÔN HỌC
KĨ THUẬT LẬP TRÌNH

Thành Phố Hồ Chí Minh, Tháng 04 năm 2023

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC
KĨ THUẬT LẬP TRÌNH
ĐỀ TÀI:
XÂY DỰNG GAME CỜ CARO

Nhóm 11

22127401 - Nguyễn Quang Thông

22127298 - Hoàng Trung Nguyên

22127339 - Võ Nhật Phước

22127463 - Nguyễn Anh Vũ

Giáo Viên Hướng Dẫn: Trương Toàn Thịnh

Thành Phố Hồ Chí Minh, Tháng 04 năm 2023

Lời cảm ơn

Mục lục

Danh sách hình

Chương 1.

Tổng quan về trò chơi

1.1. Giới thiệu về trò chơi

1.1.1. Gomoku

Nguyên

1.1.2. Mục tiêu đề ra

- Game có nhiều ngôn ngữ, người dùng có thể thêm được ngôn ngữ mới
- Có thể load được các theme(chủ đề) bên ngoài
- Lưu được các thiết lập của người chơi
- Có thể save, load game đang chơi
- Có thể lưu và phát lại các game đã hoàn thành
- Có nhiều chế độ chơi
- Có thể chơi với máy, máy có nhiều mức độ
- Game có thể phát nhạc nền, hiệu ứng. Có thể bật tắt được

1.1.3. Thông tin chung về trò chơi

Nguyên Link souce code, chạy trên nền tảng nào, ...

1.2. Mô tả về các tính năng của game

1.2.1. Đa ngôn ngữ

1.2.2. Thay đổi Theme(Chủ đề)

1.2.3. Lưu thiết lập của người chơi

1.2.4. Save/Load game đang chơi, replay game đã chơi xong

1.2.5. Chế độ chơi Thường

1.2.6. Chế độ chơi Rush

1.2.7. Đánh với máy

1.2.8. Đánh với người

1.2.9. Các hỗ trợ trong lúc chơi game

1.2.9.1. Gợi ý

1.2.9.2. Nổi bật nước mới đi

1.2.9.3. Cảnh báo nước 4

1.2.9.4. Hoàn tác nước đi

1.2.9.5. Đi nháp

Chương 2.

Chi tiết các chức năng

2.1. Logic

2.1.1. Chơi hiệu ứng, nhạc nền

Các file âm thanh được đặt trong thư mục asset/audio và có thể truy cập bằng các `enum`. Các `enum` được map sang một mảng chứa tên các file âm thanh. Các hàm và class nằm trong `namespace Audio`, file `Audio.h`, `Audio.cpp`

```
enum class Sound : char {
    NoSound = 0,
    OnKey,
    Draw,
    Win,
    Lose,
    MenuBGM,
    MenuMove,
    MenuSelect,
    GameBGM,
    WinSound,
    GamePlace,
    GameStart,
    Pause,
    WarningSound
};
```

```
constexpr std::array SoundName{
    L"",
    L"Key.wav",
    L"Draw.mp3",
    L"Win.mp3",
    L"Lose.mp3",
    L"MenuBGM.mp3",
    L"MenuMove.wav",
    L"MenuSelect.wav",
    L"GameBGM.mp3",
    L"WinSound.mp3",
    L"GamePlaceMove.mp3",
    L"GameStart.wav",
    L"Pause.wav",
    L"Warning.mp3"
};
```

2.1.1.1. Hàm PlayAndForget

Hàm này sử dụng hàm PlaySound [1] để chơi nhạc. Được dùng để chơi những âm thanh ngắn, dung lượng nhỏ dưới 100kb. Khi gọi hàm sẽ tự load file vào memory, chơi và đóng file. Do phải load cả file vào bộ nhớ nên khi chơi có độ delay cao và chỉ có thể mở được file `wav`. Được ứng dụng để chơi các âm thanh liên quan tới giao diện, các âm thanh không quan tâm tới độ trễ.

Interface:

```
    bool PlayAndForget(Sound sound, bool wait)
```

Parameters:

- `Sound`: âm thanh cần chơi
- `wait`:
  - `true` => phát âm thanh một cách đồng bộ (synchronous)
  - `false` => phát âm thanh một cách bất đồng bộ (asynchronous)

Usage:

```
    Audio::PlayAndForget(Audio::Sound::MenuSelect);
```

2.1.1.2. Class AudioPlayer

Class này sử dụng **`Media Control Interface`** (`MCI`) [2] để chơi nhạc nên giải quyết được các vấn đề của hàm `PlayAndForget`. Chơi được các file âm thanh định dạng `mp3` và `wav`, chơi được các file lớn, ít delay do không cần load hết file vào bộ nhớ. Nhược điểm là cần phải quan tâm đến tuổi thọ của class nên không tiện dụng như `PlayAndForget`. Được dùng để chơi nhạc nền, những đoạn nhạc cần độ trễ thấp hoặc file `mp3`.

Interface:

```
class AudioPlayer {
    // Ngăn copy hay move class
    AudioPlayer(AudioPlayer&&) = delete;
    AudioPlayer(const AudioPlayer&) = delete;
    AudioPlayer& operator=(AudioPlayer&&) = delete;
    AudioPlayer& operator=(const AudioPlayer&) = delete;

    AudioPlayer();
    AudioPlayer(Sound song); // Khởi tạo và mở file

    // Mở file. Có thể dùng để đổi file cần chơi
    int Open(Sound song);
    Sound getCurrentSong() const;

    int Play(bool fromStart = 1, bool repeat = 0) const; // Chơi
    int Pause() const;      // Tạm dừng
    int Resume() const;     // Tiếp tục
    int Stop() const;       // Dừng chơi và trả con trỏ về đầu
    int Close();            // Đóng file đang mở

    ~AudioPlayer();         // Đóng file đang mở
}
```

Parameters:

- `song` : âm thanh cần chơi
- `fromStart` :
  - `true` => chơi từ đầu
  - `false` => chơi tiếp tại vị trí con trỏ
- `repeat` :
  - `true` => lặp lại khi kết thúc

Return:

- Các phương thức sẽ trả về `MCI code` của lệnh MCI tương ứng

Usage:

```
{
    Audio::AudioPlayer player(Audio::Sound::Draw);
    player.play(true, true);
    player.pause();
    player.resume();
    player.close();
}
```

2.1.1.3. Static class BackgroundAudioService

Class này được dùng để chơi nhạc nền, sử dụng class AudioPlayer để chơi nhạc vì có tuổi thọ dài.

Interface:

```
class BackgroundAudioService {
    BackgroundAudioService() = delete;
    static Audio::Sound GetCurrentSong();

    static int ChangeSong(Audio::Sound song) // Đổi nhạc

    static int Play(bool fromStart = 0, bool repeat = 1); // Chơi
    static int Pause();                         // Tạm dừng
    static int Resume();                        // Tiếp tục
    // Dừng chơi, trả con trỏ về đầu
    static int Stop();
};
```

Parameters:

- `song` : âm thanh cần chơi
- `fromStart` :
    - `true` => chơi từ đầu
    - `false` => chơi tiếp tại vị trí con trỏ
- `repeat` :
    - `true` => lặp lại khi kết thúc

Usage:

```
{
    BackgroundAudioService::ChangeSong(Audio::Sound::MenuBGM);
    BackgroundAudioService::Play(true, true);
}
```

2.1.2. Điều hướng trong ứng dụng

Thông

2.1.3. Đồng hồ

Thông

2.1.4. Đọc, ghi, tìm file

Các hàm nằm trong `namespace FileHandle`, file `FileHandle.h`, `FileHandle.cpp`

2.1.4.1. Các hàm hỗ trợ mở file

Hỗ trợ mở các file văn bản `utf-8`

Interface:

```
    typedef std::filesystem::path fsPath;
    std::wofstream OpenOutFile(const fsPath& filePath);
    std::wifstream OpenInFile (const fsPath& filePath);
```

Parameters:

- `filePath`: đường dẫn đến file cần mở

Usage:

```
#include <string>
{
    std::wstring str = L"Tiếng Việt";
    auto outFile = FileHandle::OpenOutFile("test.txt");
    outFile <<  str;
    outFile.close();
    auto inFile  = FileHandle::OpenInFile ("test.txt");
    infile  >> str;
}
```

2.1.4.2. Hàm Ensure

Dùng để đảm bảo đường dẫn đến file muốn mở có tồn tại, nếu không tồn tại, nếu không tồn tại thì tạo đường dẫn đó.

Interface:

```
    void Ensure(const std::filesystem::path& Dir);
```

Parameters:

- `Dir`: đường dẫn muốn kiểm tra/tạo

Return:

- Các fstream tương ứng với thao tác In/Out

Usage:

```
{
    // Đảm bảo đường dẫn tương đối "asset/language" tồn tại
    FileHandle::Ensure("asset/language");
}
```

2.1.4.3. Hàm Delete

Dùng để xóa file

Interface:

```
    bool Delete(const std::filesystem::path& target)
```

Paramterers:

- target: đường dẫn tới file cần xóa

Return:

- Trả về `true` nếu xóa thành công, `false` khi lỗi

Usage:

```
{
    // Xóa file tmp.cpp
    bool res = FileHandle::Delete("tmp.cpp");
    if (res) {
        std::cout << "Success";
    } else {
        std::cout << "Failed";
    }
}
```

2.1.4.4. Hàm GetAllTextFileInDir

Tìm các file văn bản thuần trong thư mục

Interface:

```
struct FileDetail {
    std::filesystem::path               filePath;
    std::filesystem::file_time_type     lastModified;
};

std::vector<FileHandle::FileDetail>
FileHandle::GetAllTextFileInDir(
    const std::filesystem::path& Dir
);
```

Parameters:

- `Dir`: đường dẫn đến thư mục muốn tìm

Return:

- Trả về một `vector` chứa các thông tin của các file đã tìm được

Usage:

```
{
    // Tìm các file văn bản trong đường dẫn
    // tương đối "asset/language"
    auto files = FileHandle::GetAllTextFileInDir(
        "asset/language"
    );
    for (auto& file:files) {
        std::cout << file.filePath.filename() << '\n';
    }
}
```

2.1.5. Ngôn ngữ

Các văn bản trong trò chơi sẽ được load từ một file riêng, điều này kiến cho phần ngôn ngữ trong game dễ tùy biến và thêm các ngôn ngữ mới.

File ngôn ngữ là một file văn bản thuần chứa các nhãn và phần văn bản ngăn cách bởi dấu "=", các nhãn có nằm bên trong cặp ngoặc `[]` là `meta` được dùng để chứa thông tin về file ngôn ngữ

Ví dụ file ngôn ngữ:

```
[LANGUAGE]                  =    English
[LANG_SELECT]               =    Language
ABOUT_DESC                  =    LIST OF GROUP MEMBERS AND SOURCE
CODE LINK
ABOUT_SHORTCUT              =    A
ABOUT_TITLE                 =    About us
ABOUT_US_TITLE              =    About us
```

Các phần văn bản sẽ được truy xuất thông qua nhãn tương ứng. Các phần liên quan tới ngôn ngữ nằm trong file `Language.h` và `Language.cpp`

2.1.5.1. Static class Language

Hàm chứa các phương thức và các văn bản ngôn ngữ

Interface:

```
typedef std::unordered_map<std::wstring, std::wstring> Dict;
typedef std::filesystem::path fsPath;

struct LanguageOption {
  Dict meta;
  fsPath path;
};

class Language {
  static Dict languageDict;

public:
  Language() = delete;

  // Chỉ đọc các phần thông tin về ngôn ngữ
  static Dict ExtractMetaFromFile(const fsPath& filePath);

  // Load file ngôn ngữ
  static void LoadLanguageFromFile(const fsPath& filePath);

  // Tìm các file ngôn ngữ
  static std::vector<LanguageOption>
  DiscoverLanguageFile(const fsPath& dirPath);

  // Truy xuất văn bản bằng nhãn
  static const std::wstring&
  GetString(const std::wstring& Label);

  static const std::wstring&
  GetMeta(const std::wstring& Label);
};
```

Parameters:

- `filePath`: đường dẫn tới file cần mở
- `dirPath`: đường dẫn tới thư mục cần tìm

- `Label` : nhãn của văn bản cần lấy

Usage:

```
{
    Language::LoadLanguageFromFile("asset/language/en.txt");
    std::cout << Language::GetMeta(L"[LANGUAGE]");
    std::cout << Language::GetMeta(L"ABOUT_TITLE");
}
```

2.1.6. Cài đặt

Thông

2.1.7. Chủ đề

Thông

2.1.8. Hàm trung gian hỗ trợ vẽ giao diện

Thông

2.1.9. Nhận biết thắng thua

Việc nhận biết kết quả thắng, thua, và hòa của một ván đấu được thực hiện trong `namespace Logic` của chương trình. Các kết quả này là điều kiện để chương trình quyết định kết thúc ván đấu. Ngoài ra, việc biết được kết quả thắng, thua, và hòa sẽ giúp AI của trò chơi đưa ra đánh giá về trạng thái bàn cờ một cách đúng đắn.

2.1.9.1. Hàm GetGameState

Hàm `GetGameState` có vai trò đánh giá hiện trạng của ván đấu sau nước đi mới nhất. Cụ thể hơn, hàm xem xét nước đi mới nhất có dẫn đến một **kết quả thắng** hay **kết quả hòa**. Một nước đi sẽ dẫn đến kết quả thắng nếu nước đi đó tạo nên một chuỗi 5 nước đi liên tiếp đồng chất, và một nước đi sẽ dẫn đến kết quả hòa nếu nước đi đó không phải là nước đi thắng, đồng thời là nước đi hợp lệ cuối cùng của bàn đấu.

Interface:

```
short GetGameState(
    const GameAction::Board& board,
    const short& moveCount,
    const GameAction::Point& move,
    const short& playerValue,
    GameAction::Point& winPoint = temp,
    bool getWinPoint
);
```

Parameters:

- `board`: Bàn đấu hiện tại.
- `moveCount`: Số nước đi đã thực hiện.
- `move`: Nước đi mới nhất.
- `playerValue`: Người chơi thực hiện nước đi.
- `winPoint`: Đầu mút của chuỗi thắng (nếu có).
- `getWinPoint`:
  - `true` => lấy đầu mút của chuỗi thắng (nếu có).
  - `false` => không lấy đầu mút của chuỗi thắng.

Returns:

- `Logic::WIN_VALUE`: Giá trị tượng trưng kết quả thắng.
- `Logic::DRAW_VALUE`: Giá trị tượng trưng kết quả hòa.
- `Logic::NULL_VALUE`: Giá trị tượng trưng kết quả vô định.

Usage:

```
short gameState = Logic::GetGameState(gameBoard, moveCount,
latestMove, currentPlayer, winPoint, true);
```

2.1.10. Các tương tác với bàn cờ

Trạng thái bàn cờ có thể được thay đổi qua hai hình thức: **thực hiện nước đi** và **xóa nước đi**. Hàm `MakeMove` và `UndoMove` đảm nhiệm việc thực hiện hai chức năng ấy. Hai hàm này tuy đơn giản nhưng nắm vai trò quan trọng xuyên suốt quá trình chơi, vì các thao tác người chơi chỉ thực sự được ghi lại trên

bàn cờ khi hai hàm này được gọi đến. Việc xóa nước đi là để phục vụ cho chức năng `Hoàn tác` của trò chơi, và là một phần không thể thiếu đối với thuật toán `Minimax` được sử dụng cho AI của trò chơi.

Interface:

```
 // Thực hiện nước đi lên bàn cờ
 void MakeMove(
    Board& board,
    short& moveCount,
    const Point& move,
    const short& playerValue
);

// Xóa nước đi khỏi bàn cờ
void UndoMove(
    Board& board,
    short& moveCount,
    const Point& move
);
```

Parameters:

- `board`: Bàn đấu hiện tại.
- `moveCount`: Số nước đi đã thực hiện.
- `move`: Nước đi được thực hiện.
- `playerValue`: Người chơi thực hiện nước đi.

Usage:

```
GameAction::MakeMove(board, moveCount, latestMove, currentPlayer);
GameAction::UndoMove(board, moveCount, latestMove, currentPlayer);
```

2.1.11. AI

Việc thiết kết chương trình cho chế độ `Đánh với máy` là một trong những thách thức lớn nhất của đồ án. Khác với những tính năng khác của chương trình, tính năng này đòi hỏi những mảng kiến thức chuyên biệt về các thuật toán, kĩ thuật

lập trình cụ thể. Ngoài ra, việc đánh giá độ đúng/sai của chương trình, hay nói cách khác là nước đi máy tính tìm được là tốt hay xấu, sẽ phần lớn phụ thuộc vào cảm tính và sự hiểu biết của người viết. Chính vì vậy, chương trình có thể đánh hay đối với người này, nhưng đánh không tốt đối với người khác. Phần tiếp theo sẽ trình bày những kĩ thuật mà nhóm đã sử dụng cho chương tình AI.

2.1.11.1. Thuật toán Minimax

2.1.11.2. Đánh giá bàn cờ

2.1.11.3. Cải thiện tốc độ

2.1.11.3.1. Alpha-Beta pruning

2.1.11.3.2. Sắp xếp nước đi tìm kiếm

2.1.11.3.3. Transposition table (Bảng hoán vị)

2.1.11.3.4. So sánh tốc độ

2.1.11.4. Phân độ khó

2.1.11.5. Chức năng "Gợi ý"

2.1.11.6. Những mặt cần cải thiện

2.2. Giao diện

2.2.1. Cài đặt

Thông

2.2.2. Các màn hình lưu, tải game và replay

Thông

2.2.3. Màn hình trò chơi chính

Vũ

2.2.4. Các màn hình khác

Chương 3.

Đánh giá thành viên

Chương 4.

Kết luận

4.1. Kết quả đạt được

4.1.1. Ưu điểm của trò chơi

- Có thể thêm nhiều ngôn ngữ và theme vào trò chơi
- Có nhạc hay, hiệu ứng sống động
- Có chế độ tính thời gian
- AI chạy tương đối tốt, đánh nhanh
- Lối chơi đa dạng
- Có nhiều nhân vật ngộ nghĩnh
- Có nhiều tính năng hỗ trợ khi chơi game
- Có thể xem lại trận đấu đã chơi
- Màn hình save/load có khả năng tương tác tốt
- Hướng dẫn dễ hiểu, có gợi ý ở mỗi màn hình

4.1.2. Khuyết điểm của trò chơi

-

4.2. Các khó khăn gặp phải

- Một vài thành viên không có kinh nghiệm sử dụng git và GitHub
- Khó khăn khi lập trình đa luồng do không có kinh nghiệm

- Màn hình terminal vẽ các kí tự chậm

4.3. Những gì đã học được

- Cách làm việc nhóm với git và GitHub
- Cách sử dụng các tính năng mới của C++ 20
- Cách sử dụng các tính năng liên quan tới đo hiệu năng, format code và debug trong Visual Studio
- Cách làm việc nhóm hiệu quả
- Cách lên kế hoạch, phân chia công việc
- Cách lập trình hướng đối tượng cơ bản

4.4. Các kinh nghiệm rút ra

- Khi code, nên tách nhỏ các hàm ra, không nên viết hàm dài
- Code xong một hàm phải kiểm tra kĩ, tránh phát sinh lỗi về sau
- Không nên viết code mà không thiết kế trước
- Nên viết code theo một quy chuẩn nhất định và đồng bộ trong 1 dự án
- Nên viết code có tính tái sử dụng cao
- Cần có kế hoạch và phân chia công việc rõ ràng
- Code xong phải có người review lại

4.5. Lí do hoàn thành mục tiêu

Nguyên

4.6. Hướng phát triển ứng dụng

- Có thể chơi 2 người qua mạng lan
- Thêm nhiều ngôn ngữ mới
- Thêm nhiều chủ đề hơn
- Hiện lợi thế của 2 bên
- Đưa game lên nhiều nền tảng khác

Tài liệu tham khảo

[1] “Playsound function.” https://learn.microsoft.com/en-us/previous-versions/dd743680(v=vs.85)

[2] “Mci.” https://learn.microsoft.com/vi-vn/windows/win32/multimedia/mci