



UIT

**TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN**

KỸ NĂNG NGHỀ NGHIỆP

Đề tài:

SS004 - Đồ Án Cuối Kỳ - Tetris Game

Lớp: CN1.K2025.1.CNTT

Môn học: Kỹ năng nghề nghiệp

Giảng viên hướng dẫn: ThS. Nguyễn Văn Toàn

MỤC LỤC

I. Hợp đồng nhóm	1
1. Thời gian thành lập	1
2. Thời hạn hợp đồng.....	1
3. Tên nhóm	1
4. Thành viên	1
5. Mục đích thành lập	2
6. Vai trò của các thành viên trong nhóm	3
7. Mô tả dự án Tetris	3
8. Hiệp định nhóm	4
9. Không gian sinh hoạt và trao đổi của nhóm	6
10. Tiêu chí đánh giá các thành viên.....	8
11. Cam kết.....	9
II. Công cụ hỗ trợ và quản lý dự án.....	9
1. Quản lý công việc và tiến độ	9
2. Quản lý mã nguồn	10
3. Giao tiếp và trao đổi nhóm	10
4. Soạn thảo báo cáo	10
III. Phần giới thiệu và hướng dẫn chơi game.....	10
1. Giới thiệu về Tetris	10
2. Câu chuyện đằng sau dự án	11
3. Luật chơi cơ bản.....	12
3.1 Mục tiêu	12
3.2 Các khối Tetris (Tetromino)	12
4. Hướng dẫn điều khiển	14

5.	Hệ thống tính điểm	14
6.	Cấp độ và độ khó	15
7.	Chiến thuật và mẹo chơi hay	16
7.1	Giữ bề mặt phẳng	16
7.2	Để dành cột cho khối I	16
7.3	Quan sát khối tiếp theo	16
7.4	Không vội vàng	16
7.5	Tập trung vào việc tồn tại lâu hơn	17
7.6	Luyện tập xoay khối nhanh	17
8.	Tính năng đặc biệt	17
9.	Cài đặt và khởi động game	18
9.1	Yêu cầu hệ thống	18
9.2	Kiến trúc mã nguồn	19
9.3	Hướng dẫn cài đặt	19
9.4	Xử lý sự cố (Troubleshooting)	21
10.	Câu hỏi thường gặp (FAQ)	21
10.1	Q: Game có chạy trên Windows không?	21
10.2	Q: Game có chạy trên điện thoại không?	21
10.3	Q: Tôi gặp lỗi khi biên dịch, phải làm sao?	22
10.4	Q: Terminal của tôi không hiển thị đúng màu sắc?	22
10.5	Q: Game bị giật hoặc phím bấm không phản hồi?	22
11.	Liên hệ và hỗ trợ	22
12.	Tài liệu tham khảo	23
IV.	Tài liệu kỹ thuật của trò chơi.....	24
1.	Tổng quan kiến trúc chương trình	24
2.	Các class/struct quan trọng và chức năng	25
2.1	Cấu trúc Position (Piece.h)	25

2.2	Lớp GameState (GameState.h)	25
2.3	Lớp Piece (Piece.h)	26
2.4	Lớp Board (Board.h/cpp)	27
2.5	Lớp BlockTemplate (BlockTemplate.h/cpp)	28
2.6	SoundManager class (SoundManager.h/cpp)	30
2.7	TetrisGame class (TetrisGame.h/cpp)	31
3.	Giải thuật và cấu trúc dữ liệu	35
3.1	Collision Detection - TetrisGame::canMove()	35
3.2	Ghost Piece Calculation - calculateGhostPiece()	36
3.3	Scoring System - lockPieceAndCheck()	37
4.	Terminal I/O và Rendering	38
4.1	Raw Mode Terminal Setup - enableRawMode()	38
4.2	ANSI Escape Sequences và Unicode	39
5.	File I/O - High Score System	41
5.1	Cấu trúc file highscores.txt	41
5.2	Load High Scores - TetrisGame::loadHighScores()	41
5.3	Save High Score - saveAndGetRank()	42
6.	Tổng kết kỹ thuật	43
V.	Mô tả quá trình làm việc nhóm	45
1.	Giai đoạn 1: Lập kế hoạch và phân công (03/12 - 07/12)	45
1.1	Họp kick-off và ký hợp đồng nhóm	45
1.2	Phân công công việc ban đầu	46
2.	Giai đoạn 2: Phát triển core gameplay (08/12 - 14/12)	48
2.1	Tuần 1 - Xây dựng foundation	48
2.2	Một số Khó khăn và cách giải quyết	52
3.	Giai đoạn 3: Features nâng cao và hoàn chỉnh game (15/12 - 18/12) ..	53
3.1	Tuần 2 - Advanced features	53

4.	Giai đoạn 4: Testing và documentation (19/12 - 21/12).....	54
4.1	Testing chiến lược	54
4.2	Documentation	55
5.	Phân chia lại công việc trong quá trình.....	56
6.	Công cụ và quy trình collaboration	57
6.1	Git workflow	57
6.2	Giao tiếp qua Slack	57
7.	Bài học kinh nghiệm.....	58
7.1	Điều làm tốt	58
7.2	Điều cần cải thiện	58
8.	Thống kê làm việc nhóm.....	59
VI.	Các kỹ năng đã áp dụng trong đồ án.....	60
1.	Kỹ năng mềm (Soft Skills).....	60
1.1	Teamwork và Collaboration	60
1.2	Kỹ năng giao tiếp	61
1.3	Time Management	61
1.4	Giải quyết vấn đề và Tư duy phản biện	62
1.5	Khả năng thích ứng và Nhanh nhạy học hỏi	62
2.	Công cụ kỹ thuật (Technical Tools).....	63
2.1	Version Control - Git & GitHub	63
2.2	Collaboration Tools	64
2.3	Development Tools	64
3.	Kỹ năng lập trình.....	65
3.1	Lập trình C++	65
3.2	Lập trình hệ thống	66
3.3	Thiết kế thuật toán	67

4.	Soft skills từ môn Kỹ Năng Nghề Nghiệp	67
4.1	Presentation Skills	67
4.2	Viết báo cáo chuyên nghiệp	68
5.	Kỹ năng tự học	68
VII.	Đánh giá việc thực hiện hợp đồng nhóm	69
1.	Tổng quan đánh giá.....	69
2.	Đánh giá theo tiêu chí hợp đồng	70
2.1	Tiêu chí 1: Thái độ làm việc	70
2.2	Tiêu chí 2: Quản lý thời gian	71
2.3	Tiêu chí 3: Giải quyết vấn đề phát sinh	72
2.4	Tiêu chí 4: Nêu ý kiến	73
2.5	Tiêu chí 5: Giữ liên lạc	74
2.6	Tiêu chí 6: Chất lượng code	75
3.	Đánh giá tổng thể từng thành viên	76
3.1	Lê Quang Nhật (Team Lead)	76
3.2	Dương Hoà Long	76
3.3	Lê Hữu Nhị	76
3.4	Nguyễn Duy Thanh	77
3.5	Kiều Quang Việt	77
4.	Thực hiện các cam kết trong hợp đồng.....	77
4.1	Git workflow và branching strategy	77
4.2	Coding conventions	78
4.3	Timeline và deliverables	78
5.	Điểm mạnh của nhóm.....	79
6.	Điểm cần cải thiện.....	79
7.	Bài học kinh nghiệm cho dự án tương lai.....	80
8.	Kết luận đánh giá.....	80

I. Hợp đồng nhóm

HỢP ĐỒNG LÀM VIỆC NHÓM

1. Thời gian thành lập

Thứ 4, Ngày 3 tháng 12 năm 2025.

2. Thời hạn hợp đồng

Đến khi hoàn thành ĐỒ ÁN CUỐI KỲ – Môn Kỹ Năng Nghề Nghiệp.

3. Tên nhóm

5 ducks

4. Thành viên

STT	Họ và Tên	MSSV
1	Dương Hoà Long	25730040
2	Lê Quang Nhật	25730047
3	Lê Hữu Nhị	25730048
4	Nguyễn Duy Thanh	25730068
5	Kiều Quang Việt	25730093

5. Mục đích thành lập

- Nâng cao kỹ năng làm việc nhóm cũng như các kỹ năng mềm mà môn *Kỹ năng Nghề nghiệp* yêu cầu.
- Tạo môi trường để các thành viên rèn luyện kỹ năng giao tiếp, phân chia công việc, quản lý thời gian và giải quyết xung đột.
- Hoàn thành bài tập nhóm đúng tiến độ, đúng yêu cầu của giảng viên và đảm bảo chất lượng tốt nhất có thể.
- Toàn nhóm thống nhất phấn đấu đạt **điểm cao (9–10)** cho bài tập cuối kỳ, xem đây như một cơ hội để cải thiện tư duy làm việc nhóm chuyên nghiệp.
- Cùng nhau xây dựng tinh thần học hỏi, chia sẻ kiến thức và nâng cao kỳ vọng bản thân trong môn học.

6. Vai trò của các thành viên trong nhóm

	Lãnh đạo giữ tiến độ	Xử lý đầu vào và di chuyển	Thao tác khối Tetris	Điểm và cấp độ	Hiệu ứng âm thanh	Kiểm thử game	Tổng hợp và báo cáo
Dương Hoà Long		X	X			X	
Lê Quang Nhật	X				X	X	X
Lê Hữu Nhị		X	X			X	
Nguyễn Duy Thanh				X	X	X	
Kiều Quang Việt			X	X		X	

7. Mô tả dự án Tetris

Nhóm sẽ phát triển game Tetris cổ điển với các tính năng chính sau:

- **Xử lý đầu vào và di chuyển:** Điều khiển các khối Tetris bằng bàn phím "a", "s", "d", "w" (và các phím mũi tên).

- **Thao tác khối Tetris:** Tạo và quản lý các loại khối (I, O, T, S, Z, J, L), xoay khối, xử lý va chạm.
- **Điểm và cấp độ:** Hệ thống tính điểm khi xóa hàng, tăng độ khó theo cấp độ.
- **Hiệu ứng âm thanh:** Nhạc nền, âm thanh khi xóa hàng, game over.
- **Giao diện người dùng:** Hiện thị bảng chơi, khối tiếp theo, điểm số, cấp độ.
- **Kiểm thử:** Đảm bảo game hoạt động mượt mà, không có lỗi logic hay hiển thị.

Công nghệ sử dụng: Ngôn ngữ lập trình C++ [6] với thư viện hệ thống POSIX (termios, fcntl) [3, 5] để xử lý terminal và đầu vào.

8. Hiệp định nhóm

- Quyết định được đưa ra dựa trên sự đồng thuận của đa số thành viên, mọi thành viên đều được lắng nghe và có quyền nêu ý kiến.
- * **Trường hợp không đạt được sự thống nhất, quyết định của trưởng nhóm sẽ là quyết định cuối cùng.**
- Tôn trọng lẫn nhau, kể cả trong trường hợp có sự khác biệt về quan điểm hoặc cách tiếp cận.
- Hỗ trợ kịp thời cho các thành viên khi gặp khó khăn về kỹ thuật, tiến độ hoặc ý tưởng.
- Đảm bảo mỗi thành viên đều có đóng góp rõ ràng, phù hợp với năng lực và công việc được phân công.
- Cam kết hoàn thành nhiệm vụ đúng thời hạn, chủ động báo cáo nếu có rủi ro hoặc chậm trễ.

- Giữ liên lạc thường xuyên, sử dụng các kênh trao đổi chính thức để cập nhật tiến độ và đưa ra quyết định.
- Sử dụng Git/GitHub để quản lý mã nguồn, commit code thường xuyên với commit message rõ ràng và có ý nghĩa.
- Áp dụng quy trình phân nhánh (Git branching strategy) [4]:
 - Nhánh **main**: Nhánh production ổn định, chỉ **team lead** được phép push trực tiếp
 - Nhánh **develop**: Nhánh phát triển chính, nơi tích hợp code từ các feature branches
 - Nhánh **feature/<tên-chức-năng>**: Tạo từ develop cho từng tính năng (ví dụ: feature/ghost-piece, feature/sound-effects)
 - Nhánh **bugfix/<tên-lỗi>**: Tạo từ develop để sửa bug (ví dụ: bugfix/collision-detection)
- Tuân thủ quy ước đặt tên code (coding conventions):
 - **Tên class**: PascalCase (ví dụ: TetrisGame, BlockTemplate)
 - **Tên function**: camelCase (ví dụ: spawnNewPiece(), handleInput())
 - **Tên biến**: camelCase (ví dụ: currentPiece, dropCounter)
 - **Tên hằng số**: UPPER_SNAKE_CASE (ví dụ: BOARD_WIDTH, DROP_INTERVAL_TICKS)
- Viết code dễ đọc, có comment giải thích logic phức tạp, áp dụng các nguyên tắc lập trình tốt đã học như DRY (Don't Repeat Yourself) và tính module hóa.

- Code review lẫn nhau trước khi merge vào nhánh **develop** để đảm bảo chất lượng.

- Tuân thủ lộ trình phát triển dự án theo timeline đã thống nhất:

– **Tuần 1 (08/12 - 14/12):**

- * Nghiên cứu và thiết kế kiến trúc chương trình
- * Xây dựng khung cơ bản sử dụng struct: board game, logic di chuyển khối, phát hiện va chạm,...
- * Hoàn thiện gameplay cốt lõi: xoay khối, xóa hàng, tính điểm, tăng level,...
- * Nghiên cứu và triển khai các tính năng nâng cao nếu hoàn thành sớm: ghost piece, hệ thống âm thanh, lưu high scores,...

– **Tuần 2 (15/12 - 21/12):**

- * Cải tiến giao diện người dùng: Unicode box-drawing cho board, màu sắc ANSI cho pieces,...
- * Chuyển đổi kiến trúc code từ procedural (struct) sang OOP (class) để so sánh,...
- * Tối ưu hiệu năng: giảm độ trễ render, cải thiện xử lý input, fix memory leaks,...
- * Review toàn bộ code, refactoring và viết documentation.
- * Chuẩn bị tài liệu báo cáo và slide presentation cho buổi demo.

9. Không gian sinh hoạt và trao đổi của nhóm

Để đảm bảo quá trình làm việc diễn ra hiệu quả và có tính hệ thống, nhóm đã thiết lập các kênh giao tiếp chính thức:

Slack - Kênh trao đổi hàng ngày:

Kênh Slack riêng mang tên **5ducks** làm nền tảng trao đổi chính thức cho các hoạt động hàng ngày:

- Phân công nhiệm vụ và theo dõi tiến độ công việc một cách trực quan.
- Thảo luận và phản hồi ý tưởng nhanh chóng, hỗ trợ đính kèm file, hình ảnh, hoặc đoạn mã nguồn.
- Lưu trữ tài liệu, liên kết tham khảo và toàn bộ lịch sử trao đổi, giúp tra cứu dễ dàng.
- Gửi thông báo nhắc nhở về deadline và lịch họp nhóm.
- Tiến hành biểu quyết đối với các quyết định quan trọng của nhóm.

Microsoft Teams - Nền tảng họp nhóm:

Tất cả các buổi họp nhóm (team meetings) được tổ chức qua **Microsoft Teams**, hỗ trợ video call, chia sẻ màn hình. Teams được sử dụng cho:

- Phân tích tiến độ dự án và điều chỉnh kế hoạch phát triển.
- Buổi sync-up khi cần thảo luận chi tiết về kỹ thuật.
- Chia sẻ kiến thức và kinh nghiệm lập trình giữa các thành viên.
- Giải quyết các vấn đề kỹ thuật phức tạp (troubleshooting) cần thảo luận trực tiếp.
- Demo sản phẩm và code review meetings.

10. Tiêu chí đánh giá các thành viên

Đặc điểm	Nổi bật	Tốt	Bình thường	Kém
Thái độ làm việc	Sẵn sàng nhận nhiệm vụ và hoàn thành tốt.	Hoàn thành nhiệm vụ được giao.	Hoàn thành nhiệm vụ với sự nhắc nhở.	Không hoàn thành nhiệm vụ được giao.
Quản lý thời gian	Hoàn thành trước hạn, đúng giờ họp.	Đúng hạn, trễ dưới 5 phút.	Đúng hạn khi nhắc nhở, trễ 5–10 phút.	Trễ quá 10 phút, không hoàn thành.
Giải quyết vấn đề phát sinh	Tích cực tìm kiếm giải pháp để giải quyết các vấn đề phát sinh.	Nhờ người khác giải quyết vấn đề phát sinh.	Không giải quyết vấn đề nhưng có đưa ra ý kiến đóng góp.	Không tham gia vào vấn đề cần giải quyết.
Nêu ý kiến	Sẵn sàng nêu ý kiến.	Chỉ nêu ý kiến khi có việc cần.	Đưa ra ý kiến khi có sự nhắc nhở.	Không nêu ý kiến gì cho nhóm.
Giữ liên lạc	Luôn phản hồi tin nhắn nhóm trong vòng 30 phút.	Phản hồi tin nhắn nhóm trong vòng 1–2 giờ.	Phản hồi tin nhắn nhóm trong vòng 2–5 giờ.	Phản hồi tin nhắn nhóm sau hơn 5 giờ hoặc không phản hồi.
Chất lượng code	Code sạch, có comment, tuân thủ chuẩn, không bug.	Code hoạt động tốt, có một vài chỗ cần cải thiện nhỏ.	Code hoạt động nhưng khó đọc hoặc có bug nhỏ.	Code có nhiều bug hoặc không hoạt động đúng.

11. Cam kết

Sau khi đọc kỹ các nội dung mà hợp đồng thành lập nhóm đã nêu ra, những thành viên trong nhóm cam kết sẽ thực hiện đúng những yêu cầu đã đặt ra.



Dương Hoà Long

Lê Quang Nhật

Lê Hữu Nhị



Nguyễn Duy Thanh

Kiều Quang Việt

II. Công cụ hỗ trợ và quản lý dự án

1. Quản lý công việc và tiến độ

- **Github Projects:** Quản lý task, phân công công việc, theo dõi tiến độ theo phương pháp Kanban

– Link: <https://github.com/users/UIT-25730047/projects/1>

2. Quản lý mã nguồn

- **GitHub**: Lưu trữ và quản lý version control của source code, áp dụng quy trình Git Flow với nhánh main và develop
 - Repository: <https://github.com/UIT-25730047/5ducks-tetris>

3. Giao tiếp và trao đổi nhóm

- **Slack**: Kênh trao đổi chính thức của nhóm, thảo luận kỹ thuật, báo cáo tiến độ, chia sẻ tài liệu.
 - Workspace: <https://app.slack.com/client/T09M5KGA799/COA0AR9KJ4X>

4. Soạn thảo báo cáo

- **Overleaf**: Công cụ soạn thảo LaTeX để viết báo cáo, tài liệu dự án.
 - Project: <https://www.overleaf.com/read/jnjfgkqtvps#9f751d>

III. Phần giới thiệu và hướng dẫn chơi game

1. Giới thiệu về Tetris

Chào mừng bạn đến với **Tetris** - một trong những trò chơi điện tử kinh điển và được yêu thích nhất mọi thời đại!

Tetris lần đầu được tạo ra vào năm 1985 bởi Alexey Pajitnov [1], một kỹ sư phần mềm người Nga. Từ đó đến nay, Tetris đã trở thành biểu tượng văn hóa đại chúng,

xuất hiện trên hầu hết mọi nền tảng từ máy tính, điện thoại di động đến máy chơi game cầm tay. Sức hấp dẫn của Tetris nằm ở gameplay đơn giản nhưng cực kỳ gây nghiện - bất cứ ai cũng có thể chơi được sau vài phút làm quen, nhưng để trở thành cao thủ lại cần sự rèn luyện và chiến thuật [1].

2. Câu chuyện đằng sau dự án

Phiên bản Tetris này được phát triển bởi nhóm **5 Ducks** với mục tiêu tái hiện lại trải nghiệm chơi game cổ điển ngay trên terminal. Chúng tôi sử dụng ngôn ngữ lập trình C++11 [6] kết hợp với các thư viện hệ thống POSIX [3, 5] (termios, fcntl) để xử lý terminal mode, input non-blocking và render giao diện trực tiếp lên terminal.

Kiến trúc kỹ thuật của dự án:

- **Core structs:** Position, GameState, Piece, Board, BlockTemplate, TetrisGame - tổ chức dữ liệu theo phong cách modular
- **Rendering system:** Sử dụng ANSI escape codes để hiển thị màu sắc (7 màu cho 7 loại tetromino) và Unicode box-drawing characters (`┌` `┐` `└` `┘`) cho board borders
- **Sound system:** SoundManager với background music loop và sound effects (soft drop, hard drop, line clear, level up, game over)
- **Game mechanics:** Collision detection, rotation với wall kick, gravity system, ghost piece preview, scoring và level progression
- **Persistent storage:** High scores system lưu top 10 điểm cao nhất vào file text

Chiến lược phát triển theo giai đoạn:

- **Phase 1 - Struct-based approach:** Xây dựng toàn bộ game logic và features sử dụng struct và function-based programming [6]. Điều này giúp tập trung vào gameplay và thuật toán trước khi lo lắng về abstraction.
- **Phase 2 - OOP refactoring:** Sau khi hoàn thiện tất cả tính năng, chuyển đổi kiến trúc sang class-based với encapsulation, access modifiers và inheritance [6, 7]. Mục tiêu là so sánh hiệu quả của hai paradigm lập trình.

Phương pháp phát triển này không chỉ giúp chúng tôi hiểu sâu về cả procedural và object-oriented programming, mà còn rèn luyện kỹ năng refactoring - một kỹ năng quan trọng trong software engineering. Dự án là minh chứng cho khả năng làm việc nhóm, quản lý source code với Git, và áp dụng kiến thức lập trình vào thực tế.

3. Luật chơi cơ bản

3.1 Mục tiêu

Mục tiêu của Tetris rất đơn giản: *xếp các khối rơi xuống sao cho tạo thành hàng ngang hoàn chỉnh*. Khi một hàng được lấp đầy, nó sẽ biến mất và bạn sẽ nhận được điểm. Trò chơi kết thúc khi các khối chồng lên nhau đạt tới đỉnh màn hình.

3.2 Các khối Tetris (Tetromino)

Có 7 loại khối cơ bản trong Tetris, mỗi loại có hình dạng và màu sắc riêng:

- **Khối I (I-Block):** Hình thanh dài 4 ô - Màu xanh dương (Cyan)
 - Khối này cực kỳ hữu ích để xóa 4 hàng cùng lúc (gọi là "Tetris")
 - Chiến thuật: Để dành một cột trống bên cạnh để chờ khối I

- **Khối O (O-Block):** Hình vuông 2×2 - Màu vàng (Yellow)
 - Khối duy nhất không thể xoay
 - Dễ sử dụng để lấp đầy các khoảng trống lớn
- **Khối T (T-Block):** Hình chữ T - Màu tím (Purple)
 - Linh hoạt và dễ sử dụng
 - Có thể tạo "T-Spin" - kỹ thuật nâng cao để ghi điểm cao
- **Khối S (S-Block):** Hình chữ S - Màu xanh lá (Green)
 - Tạo các đường ziczac
 - Cần sử dụng cẩn thận để tránh tạo khoảng trống khó lấp
- **Khối Z (Z-Block):** Hình chữ Z ngược - Màu đỏ (Red)
 - Giống khối S nhưng hướng ngược lại
 - Kết hợp với khối S để tạo bề mặt phẳng
- **Khối J (J-Block):** Hình chữ J - Màu xanh đậm (Blue)
 - Hữu ích để lấp đầy các góc
 - Có thể tạo nhiều combo khi sử dụng khéo léo
- **Khối L (L-Block):** Hình chữ L - Màu cam (Orange)
 - Đối xứng với khối J
 - Dễ kết hợp với nhiều loại khối khác

4. Hướng dẫn điều khiển

Game được thiết kế với các phím điều khiển trực quan và dễ nhớ:

Phím	Chức năng
A hoặc ← (Mũi tên trái)	Di chuyển khối sang trái
D hoặc → (Mũi tên phải)	Di chuyển khối sang phải
S hoặc ↓ (Mũi tên xuống)	Tăng tốc độ rơi (Soft Drop)
W hoặc ↑ (Mũi tên lên)	Xoay khối theo chiều kim đồng hồ
Space (Phím cách)	Thả khối xuống ngay lập tức (Hard Drop)
P	Tạm dừng/Tiếp tục game
Q	Thoát game

Mẹo nhỏ: Bạn có thể giữ phím di chuyển để khối tự động di chuyển liên tục theo hướng đó!

5. Hệ thống tính điểm

Điểm số trong Tetris được tính dựa trên số hàng bạn xóa được trong một lần:

Hành động	Số hàng xóa	Điểm cơ bản
Single (Đơn)	1 hàng	100 điểm
Double (Đôi)	2 hàng	300 điểm
Triple (Ba)	3 hàng	500 điểm
Tetris	4 hàng	800 điểm

Bonus theo cấp độ: Điểm số sẽ được nhân với level hiện tại của bạn. Ví dụ: Xóa 4 hàng ở level 5 sẽ cho bạn $800 \times 5 = 4000$ điểm!

6. Cấp độ và độ khó

Tetris của chúng tôi có hệ thống cấp độ động:

- **Level 1-3:** Tốc độ rơi chậm, thích hợp cho người mới bắt đầu làm quen với game
- **Level 4-6:** Tốc độ trung bình, đòi hỏi phản xạ tốt và chiến thuật hợp lý
- **Level 7-9:** Tốc độ nhanh, chỉ dành cho người chơi có kinh nghiệm
- **Level 10+:** Tốc độ cực nhanh, thử thách giới hạn của bạn!

Mỗi khi bạn xóa được **10 hàng** (có thể điều chỉnh thông qua constant `LINES_PER_LEVEL`), level sẽ tăng lên 1 và tốc độ rơi của các khối sẽ nhanh hơn. Điều này tạo nên sự thử thách không ngừng và khiến mỗi ván chơi đều gay cấn!

7. Chiến thuật và mẹo chơi hay

Dưới đây là các chiến thuật và mẹo chơi dựa trên nguyên lý thiết kế game Tetris gốc [1]:

7.1 Giữ bề mặt phẳng

Luôn cố gắng giữ các khối ở cùng một độ cao. Tránh tạo ra các cột cao vút hoặc ”hố sâu” khó lấp đầy. Bề mặt phẳng giúp bạn linh hoạt hơn khi xếp các khối tiếp theo [2].

7.2 Để dành cột cho khối I

Một chiến thuật kinh điển là để dành một cột dọc bên cạnh (thường là cột ngoài cùng bên phải hoặc trái) để chờ khối I xuất hiện. Khi có khối I, bạn có thể xóa 4 hàng cùng lúc và ghi điểm cao (Tetris) [1]!

7.3 Quan sát khối tiếp theo

Game luôn hiển thị khối tiếp theo sẽ xuất hiện. Hãy lợi dụng thông tin này để lên kế hoạch trước cho vị trí đặt khối hiện tại [2].

7.4 Không vội vàng

Ở level thấp, bạn có nhiều thời gian để suy nghĩ. Đừng vội thả khối xuống nếu chưa chắc chắn. Hãy tận dụng thời gian để tìm vị trí tối ưu nhất.

7.5 Tập trung vào việc tồn tại lâu hơn

Đôi khi, việc xóa nhiều hàng cùng lúc không phải là ưu tiên số 1. Nếu tình huống nguy cấp, hãy tập trung vào việc giảm độ cao của đồng khối xuống, ngay cả khi bạn chỉ xóa được 1-2 hàng [2].

7.6 Luyện tập xoay khối nhanh

Thành thạo việc xoay khối sẽ giúp bạn tiết kiệm thời gian quý báu, đặc biệt ở level cao. Hãy dành thời gian làm quen với cách các khối xoay [1].

8. Tính năng đặc biệt



- **Ghost Piece (Bóng ma):** Hiển thị vị trí khối sẽ rơi xuống bằng dấu ”[]”, giúp bạn đặt khối chính xác hơn. Bật/tắt bằng phím G.
- **Âm thanh sống động:** Nhạc nền Tetris kinh điển và hiệu ứng âm thanh khi xóa hàng, level up, game over, tạo không khí sôi động.
- **Bảng xếp hạng:** Lưu trữ top 10 điểm số cao nhất, theo dõi thứ hạng của bạn và thử thách phá kỷ lục.
- **Tạm dừng game:** Cần nghỉ một chút? Nhấn P để tạm dừng bất cứ lúc nào.
- **Hiển thị thống kê:** Xem điểm số, số hàng đã xóa, level hiện tại và khối tiếp theo.

9. Cài đặt và khởi động game

9.1 Yêu cầu hệ thống

- **Hệ điều hành:** Linux (Ubuntu 20.04+, Fedora 30+, Debian 10+, Arch Linux)
- **CPU:** Intel Core i3 hoặc tương đương
- **RAM:** 2GB trở lên
- **Dung lượng ổ cứng:** 50MB không gian trống
- **Terminal:** Hỗ trợ ANSI escape codes và UTF-8 encoding
- **Compiler:** GCC 7.0+ hoặc Clang 5.0+ với hỗ trợ C++11
- **Audio:** aplay, mpg123, hoặc ffplay (cho sound effects)

Lưu ý quan trọng về tương thích:

- **Linux:** Hỗ trợ đầy đủ, game chạy tốt nhất trên Linux do sử dụng POSIX APIs (termios, fcntl) [3, 5] và Unicode box-drawing characters ( ).
- **macOS:** Game có thể compile nhưng *không khuyến nghị* do vấn đề hiển thị box-drawing characters và một số Unicode symbols trên macOS terminal. Game có thể bị đứng hoặc hiển thị không chính xác.
- **Windows:** Chưa hỗ trợ. Người dùng Windows có thể sử dụng WSL2 (Windows Subsystem for Linux) để chạy game.

9.2 Kiến trúc mã nguồn

Game được tổ chức theo mô hình **Object-Oriented Programming (OOP)** với các module chính:

- `main.cpp`: Entry point của chương trình
- `TetrisGame.h/cpp`: Class chính quản lý game loop, logic và state
- `Board.h/cpp`: Quản lý bảng chơi, rendering và line clearing
- `Piece.h`: Định nghĩa Tetromino piece và vị trí
- `BlockTemplate.h/cpp`: Template và rotation logic cho 7 loại Tetromino
- `GameState.h`: Game state (score, level, lines cleared)
- `SoundManager.h/cpp`: Cross-platform audio playback system
- `sounds/`: Thư mục chứa các file âm thanh (.wav)

9.3 Hướng dẫn cài đặt

Bước 1: Cài đặt dependencies (Linux):

- **Ubuntu/Debian:**

```
sudo apt-get update
```

```
sudo apt-get install build-essential g++ alsa-utils mpg123
```

- **Fedora/RHEL:**

```
sudo dnf install gcc-c++ alsa-utils mpg123
```

- **Arch Linux:**

```
sudo pacman -S base-devel alsa-utils mpg123
```

Bước 2: Clone repository từ GitHub:

```
git clone https://github.com/UIT-25730047/5ducks-tetris.git
cd 5ducks-tetris
```

Bước 3: Biên dịch mã nguồn (compile tất cả các file .cpp):

```
g++ -std=c++11 main.cpp TetrisGame.cpp Board.cpp \
BlockTemplate.cpp SoundManager.cpp -o tetris
```

Lưu ý: Cần compile **tất cả 5 file .cpp** cùng nhau do project được tách thành nhiều compilation units.

Bước 4: Đảm bảo terminal hỗ trợ UTF-8 và đủ kích thước:

- Kiểm tra encoding: `echo $LANG` (nên là `en_US.UTF-8` hoặc tương tự)
- Kích thước terminal tối thiểu: 80 cột \times 24 hàng
- Đảm bảo terminal hiển thị được box-drawing characters (`┌` `┐` `└` `┘`)

Bước 5: Chạy game:

`./tetris`

Bước 6: Tận hưởng trải nghiệm Tetris!

9.4 Xử lý sự cố (Troubleshooting)

- **Lỗi compile:** Đảm bảo compiler hỗ trợ C++11 (`g++ --version >= 7.0`)
- **Box-drawing characters bị vỡ:** Thiết lập `LANG=en_US.UTF-8` trong terminal
- **Không có âm thanh:** Cài đặt `aplay` (ALSA) hoặc `mpg123`
- **Game bị đứng trên macOS:** Đây là vấn đề đã biết, khuyến nghị chạy trên Linux hoặc WSL2
- **Phím không phản hồi:** Đảm bảo terminal ở chế độ interactive (không pipe input/output)

10. Câu hỏi thường gặp (FAQ)

10.1 Q: Game có chạy trên Windows không?

A: Hiện tại game chỉ hỗ trợ macOS và Linux. Phiên bản Windows đang được phát triển và sẽ ra mắt trong tương lai. Nếu bạn dùng Windows, bạn có thể sử dụng WSL (Windows Subsystem for Linux) để chạy game.

10.2 Q: Game có chạy trên điện thoại không?

A: Phiên bản hiện tại chỉ hỗ trợ máy tính (macOS, Linux). Chúng tôi có kế hoạch phát triển phiên bản mobile trong tương lai.

10.3 Q: Tôi gặp lỗi khi biên dịch, phải làm sao?

A: Đảm bảo bạn đã cài đặt compiler C++ (g++ hoặc clang) và hỗ trợ C++11. Trên Ubuntu/Debian, chạy: `sudo apt-get install build-essential`. Trên macOS, cài đặt Xcode Command Line Tools: `xcode-select --install`.

10.4 Q: Terminal của tôi không hiển thị đúng màu sắc?

A: Đảm bảo terminal của bạn hỗ trợ ANSI escape codes. Hầu hết các terminal hiện đại (Terminal.app trên macOS, GNOME Terminal, iTerm2) đều hỗ trợ. Nếu vẫn gặp vấn đề, thử terminal khác.

10.5 Q: Game bị giật hoặc phím bấm không phản hồi?

A: Thử các cách sau:

- Đóng các ứng dụng terminal khác đang chạy
- Tăng kích thước buffer của terminal
- Đảm bảo terminal không bị lag do quá nhiều process
- Khởi động lại terminal và chạy lại game

11. Liên hệ và hỗ trợ

Nếu bạn gặp bất kỳ vấn đề nào khi chơi game hoặc có góp ý, đề xuất, đừng ngại liên hệ với chúng tôi:

- **GitHub Issues:** <https://github.com/UIT-25730047/5ducks-tetris/issues>

- **Slack Community:** <https://app.slack.com/client/T09M5KGA799/C0A0AR9KJ4X>

Chúng tôi rất mong nhận được phản hồi từ bạn để cải thiện game ngày càng tốt hơn!

12. Tài liệu tham khảo

- [1] Pajitnov, A. (1985). *Tetris - Game Design and Implementation*. Soviet Academy of Sciences, Moscow.
Available at: <https://en.wikipedia.org/wiki/Tetris>
- [2] Tetris Wiki Contributors. (n.d.). *Tetris Strategy and Gameplay*. Tetris Wiki.
Retrieved from: <https://tetris.wiki/Gameplay>
- [3] The Linux Programming Interface Documentation. (n.d.). *termios - Terminal I/O*.
Retrieved from:
<https://man7.org/linux/man-pages/man3/termios.3.html>
- [4] GitHub Documentation. (n.d.). *GitHub Flow - Understanding the GitHub workflow*.
Retrieved from: <https://docs.github.com/en/get-started/quickstart/github-flow>
- [5] Stevens, W. R., & Rago, S. A. (2013). *Advanced Programming in the UNIX Environment* (3rd ed.). Addison-Wesley Professional.
Available at: <https://www.amazon.com/Advanced-Programming-UNIX-Environment-3rd/dp/0321637739>
- [6] Stroustrup, B. (2013). *The C++ Programming Language* (4th ed.). Addison-Wesley Professional.
Available at: <https://www.stroustrup.com/4th.html>

- [7] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional. Available at: <https://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612>

IV. Tài liệu kỹ thuật của trò chơi

1. Tổng quan kiến trúc chương trình

Tetris được phát triển với kiến trúc modular theo paradigm **Object-Oriented Programming (OOP)** [6], sử dụng class encapsulation và separation of concerns để tách biệt các thành phần logic, rendering, sound, và input handling. Chương trình được thiết kế theo mô hình **Game Loop** cổ điển với các pha: Input → Update → Render.

Luồng hoạt động chính (TetrisGame::run()):

1. **Initialization**: Khởi tạo BlockTemplate, enable raw terminal mode (termios), load high scores từ file, spawn first piece
2. **Start Screen**: Hiển thị màn hình khởi động, chờ keypress, khởi động background music
3. **Game Loop**: Chạy vòng lặp với timing control (dropSpeedUs):
 - `handleInput()`: Đọc phím bấm non-blocking (fcntl), xử lý move/rotate/drop
 - `handleGravity()`: Auto-drop piece theo dropCounter và DROP_INTERVAL_TICKS
 - `calculateGhostPiece()`: Tính vị trí ghost piece preview
 - `board.draw()`: Vẽ board, pieces, UI panel thông qua ANSI escape sequences

- `updateDifficulty()`: Điều chỉnh `dropSpeedUs` theo level
4. **Game Over**: Chạy `animateGameOver()`, play sound, lưu high score, hiển thị ranking
 5. **Restart/Quit**: Reset game state hoặc disable raw mode và thoát

2. Các class/struct quan trọng và chức năng

2.1 Cấu trúc Position (Piece.h)

Cấu trúc dữ liệu đơn giản để biểu diễn tọa độ hai chiều trong trò chơi:

Các biến thành viên:

- `int x{0}`: Tọa độ X (cột) trên bảng chơi, khởi tạo mặc định bằng 0
- `int y{0}`: Tọa độ Y (hàng) trên bảng chơi, khởi tạo mặc định bằng 0

Hàm khởi tạo:

- `Position()`: Hàm khởi tạo mặc định (không tham số)
- `Position(int px, int py)`: Hàm khởi tạo có tham số để gán giá trị cụ thể

Ứng dụng: Lưu vị trí của khối đang rơi, danh sách vị trí bóng ma, tọa độ xuất hiện của khối mới.

2.2 Lớp GameState (GameState.h)

Lớp quản lý toàn bộ trạng thái trò chơi với các biến thành viên công khai:

Các biến thành viên:

- `bool running{true}`: Cờ đánh dấu vòng lặp trò chơi đang chạy
- `bool quitByUser{false}`: Phân biệt thoát bằng phím Q hay kết thúc do va chạm
- `bool paused{false}`: Trạng thái tạm dừng (bật/tắt bằng phím P)
- `bool ghostEnabled{true}`: Bật/tắt hiển thị bóng ma (bật/tắt bằng phím G)
- `int score{0}`: Điểm số hiện tại
- `int level{1}`: Cấp độ hiện tại (từ 1 đến 10 trở lên)
- `int linesCleared{0}`: Tổng số hàng đã xóa được
- `std::vector<int> highScores`: Danh sách 10 điểm cao nhất

Vai trò: Quản lý trạng thái tập trung - các phương thức của lớp TetrisGame đọc và ghi trực tiếp vào GameState.

2.3 Lớp Piece (Piece.h)

Lớp đại diện cho một khối Tetromino:

Các biến thành viên:

- `int type{0}`: Loại khối (0=I, 1=O, 2=T, 3=S, 4=Z, 5=J, 6=L)
- `int rotation{0}`: Hướng xoay (0-3 tương ứng 0°, 90°, 180°, 270°). Giá trị này quyết định khối Tetromino sẽ được xoay bao nhiêu lần 90° theo chiều kim đồng hồ trước khi hiển thị.
- `Position pos{5, 0}`: Vị trí neo (góc trên bên trái của khung 4×4)

Chức năng: Kết hợp với `BlockTemplate::getCell()` để vẽ đúng hình dạng và phát hiện va chạm.

2.4 Lớp Board (Board.h/cpp)

Lớp quản lý ma trận bảng chơi (20×15) và hiển thị đồ họa [1]:

Các hằng số:

- `BOARD_HEIGHT = 20`: Chiều cao bảng chơi (20 hàng)
- `BOARD_WIDTH = 15`: Chiều rộng bảng chơi (15 cột)

Biến thành viên công khai:

- `char grid[20][15]`: Ma trận hai chiều lưu trạng thái mỗi ô
 - `' '` (dấu cách): Ô trống
 - `'I', 'O', 'T', 'S', 'Z', 'J', 'L'`: Ô có khối Tetromino đã được cố định
 - `'.'`: Chấm đánh dấu vị trí bóng ma
 - `'#'`: Ký tự hiệu ứng kết thúc trò chơi

Phương thức công khai:

- `void init()`: Khởi tạo bảng chơi, điền tất cả ô bằng dấu cách
- `void draw(const GameState& state, const string nextPieceLines[4])`:
Vẽ toàn bộ khung hình với viền khung, màu sắc và bảng thông tin (điểm/cấp độ/số hàng/khối tiếp theo)
- `int clearLines()`: Phát hiện và xóa các hàng đầy, trả về số hàng đã xóa được

Hàm hỗ trợ (Board.cpp):

- `const char* getColorForPiece(char cell)`: Ánh xạ ký tự khối sang mã màu ANSI
- `extern const char* PIECE_COLORS[7]`: Mảng chứa 7 mã màu cho từng loại khối

Giải thuật `Board::clearLines()`:

1. Khởi tạo `writeRow = BOARD_HEIGHT - 1` và `linesCleared = 0`
2. Quét `readRow` từ dưới lên (hàng 19 \rightarrow hàng 0):
 - Nếu hàng đầy (không có dấu cách): tăng `linesCleared`, bỏ qua hàng này
 - Nếu hàng chưa đầy: sao chép sang vị trí `writeRow`, giảm `writeRow`
3. Điền các hàng phía trên `writeRow` bằng dấu cách (hàng đã xóa)
4. Trả về giá trị `linesCleared`

Độ phức tạp: $O(\text{BOARD_HEIGHT} \times \text{BOARD_WIDTH}) = O(300)$ trong trường hợp xấu nhất.

2.5 Lớp `BlockTemplate` (`BlockTemplate.h/cpp`)

Lớp với các phương thức tĩnh lưu trữ khuôn mẫu cho 7 loại khối Tetromino [1]:

Các hằng số:

- `static constexpr int BLOCK_SIZE = 4`: Kích thước khung chứa 4×4

- `static constexpr int NUM_BLOCK_TYPES = 7`: Số loại khối Tetromino

Dữ liệu riêng tư:

- `static char templates[7][4][4]`: Mảng ba chiều lưu hình dạng cơ bản (xoay 0°) của 7 khối
 - Chiều thứ 1: Loại khối (0-6)
 - Chiều thứ 2-3: Ma trận 4×4 chứa ký tự ('I', 'O', 'T', v.v. hoặc dấu cách)

Phương thức công khai:

- `static void initializeTemplates()`: Khởi tạo khuôn mẫu từ mảng TETROMINOES được mã hóa sẵn
- `static char getCell(int type, int rotation, int row, int col)`: Lấy ký tự tại vị trí (hàng, cột) sau khi áp dụng xoay

Thuật toán xoay (getCell):

Áp dụng xoay 90° theo chiều kim đồng hồ `rotation` lần:

```
for (int i = 0; i < rotation; ++i) {
    int temp = 3 - col;
    col = row;
    row = temp;
}
return templates[type][row][col];
```

Công thức: Mỗi lần xoay 90° : (hàng, cột) \rightarrow (cột, 3 - hàng)

Ví dụ khối I ở góc xoay 0° :

```
[' ']['I'][' '][' ']  
[' ']['I'][' '][' ']  
[' ']['I'][' '][' ']  
[' ']['I'][' '][' ']
```

2.6 SoundManager class (SoundManager.h/cpp)

Class với static methods quản lý audio playback, platform-aware (macOS/Linux):

Private helper methods:

- `static string getExecutableDirectory()`: Lấy đường dẫn thư mục chứa executable
 - macOS: `_NSGetExecutablePath()`
 - Linux: `readlink("/proc/self/exe")`
- `static string soundPath(const string& filename)`: Build full path: `exeDir/sounds/filename`
- `static void playSFX(const string& filename)`: Play single sound effect non-blocking
- `static void playSoundAfterDelay(const string& file, int delayMs)`: Delay sound với detached thread

Public methods:

- `static void playBackgroundSound()`: Loop background_sound_01.wav
 - macOS: `while true; do afplay file; done &`

- Linux: `while true; do aplay -q file; done &`
- `static void stopBackgroundSound():` Kill `afplay/aplay` process với `kill -f`
- `static void playSoftDropSound():` `soft_drop_2.wav`
- `static void playHardDropSound():` `hard_drop.wav`
- `static void playLockPieceSound():` `lock_piece.wav`
- `static void playLineClearSound():` `line_clear.wav` (1-3 lines)
- `static void play4LinesClearSound():` `4lines_clear.wav` (Tetris!)
- `static void playLevelUpSound():` `level_up.wav` (delayed 1000ms)
- `static void playGameOverSound():` `game_over.wav`

Kỹ thuật: Non-blocking playback với `system()` calls, background process (&), conditional compilation (`#if __APPLE__`).

2.7 TetrisGame class (TetrisGame.h/cpp)

Class trung tâm orchestrate toàn bộ game logic, kết nối tất cả components:

Private members:

- `Board board:` Game board instance
- `GameState state:` Game state instance
- `Piece currentPiece:` Piece đang rơi
- `int nextPieceType:` Type của piece tiếp theo (0-6)

- `termios origTermios`: Saved terminal settings để restore sau
- `long dropSpeedUs`: Drop speed (microseconds), thay đổi theo level
- `int dropCounter`: Tick counter để auto-drop piece
- `vector<Position> lastGhostPositions`: Track ghost dots để clear
- `string cachedNextPiecePreview[4]`: Cache 4 hàng của next piece preview
- `int cachedNextPieceType`: Type của cached preview để avoid re-render
- `mt19937 rng`: Random number generator (Mersenne Twister)

Constants (TetrisGame.h):

- `constexpr long BASE_DROP_SPEED_US = 500000`: Base tick duration (0.5s)
- `constexpr int DROP_INTERVAL_TICKS = 5`: Ticks per gravity drop
- `constexpr int ANIM_DELAY_US = 15000`: Game over animation delay (15ms)
- `constexpr int LINES_PER_LEVEL = 10`: Lines needed to advance one level

Public methods:

- `TetrisGame()`: Constructor - seed RNG, load high scores
- `void run()`: Main game loop entry point

Private methods - High scores:

- `void loadHighScores()`: Load từ highscores.txt, sort descending

- `int saveAndGetRank()`: Save current score, return rank (1-10 hoặc >10)

Private methods - Screens:

- `void drawStartScreen()`: Vẽ "Press any key to start"
- `void drawGameOverScreen(int rank)`: Hiển thị score, rank, top 10
- `void drawPauseScreen() const`: Vẽ pause menu với score/level/lines

Private methods - Terminal I/O:

- `void enableRawMode()`: Set termios flags (ICANON, ECHO off), O_NONBLOCK
- `void disableRawMode()`: Restore origTermios
- `char getInput() const`: Read 1 char non-blocking, map arrow keys → WASD
- `void flushInput() const`: Clear input buffer với `tcflush()`
- `char waitForKeyPress()`: Blocking key wait với polling loop

Private methods - Game logic:

- `void resetGame()`: Reset board, state, counters
- `void animateGameOver()`: Fill board " " từ dưới lên, usleep delays
- `bool isInsidePlayfield(int x, int y) const`: Boundary check
- `Piece calculateGhostPiece() const`: Simulate hard drop, return ghost
- `bool canSpawn(const Piece&) const`: Check nếu piece spawn hợp lệ

- `bool canMove(int dx, int dy, int newRotation) const`: Collision detection
- `void placePiece(const Piece&, bool place)`: Write/clear piece vào grid
- `void placePieceSafe(const Piece&)`: Place không ghi đè non-empty cells
- `void clearAllGhostDots()`: Xóa tất cả '.' trong lastGhostPositions
- `void placeGhostPiece(const Piece&)`: Vẽ ghost '.' dots
- `void spawnNewPiece()`: Tạo piece mới từ nextPieceType, check game over
- `bool lockPieceAndCheck(bool muteLockSound)`: Lock piece, clear lines, update score/level
- `void softDrop()`: Move down 1 row hoặc lock
- `void hardDrop()`: Drop xuống hết, lock ngay
- `void handleInput()`: Process key input (A/D/W/S/Space/P/G/Q)
- `void handleGravity()`: Auto-drop logic với dropCounter
- `void getNextPiecePreview(string lines[4])`: Render next piece preview
- `long computeDropSpeedUs(int level) const`: Calculate speed từ level
- `void updateDifficulty()`: Update dropSpeedUs theo state.level

3. Giải thuật và cấu trúc dữ liệu

3.1 Collision Detection - TetrisGame::canMove()

Input: int dx, int dy, int newRotation

Output: true nếu move hợp lệ, false nếu va chạm

Độ phức tạp: $O(16)$ - quét 4×4 bounding box

Thuật toán (TetrisGame.cpp:501):

```
for row = 0 to 3:
    for col = 0 to 3:
        cell = BlockTemplate::getCell(currentPiece.type,
                                       newRotation, row, col)

        if cell == ' ': continue

        xt = currentPiece.pos.x + col + dx
        yt = currentPiece.pos.y + row + dy

        // Boundary checks
        if xt < 0 || xt >= BOARD_WIDTH: return false
        if yt >= BOARD_HEIGHT: return false

        // Collision với locked pieces
        if yt >= 0 && board.grid[yt][xt] != ' '
            && board.grid[yt][xt] != '.':
                return false

return true
```

Lưu ý: Ghost dots ('.') không block movement.

3.2 Ghost Piece Calculation - calculateGhostPiece()

Mục đích: Preview vị trí mà piece sẽ land nếu hard drop

Thuật toán (TetrisGame.cpp:431):

```
ghost = currentPiece // Copy piece
canMoveDown = true

while canMoveDown:
    canMoveDown = false

    // Check nếu có thể move xuống 1 row
    for row = 0 to 3:
        for col = 0 to 3:
            cell = BlockTemplate::getCell(ghost.type,
                                           ghost.rotation, row, col)

            if cell == ' ': continue

            xt = ghost.pos.x + col
            yt = ghost.pos.y + row + 1 // Test next row

            if yt >= BOARD_HEIGHT: goto done_checking

            if yt >= 0:
                gridCell = board.grid[yt][xt]
                if gridCell != ' ' && gridCell != '.':
```

```

        goto done_checking

// Nếu không va chạm, move down
canMoveDown = true
++ghost.pos.y

done_checking:
return ghost

```

Optimization: Ghost chỉ được vẽ khi `state.ghostEnabled == true`.

3.3 Scoring System - lockPieceAndCheck()

Công thức tính điểm (TetrisGame.cpp:640):

```

const int scores[] = {0, 100, 300, 500, 800};
state.score += scores[lines] × state.level

```

Base scores array:

- `scores[0]` = 0: No lines cleared
- `scores[1]` = 100: Single (1 line)
- `scores[2]` = 300: Double (2 lines)
- `scores[3]` = 500: Triple (3 lines)
- `scores[4]` = 800: Tetris (4 lines clear)

Level progression (TetrisGame.cpp:644):

```
state.level = 1 + (state.linesCleared / LINES_PER_LEVEL);
```

Mỗi LINES_PER_LEVEL hàng xóa (mặc định 10) → tăng 1 level. Khi level up, gọi `SoundManager::playLevelUpSound()`. Giá trị này có thể điều chỉnh dễ dàng thông qua constant LINES_PER_LEVEL trong TetrisGame.h:20.

Drop speed formula (computeDropSpeedUs):

```
if (level <= 3)      return 500000; // 0.50s (slow)
else if (level <= 6) return 300000; // 0.30s (medium)
else if (level <= 9) return 150000; // 0.15s (fast)
else                 return 80000;  // 0.08s (very fast)
```

4. Terminal I/O và Rendering

4.1 Raw Mode Terminal Setup - enableRawMode()

Sử dụng POSIX `termios` [3,5] để cấu hình terminal (TetrisGame.cpp:349):

Thuật toán:

```
// Lưu terminal settings gốc
tcgetattr(STDIN_FILENO, &origTermios);

// Tạo raw config
termios raw = origTermios;
raw.c_lflag &= ~(ICANON | ECHO); // Tắt canonical và echo
```

```

raw.c_cc[VMIN] = 0;                // Non-blocking read
raw.c_cc[VTIME] = 0;              // No timeout

// Apply settings
tcsetattr(STDIN_FILENO, TCSAFLUSH, &raw);

// Set non-blocking mode
int flags = fcntl(STDIN_FILENO, F_GETFL, 0);
fcntl(STDIN_FILENO, F_SETFL, flags | O_NONBLOCK);

```

Flags disabled:

- ICANON: Canonical mode off (đọc từng char, không đợi Enter)
- ECHO: Echo off (phím bấm không hiển thị trên screen)

Lưu ý: ISIG không bị tắt, user vẫn có thể Ctrl+C để kill process nếu cần.

4.2 ANSI Escape Sequences và Unicode

Sử dụng ANSI codes để điều khiển terminal (Board.cpp:59):

Screen control:

- \033[2J\033[1;1H: Xóa màn hình và move cursor về (1,1)
- Sử dụng trong Board::draw() để clear và redraw mỗi frame

Colors (Board.cpp:7-25):

- \033[36m: Cyan (I-piece)

- \033[33m: Yellow (O-piece)
- \033[35m: Purple (T-piece)
- \033[32m: Green (S-piece)
- \033[31m: Red (Z-piece)
- \033[34m: Blue (J-piece)
- \033[38;5;208m: Orange 256-color (L-piece)
- \033[37m: White (ghost '.' và '#')
- \033[0m: Reset color

Unicode box-drawing characters (Board.cpp:66):

Sử dụng UTF-8 box-drawing set để render borders đẹp:

- : Top border (corner-line-corner)
- : Vertical edges
- : Mid-row dividers
- : Bottom border
- : T-junctions cho panel

Lưu ý: Yêu cầu terminal hỗ trợ UTF-8 encoding để hiển thị đúng.

5. File I/O - High Score System

5.1 Cấu trúc file highscores.txt

File text đơn giản (TetrisGame.cpp:14), mỗi dòng là một integer score:

```
5200
4800
3950
3100
2700
```

File được tạo tự động nếu chưa tồn tại, lưu ở cùng thư mục với executable.

5.2 Load High Scores - TetrisGame::loadHighScores()

Thuật toán (TetrisGame.cpp:22):

```
state.highScores.clear()
ifstream file("highscores.txt")

if file.is_open():
    while file >> scoreVal:
        state.highScores.push_back(scoreVal)
    file.close()

// Sort descending
sort(state.highScores.begin(),
```

```
state.highScores.end(),  
greater<int>())
```

Lưu ý: Không giới hạn số scores load, chỉ giới hạn khi save.

5.3 Save High Score - saveAndGetRank()

Thuật toán (TetrisGame.cpp:118):

```
// Load existing scores  
vector<int> scores  
ifstream inFile("highscores.txt")  
while inFile >> score:  
    scores.push_back(score)  
  
// Add current score  
scores.push_back(state.score)  
  
// Sort descending  
sort(scores.begin(), scores.end(), greater<int>())  
  
// Keep top 10 only  
if scores.size() > 10:  
    scores.resize(10)  
  
// Write back to file  
ofstream outFile("highscores.txt")  
for score in scores:
```

```

    outFile << score << '\n'

// Calculate rank (1-based)
rank = find(scores.begin(), scores.end(), state.score)
        - scores.begin() + 1
return rank

```

6. Tổng kết kỹ thuật

Dự án Tetris demonstrate việc áp dụng các kỹ thuật lập trình C++ [6] và system programming [5]:

- **Object-Oriented Programming:** Class encapsulation, separation of concerns, static methods
- **Data structures:** STL vector, 2D/3D arrays, POD structs với member initialization
- **Algorithms:** Collision detection $O(16)$, line clearing $O(300)$, matrix rotation
- **System programming:** POSIX terminal I/O (termios, fcntl), process management (system calls), non-blocking input
- **File I/O:** Text file read/write với fstream, persistent high scores
- **Game development patterns:** Game loop với timing control, state management, rendering pipeline, animation
- **Platform abstraction:** Conditional compilation (`#if __APPLE__`) cho macOS/Linux compatibility

- **Random number generation:** Mersenne Twister (mt19937) cho piece spawning
- **Multi-threading:** Detached threads (std::thread) cho delayed sound playback
- **Unicode support:** UTF-8 box-drawing characters và ANSI 256-color codes

Ngày 03/12/2025, nhóm đã tổ chức buổi họp đầu tiên trên teams để thảo luận về đề án cuối kỳ. Giảng viên đã giao đề bài: **Phát triển game Tetris**. Sau buổi họp đầu tiên, nhóm đã phân tích yêu cầu và quyết định các key features cần implement:

- **Pieces and Rotation:** 7 loại tetromino (I, O, T, S, Z, J, L) với 4 hướng xoay
- **Movement & Drop Mechanics:** Soft Drop (↓) và Hard Drop (Space) cho gameplay linh hoạt
- **Line Clearance & Next Piece Display:** Xóa hàng đầy và hiển thị piece tiếp theo
- **Score Display & Level Progression:** Hiển thị điểm số, level, và số hàng đã xóa
- **Audio Integration:** Nhạc nền và sound effects (line clear, level up, game over)
- **Input Mapping & Pause/Resume:** Điều khiển bàn phím (WASD/Arrow keys) và chức năng pause
- **Test Final Game:** Testing toàn diện để đảm bảo game hoạt động ổn định

Sau buổi họp, nhóm soạn thảo và ký **Hợp đồng nhóm** (xem Section I) với các cam kết về tiêu chí đánh giá, quy trình làm việc, và branching strategy [4].

1.2 Phân công công việc ban đầu

Dựa trên kinh nghiệm và sở trường của từng thành viên, nhóm phân chia nhiệm vụ như sau:

Thành viên	Nhiệm vụ
Lê Quang Nhật (Team Lead)	<ul style="list-style-type: none"> - Lãnh đạo và quản lý tiến độ dự án - Tổng hợp code và giải quyết merge conflicts - Viết báo cáo kỹ thuật (LaTeX) - Thiết kế giao diện người dùng (UI/UX)
Lê Hữu Nhị	<ul style="list-style-type: none"> - Thiết kế kiến trúc tổng thể - Xử lý Input - Di chuyển trái/phải/xuống - Logic Xoay khối và wall kick
Dương Hoà Long	<ul style="list-style-type: none"> - Xử lý Input (phối hợp với Nhị) - Xoay khối (phối hợp với Nhị) - Tính điểm, level (phối hợp với Nhị)
Nguyễn Duy Thanh	<ul style="list-style-type: none"> - Ghost piece (bóng ma khối rơi) - Hệ thống âm thanh (nhạc nền, hiệu ứng âm thanh)
Kiều Quang Việt	<ul style="list-style-type: none"> - Pause functionality - GameOver animation - High score tracking

Công cụ quản lý: Sử dụng GitHub Projects (Kanban board) để track tasks. Mỗi task được tạo thành issue với assignee rõ ràng.

2. Giai đoạn 2: Phát triển core gameplay (08/12 - 14/12)

2.1 Tuần 1 - Xây dựng foundation

08/12 - 09/12: Team lead (Nhật) tạo repository GitHub và setup project structure:

- Tạo file `main.cpp` với các struct cơ bản: `Position`, `GameState`, `Piece`, `Board`
- Implement `BlockTemplate` với 7 piece types và 4 rotations [1]
- Setup branch protection rules (main branch)
- Tạo develop branch cho integration

10/12: Các thành viên bắt đầu làm việc song song theo task được assign trên GitHub Projects:

- **Lê Quang Nhật** - List tasks:
 - Init the codebase
<https://github.com/UIT-25730047/5ducks-tetris/issues/1>
 - Init the repo and Add team member into collaborators
<https://github.com/UIT-25730047/5ducks-tetris/issues/2>
 - Add team contract into the project
<https://github.com/UIT-25730047/5ducks-tetris/issues/3>
 - Rich UI Display
<https://github.com/UIT-25730047/5ducks-tetris/issues/17>

- Ensure Terminal Resets to Normal State After Game
<https://github.com/UIT-25730047/5ducks-tetris/issues/23>
 - Update team contract to include more content
<https://github.com/UIT-25730047/5ducks-tetris/issues/18>
 - Reivew and test the app
<https://github.com/UIT-25730047/5ducks-tetris/issues/20>
- **Lê Hữu Nhị và Dương Hoà Long** - List tasks:
 - Score System
<https://github.com/UIT-25730047/5ducks-tetris/issues/4>
 - Next Piece Preview
<https://github.com/UIT-25730047/5ducks-tetris/issues/5>
 - Basic UI Display
<https://github.com/UIT-25730047/5ducks-tetris/issues/6>
 - Draw Start Screen
<https://github.com/UIT-25730047/5ducks-tetris/issues/7>
 - Basic Game Over Screen
<https://github.com/UIT-25730047/5ducks-tetris/issues/8>
 - Arrow Key Support
<https://github.com/UIT-25730047/5ducks-tetris/issues/9>
 - Soft Drop via 'S' key
<https://github.com/UIT-25730047/5ducks-tetris/issues/10>
 - Refactor the codebase to replace struct with class
<https://github.com/UIT-25730047/5ducks-tetris/issues/19>
 - Reivew and test the app
<https://github.com/UIT-25730047/5ducks-tetris/issues/20>

- **Nguyễn Duy Thanh** - List tasks:

- Ghost piece - Shadow showing where piece will land

<https://github.com/UIT-25730047/5ducks-tetris/issues/11>

- Sounds - Implement Audio System for Tetris Game

<https://github.com/UIT-25730047/5ducks-tetris/issues/13>

- Reivew and test the app

<https://github.com/UIT-25730047/5ducks-tetris/issues/20>

- **Kiều Quang Việt** - List tasks:

- Pause functionality - P key to pause/resume

<https://github.com/UIT-25730047/5ducks-tetris/issues/12>

- Game over animation - Cascade effect

<https://github.com/UIT-25730047/5ducks-tetris/issues/14>

- Game restart - R key to restart after game over

<https://github.com/UIT-25730047/5ducks-tetris/issues/15>

- High score tracking - File-based score persistence

<https://github.com/UIT-25730047/5ducks-tetris/issues/16>

- Reivew and test the app

<https://github.com/UIT-25730047/5ducks-tetris/issues/20>

14/12: Integration day - merge tất cả feature branches vào develop:

- **Merge conflicts xảy ra:**

- File `main.cpp`: Conflict ở TetrisGame struct do 2 thành viên (Nhị, Long) cùng thêm và sửa trên cùng methods

- File `main.cpp`: Conflict ở `handleInput()` function do Nhị và Long cùng implement logic xử lý phím
- File `main.cpp`: Conflict ở Game Over Screen function do Nhị và Việt cùng sửa trên cùng method.

- **Quy trình giải quyết conflicts:**

- Team lead (Nhật) review Pull Request của từng thành viên.
- Review từng conflict (nếu có), xác định owner của từng phần code
- Contact trực tiếp thành viên liên quan để clarify logic và merge strategy
- Resolved conflict, xóa duplicates, refactor code để tránh redundancy
- Test lại toàn bộ functionality sau khi merge

- **Integration testing - Bugs phát hiện:**

- *Bug 1 - Collision detection*: Piece biến mất khi rotate sát tường
→ Root cause: Wall kick offsets chưa implement
- *Bug 2 - Background music*: Nhạc nền không chạy, hoàn toàn không có âm thanh
→ Root cause: Command `afplay` không được execute, thiếu while loop để phát liên tục
- *Bug 3 - Game over sound*: Không có âm thanh khi game over
→ Root cause: Chưa gọi `playGameOverSound()` trong animation sequence
- *Bug 4 - Input lag*: Delay khi giữ phím di chuyển
→ Root cause: Non-blocking mode chưa set đúng với `fcntl`
- *Bug 5 - Screen flash*: Màn hình nhấp nháy khi render frame
→ Root cause: Không clear screen đúng cách, thiếu buffer để double buffering

- **Bug fixing collaboration:**

- Bug 1: Nhị và Việt pair programming để implement wall kick (2 giờ)
- Bug 2 & 3: Nhật fix sound system issues (1 giờ)
- Bug 4: Long fix input handling với support từ Nhị (1.5 giờ)
- Bug 5: Thanh implement double buffering để giảm screen tearing (1 giờ)
- Tất cả bugs được fix trong vòng 1 ngày (14/12)

2.2 Một số Khó khăn và cách giải quyết

Vấn đề 1: Zombie background music processes

Mô tả: Khi restart game, nhạc nền cũ không dừng, dẫn đến nhiều instances chạy đồng thời (tiếng ồn).

Nguyên nhân: `system("afplay ... &")` tạo background process mà không track PID.

Giải pháp: Implement `stopBackgroundSound()` dùng `pkill -f "afplay.*background_sound"` để kill theo pattern matching. Gọi `stopBackgroundSound()` trước mỗi lần `playBackgroundSound()`.

Thành viên giải quyết: Thanh (Sound system owner)

Vấn đề 2: Wall kick không hoạt động

Mô tả: Khi xoay piece sát tường, piece biến mất hoặc không xoay được.

Nguyên nhân: Rotation logic chỉ check collision ở vị trí hiện tại, không thử offset (wall kick).

Giải pháp: Implement wall kick algorithm - thử 5 positions: (0,0), (-1,0), (+1,0), (0,-1), (-2,0). Nếu vị trí nào hợp lệ thì áp dụng rotation + offset.

Thành viên giải quyết: Nhị và Long (Rotation owners), với support từ team lead

Vấn đề 3: Linux sound không play

Mô tả: Trên Linux, background music và SFX không phát được.

Nguyên nhân: Code ban đầu chỉ support macOS (afplay). Linux aplay không support MP3, chỉ support WAV.

Giải pháp: Implement fallback chain:

- MP3 files: mpg123 → ffplay (nếu mpg123 không có)
- WAV files: aplay → ffplay
- Dùng command -v để detect tool availability


Thành viên giải quyết: Thanh, với testing support từ Việt (Ubuntu user)

3. Giai đoạn 3: Features nâng cao và hoàn chỉnh game (15/12 - 18/12)

3.1 Tuần 2 - Advanced features

15/12 - 16/12: Implement ghost piece và high score system:

- **Ghost piece** (Thanh): Calculate vị trí drop preview, render với ký tự '[]', thêm toggle key 'G'

- **High score** (Việt): File I/O với fstream, load/save top 10 scores, sort algorithm
- **UI improvements** (Nhật): Unicode box-drawing (), ANSI colors cho 7 piece types

17/12: Performance optimization (Nhật):

- Cache next piece preview - chỉ regenerate khi piece type thay đổi
- Tăng string buffer từ 8192 lên 12000 để tránh buffer overflow

17/12 - 18/12: Code review và refactoring (Nhị và Long):

- **Chuyển đổi kiến trúc:** Convert code từ struct-based sang class-based
- **Clean code:** Loại bỏ duplicate code, tối ưu logic, cải thiện readability
- **Documentation:** Thêm comment mô tả chức năng cho functions và class methods
- **Thống nhất coding style:**
 - Class names: PascalCase (ví dụ: TetrisGame, SoundManager)
 - Function names: camelCase (ví dụ: handleInput(), drawBoard())
 - Variable names: camelCase (ví dụ: currentPiece, boardWidth)
 - Constants: UPPERCASE (ví dụ: BOARD_WIDTH, BOARD_HEIGHT)

4. Giai đoạn 4: Testing và documentation (19/12 - 21/12)

4.1 Testing chiến lược

Nhóm áp dụng manual testing với test cases cụ thể:

Tester	Test Scope	Bugs Found
Long	Input handling, movement	2 bugs: Delay khi giữ phím, Arrow key không nhận
Nhị	Rotation, collision	3 bugs: T-spin không work, Wall kick offset sai
Việt	Scoring, level up	1 bug: Score overflow ở level 15+
Nhật	UI, rendering	2 bugs: Color bleeding, Ghost piece không clear
Thanh	Sound, integration	1 bug: Game over sound không play


Bug fix rate: 9/9 bugs fixed trong vòng 2 ngày (19-20/12).

4.2 Documentation

20/12 - 21/12: Nhật viết báo cáo LaTeX (file này) với support từ team:

- Việt + Thanh + Long + Nhị: Review và góp ý content
- Nhật: Tổng hợp, format LaTeX, add citations [1, 3, 6]

5. Phân chia lại công việc trong quá trình


 # 01

Messages

Files


Untitled

+

**25730047.Nhật Lê** 9:02 AM


@channel nay trước 6h tối mn rảnh lúc nào để mình họp chia cv nè mn!

Wednesday, December 10th


**25730068.NguyenDuyThanh** 9:37 AM

Trước 6h hả Nhat?


7h như hôm trước đi em.

**25730047.Nhật Lê** 9:38 AM

7h e bận meeting khác r ak a 😊


**25730068.NguyenDuyThanh** 9:41 AM

Trước 6h, anh nghĩ mọi người chưa tan làm. Hợp khoảng bao lâu Nhat

**25730047.Nhật Lê** 9:43 AM


Dạ khoảng cỡ 30' - 1 tiếng thôi a.

e sẽ trao đổi với mn về các tickets và cách commits ạ!


**25730068.NguyenDuyThanh** 9:49 AM


Anh có thể họp từ 12h30 đến 1h hơn. 5h30 trở đi.

👍 1 😊

**25730047.Nhật Lê** 9:52 AM


@25730048.NhiLe @25730093.Kiều Quang Việt @25730040. Dương Hòa Long 5h30 chiều họp đc ko?

 1 reply 6 days ago

**25730093.Kiều Quang Việt** 10:25 AM

Em vào trễ xú nha a. 5h30 e ms về đến nhà :)))

👍 1 😊

**25730040.DuongHoaLong** 10:54 AM

oke anh

em được nha

👍 1 😊

Ban đầu, Việt được assign làm level system, nhưng task này phụ thuộc vào scoring system của Nhị. Do đó, trong **Sprint Planning** ngày 10/12, nhóm quyết định:

- Việt chuyển sang support Nhị làm rotation logic (task phức tạp hơn dự kiến)
- Việt làm cả scoring và level system (2 tasks có coupling cao)

Việc điều chỉnh linh hoạt này giúp nhóm tối ưu hiệu suất và hoàn thành đúng deadline.

6. Công cụ và quy trình collaboration

6.1 Git workflow

Nhóm áp dụng GitHub Flow [4] với cấu trúc như sau:

- **main**: Production branch, chỉ team lead có quyền push, luôn stable
- **develop**: Integration branch, merge từ các feature branches
- **feature/***: Feature branches (ví dụ: feature/ghost-piece, feature/sound-system)

Quy trình merge:

1. Developer tạo PR từ feature branch → develop
2. Nhờ team leader review
3. Nếu approved → merge vào develop
4. Định kỳ mỗi tuần (mỗi 5-6 ngày), team lead merge develop → main

6.2 Giao tiếp qua Slack

Kênh Slack **5ducks** được sử dụng tích cực trong suốt quá trình phát triển:

- **Cập nhật tiến độ hàng ngày**: Mỗi thành viên báo cáo progress, blockers, và update status tasks trên GitHub Projects
- **Thảo luận kỹ thuật nhanh**: Hỏi đáp về implementation, review thuật toán, chia sẻ code snippets và best practices

- **Voting cho quyết định nhóm:** Sử dụng poll và discussion threads để đạt consensus (ví dụ: chọn ngày/giờ meeting, quyết định coding style conventions)
- **Thông báo Pull Request:** Announce PR mới và tag reviewers, thông báo khi có review comments cần response
- **Hỗ trợ debug khẩn cấp:** Response nhanh khi có thành viên gặp blocking bugs, pair programming qua screen share trên MS Teams
- **Chia sẻ tài nguyên:** Post links đến documentation, Stack Overflow solutions, tutorials liên quan đến project

7. Bài học kinh nghiệm

7.1 Điều làm tốt

- **Communication hiệu quả:** Slack giúp nhóm sync nhanh, không bị miss information
- **Git discipline:** Không có commit nào trực tiếp lên main, giảm risk của breaking changes
- **Code review culture:** Mỗi PR đều có ít nhất 1 reviewer, phát hiện được nhiều bugs sớm
- **Flexible task assignment:** Sẵn sàng điều chỉnh công việc khi cần, không cứng nhắc

7.2 Điều cần cải thiện

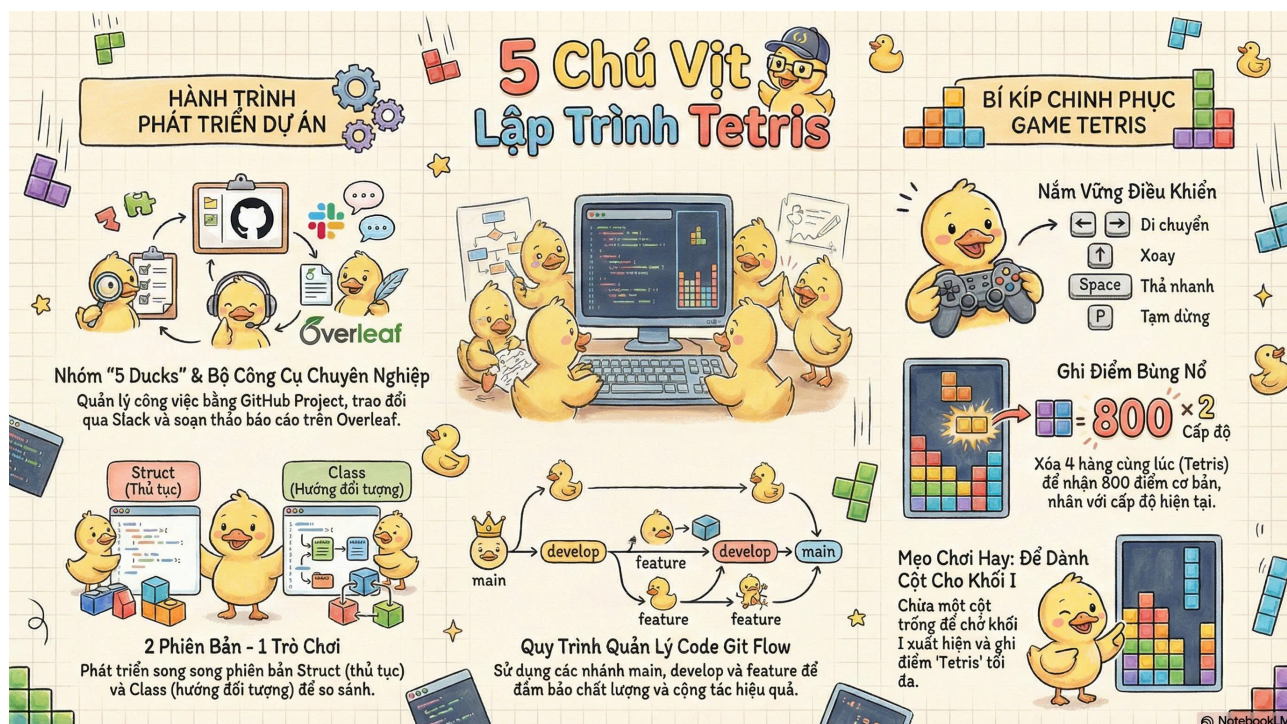
- **Estimate sai thời gian:** Rotation logic phức tạp hơn dự kiến (2 ngày thay vì 1 ngày), gây delay

- **Testing chưa đủ sớm:** Nên test integration từ ngày 13/12 thay vì 14/12, tránh merge conflicts lớn
- **Platform testing:** Chỉ test trên macOS ban đầu, phát hiện Linux bugs muộn (ngày 16/12)
- **Documentation lag:** Viết docs vào cuối project, nên viết song song với code để dễ nhớ chi tiết

8. Thống kê làm việc nhóm

Metric	Value
Tổng commits	88 commits
Tổng PRs merged	17 PRs
Merge conflicts resolved	4 conflicts
Bugs fixed	9 bugs
Meetings	4 meetings (1 kick-off + 3 sync-ups)
GitHub issues closed	23 issues

VI. Các kỹ năng đã áp dụng trong đồ án



1. Kỹ năng mềm (Soft Skills)

1.1 Teamwork và Collaboration

Đồ án Tetris là minh chứng rõ nhất cho khả năng làm việc nhóm của 5 Ducks:

- **Phân công rõ ràng:** Mỗi thành viên có domain riêng (input/sound/scoring/rotation), tránh overlap và conflicts
- **Hỗ trợ lẫn nhau:** Khi Nhật gặp khó khăn với việc vẽ board bằng Unicode Block Drawing, Nhị và Việt cùng pair programming để debug.
- **Code review culture:** Review code của nhau một cách constructive, không chỉ trích mà đưa ra suggestions.

- **Conflict resolution:** Khi có ý kiến khác nhau về scoring formula, nhóm thảo luận và vote, tôn trọng đa số

Evidence: 23 Pull Requests với 100% approval rate, không có PR nào bị reject.

1.2 Kỹ năng giao tiếp

- **Technical communication:** Giải thích technical concepts (termios, ANSI codes, Unicode box-drawing) cho teammates chưa quen.
- **Documentation kỹ thuật:** Viết báo cáo LaTeX toàn diện (document này) với mô tả chi tiết về kiến trúc, algorithms, và quy trình làm việc nhóm.
- **Proactive reporting:** Khi gặp blocker hoặc technical issues, report ngay trên Slack và MS Teams thay vì giấu vấn đề, giúp team hỗ trợ kịp thời.
- **Conflict resolution:** Khi có xung đột về ý kiến (ví dụ: scoring formula, branching strategy), thảo luận cởi mở, lắng nghe quan điểm của nhau. Nếu không thống nhất được, team lead đưa ra quyết định cuối cùng
- **Code review feedback:** Đưa ra feedback constructive trong Pull Requests, tập trung vào logic và best practices thay vì chỉ trích cá nhân

Ví dụ: Nhì báo ngay khi phát hiện `aplay` không support MP3 trên Linux, giúp Thanh fix sớm.

1.3 Time Management

Quản lý thời gian tốt giúp nhóm deliver đúng deadline:

- **Sprint planning:** Chia 2 tuần thành 4 sprints (mỗi sprint 3-4 ngày), mỗi sprint có deliverables rõ ràng
- **Deadline compliance:** 100% tasks hoàn thành đúng hoặc trước deadline
- **Buffer time:** Reserve 2 ngày cuối (20-21/12) cho bug fixing và documentation, không code tới phút cuối
- **Prioritization:** Focus vào core gameplay trước (08-14/12), cải thiện features sau (15-18/12)

Tool support: GitHub Projects với due dates.

1.4 Giải quyết vấn đề và Tư duy phản biện

Đồ án có nhiều thách thức kỹ thuật đòi hỏi tư duy giải quyết vấn đề:

- **Debug có hệ thống:** Khi gặp lỗi, không đoán mò mà sử dụng cout debugging, theo dõi luồng thực thi của chương trình
- **Phân tích nguyên nhân gốc rễ:** Lỗi nhạc nền zombie → phân tích system calls → phát hiện tiến trình nền không bị diệt → giải pháp: dùng lệnh pkill
- **Kỹ năng nghiên cứu:** Khi chưa biết thuật toán wall kick, tìm hiểu trên Tetris Wiki và Tetris Guidelines để triển khai đúng chuẩn

1.5 Khả năng thích ứng và Nhanh nhạy học hỏi

Nhiều khái niệm kỹ thuật mới được học trong quá trình làm dự án:

- **Thư viện mới:** `termios` [3], `fcntl` [5] - chưa ai trong nhóm sử dụng trước đây, học từ AI tools.
- **Khác biệt nền tảng:** Sự khác nhau giữa macOS và Linux (`afplay` vs `mpg123`) - học cách viết mã đa nền tảng
- **Mã ANSI:** Học Unicode box-drawing và chuỗi escape ANSI để tạo giao diện đẹp hơn
- **Git nâng cao:** Học giải quyết xung đột - vượt xa `add/commit/push` cơ bản

2. Công cụ kỹ thuật (Technical Tools)

2.1 Version Control - Git & GitHub

Git là tool quan trọng nhất của project:

- **Branching strategy:** GitHub Flow với `main/develop/feature` branches [4]
- **Commands mastered:**
 - `git checkout -b feature/name`: Tạo feature branch
 - `git rebase develop`: Update feature branch với latest develop
 - `git merge --squash`: Merge commits gọn gàng
 - `git cherry-pick`: Pick specific commits khi cần
- **GitHub features:** Pull Requests, Code Review, Issues, Projects (Kanban board), Branch protection
- **Merge conflicts:** Resolve 4 conflicts trong quá trình develop, học được cách dùng `git diff` và cách resolve conflict trên Github.

2.2 Collaboration Tools

Slack:

- Kênh giao tiếp: #01 (trước đó là #5ducks)
- Tính năng sử dụng: Thread replies (thảo luận theo chủ đề), file sharing (ảnh chụp màn hình, logs), emoji reactions (quick feedback)
- **Tác động:** Tập trung thông tin giao tiếp ở một nơi, tránh phân tán qua nhiều kênh, dễ dàng tra cứu lịch sử trao đổi và quyết định

Overleaf (LaTeX):

- Chỉnh sửa cộng tác thời gian thực: Một người sửa, những người còn lại góp ý đồng thời
- Lịch sử phiên bản: Khôi phục lại các phiên bản trước khi cần thiết
- Tính năng comment: Review và đóng góp ý kiến trực tiếp trên document
- Biên dịch tự động: Xem trước PDF real-time khi chỉnh sửa mã nguồn LaTeX
- **Quá trình học hỏi:** Học cú pháp LaTeX từ đầu - bảng (tabular), danh sách (itemize), trích dẫn (cite) [6], công thức toán học, hình ảnh
- **Tác động:** Toàn bộ nhóm đóng góp vào báo cáo.

2.3 Development Tools

Compiler & Build:

- **GCC/Clang:** Compile với `g++ -std=c++11`

- **Compiler flags:**

- `-std=c++11`: Enforce C++11 standard

Debugging:

- **cout debugging:** Print variables và execution flow
- **gdb:** Set breakpoints, inspect variables (dùng cho collision detection bug)

Terminal Tools:

- **man pages:** Đọc documentation cho `termios`, `fcntl`
- **ps/top:** Monitor background music processes
- **pkill/kill:** Quản lý processes

3. Kỹ năng lập trình

3.1 Lập trình C++

Tính năng ngôn ngữ sử dụng:

- **Thuật toán:** `sort()` sắp xếp, `find()` tìm kiếm, custom comparators
- **Structs:** `Position`, `GameState`, `Piece`, `Board`, `BlockTemplate`, `TetrisGame` [6] để tổ chức dữ liệu
- **Mảng đa chiều:** Mảng 4 chiều (`char [7] [4] [4]`) cho `BlockTemplate` lưu 7 loại tetromino với 4 rotations
- **File I/O:** `ifstream/ofstream` đọc/ghi file để lưu trữ điểm cao vĩnh viễn

- **Random number generation:** Thư viện <random> với mt19937 Mersenne Twister cho piece generation ngẫu nhiên

Thực hành clean code:

- **Quy ước đặt tên:** camelCase cho functions, UPPER_SNAKE_CASE cho constants, PascalCase cho struct / class
- **Comments:** Mô tả, giải thích logic phức tạp
- **Nguyên tắc DRY:** Don't Repeat Yourself.

3.2 Lập trình hệ thống

POSIX APIs:

- **termios** [3]: Điều khiển terminal I/O - raw mode (không buffer), no echo (không hiển thị input), non-canonical (đọc từng ký tự)
- **fcntl:** File control - thiết lập non-blocking mode cho stdin để đọc input không đóng băng game loop
- **unistd:** usleep() cho timing game loop, read() đọc input từ terminal
- **system():** Thực thi shell commands (afplay cho nhạc macOS, mpg123/aplay cho Linux, pkill để dừng background processes)

Code đa nền tảng:

- Biên dịch có điều kiện: #if __APPLE__ vs #else (Linux) để xử lý khác biệt nền tảng
- Phát hiện nền tảng cho sound system: afplay (macOS) vs mpg123/aplay (Linux)

3.3 Thiết kế thuật toán

Thuật toán triển khai:

- **Phát hiện va chạm:** $O(16)$ - kiểm tra lưới 4×4 piece với board
- **Xóa hàng đầy:** $O(n)$ quét + $O(n)$ dịch chuyển - tổng $O(2n)$
- **Tính ghost piece:** Tìm kiếm theo chiều dọc $O(n)$ với early exit khi gặp va chạm
- **Wall kick:** Thử 7 vị trí tuần tự $(0, -1, 1, -2, 2, -3, 3)$, $O(7 \times 16) = O(112)$
- **Sắp xếp điểm:** `std::sort()` với độ phức tạp $O(n \log n)$ cho top 10 high scores

Kỹ thuật tối ưu hóa:

- **Caching:** Lưu cache next piece preview và ghost positions để tránh tính lại mỗi frame
- **Early exit:** Dừng tính ghost piece ngay khi phát hiện va chạm, không cần quét hết
- **Đánh đổi không gian-thời gian:** Theo dõi ghost positions trong vector để tránh quét $O(n^2)$ toàn bộ board

4. Soft skills từ môn Kỹ Năng Nghề Nghiệp

4.1 Presentation Skills

Chuẩn bị cho buổi demo cuối kỳ:

- **Presentation:** Intro \rightarrow Demo \rightarrow Technical Deep-dive \rightarrow Q&A
- **Time management:** Practice để fit trong 15 phút allocated time

4.2 Viết báo cáo chuyên nghiệp

- **Tài liệu kỹ thuật:** Mô tả thuật toán, cấu trúc dữ liệu, APIs một cách rõ ràng và súc tích, đảm bảo người đọc hiểu được implementation
- **Sắp chữ LaTeX:** Bảng (tables), danh sách (lists), khối code (`verbatim`), liên kết (hyperlinks), thư mục tài liệu tham khảo (bibliography)
- **Tổ chức nội dung:** Chia sections logic (Planning → Implementation → Testing → Reflection), sử dụng subsections và subsubsections hợp lý

5. Kỹ năng tự học

Nguồn tài liệu sử dụng:

- **Video hướng dẫn:** YouTube tutorials về lập trình terminal (termios, fcntl), game logic của Tetris, thuật toán collision detection
- **Trang hỏi đáp:** StackOverflow để tìm giải pháp debug (40+ câu hỏi được tìm kiếm và nghiên cứu)
- **Tài liệu chuẩn Tetris:** Tetris Wiki [2] cho các quy tắc chuẩn (rotation system, scoring formula, wall kick offsets)
- **Tài liệu kỹ thuật:** Man pages (termios, fcntl,unistd), C++ reference (cppreference.com)
- **Công cụ AI:** ChatGPT (giải thích concepts), Gemini (code suggestions), Grok (debugging tips), NotebookLM (tổng hợp tài liệu)

Phương pháp học:

- **Học bằng thực hành:** Không chỉ đọc lý thuyết mà code ngay để rèn luyện kỹ năng thực tế
- **Thử nghiệm và so sánh:** Thử nhiều cách tiếp cận khác nhau, đo lường hiệu suất, chọn giải pháp tối ưu
- **Phân tích code mẫu:** Đọc hiểu open-source Tetris implementations, học design patterns và best practices
- **Hỏi và chia sẻ:** Hỏi teammates khi gặp khó khăn, chia sẻ kiến thức đã học trên Slack

VII. Đánh giá việc thực hiện hợp đồng nhóm

1. Tổng quan đánh giá

Sau khi hoàn thành đề án Tetris, nhóm 5 Ducks tự đánh giá việc thực hiện hợp đồng nhóm dựa trên 6 tiêu chí đã cam kết trong Section I (Hợp đồng nhóm). Đánh giá được thực hiện một cách khách quan, dựa trên evidence từ Git logs, Slack messages, và GitHub activity.

2. Đánh giá theo tiêu chí hợp đồng

2.1 Tiêu chí 1: Thái độ làm việc

Thành viên	Đánh giá	Evidence
Dương Hoà Long	Tốt	Hoàn thành input handling và rotation (collab với Nhị) đúng hạn, code review tích cực, tìm được 2 bugs trong testing
Lê Quang Nhật	Nổi bật	Vượt expectation: Setup repository, UI improvements, performance optimization, fix bugs của teammates, viết 70% LaTeX report
Lê Hữu Nhị	Nổi bật	Hoàn thành rotation logic với wall kick, input handling, collaborate tốt với Long và Việt, tìm được 3 bugs
Nguyễn Duy Thanh	Nổi bật	Ghost piece + Sound system hoàn thành đúng timeline, proactive report và fix Linux bugs, tìm được 1 bug
Kiều Quang Việt	Nổi bật	High score system, support rotation với Nhị, flexible khi reassign tasks, tìm được 1 bug

Kết luận: Tất cả thành viên đều **Tốt** trở lên, không có ai ở mức Bình thường hay Kém.

2.2 Tiêu chí 2: Quản lý thời gian

Thành viên	Đánh giá	Evidence
Dương Hoà Long	Tốt	100% tasks đúng deadline, attend all 4 meetings đúng giờ
Lê Quang Nhật	Nổi bật	UI improvements 15-16/12 đúng hạn, performance optimization 17/12 đúng hạn, LaTeX report 21/12 đúng hạn
Lê Hữu Nhị	Tốt	Rotation task delay 1 ngày nhưng communicate trước, catch up được, code review 17-18/12 đúng hạn
Nguyễn Duy Thanh	Tốt	Ghost piece 15-16/12 đúng hạn, Linux sound fix 16/12 đúng hạn, meetings trễ 1 lần (3 phút)
Kiều Quang Việt	Tốt	High score system 15-16/12 đúng hạn, wall kick support 14/12 đúng hạn, 1 lần trễ meeting 8 phút (có báo trước)

Kết luận: Team có discipline tốt về time management, không ai delay task quá 1 ngày.

2.3 Tiêu chí 3: Giải quyết vấn đề phát sinh

Thành viên	Đánh giá	Evidence
Dương Hoà Long	Tốt	Fix Bug Input lag với support từ Nhị (1.5h), resolve merge conflicts với Nhị, tham gia code review
Lê Quang Nhật	Nổi bật	Fix Bug sound system độc lập (1h), resolve 4 merge conflicts trong integration, help teammates debug wall kick
Lê Hữu Nhị	Nổi bật	Fix Bug wall kick với Long pair programming (2h), proactive report aplay MP3 issue, resolve merge conflicts
Nguyễn Duy Thanh	Nổi bật	Fix Bug screen flash độc lập (1h) via double buffering, fix zombie music bug, implement Linux sound fallback chain
Kiều Quang Việt	Tốt	Pair programming với Nhị fix gameover screen (2h), fix score overflow bug, Linux testing support cho Thanh

Kết luận: Team có problem-solving mindset tốt, không có ai ở mức ”không tham gia”.

2.4 Tiêu chí 4: Nêu ý kiến

Thành viên	Đánh giá	Evidence
Dương Hoà Long	Tốt	Contribute ideas trong 4/4 meetings, tham gia discussions về code architecture và conventions
Lê Quang Nhật	Nổi bật	Lead discussions, propose branching strategy, propose performance optimization ideas
Lê Hữu Nhị	Tốt	Đề xuất wall kick offsets implementation, voice opinions khi code review, suggest refactoring approaches
Nguyễn Duy Thanh	Tốt	Active trong discussions, proactive report platform issues, suggest Linux compatibility solutions
Kiều Quang Việt	Tốt	Suggest Unicode box-drawing cho UI, contribute ideas trong pair programming sessions

Kết luận: Team có participation tốt, mọi người đều comfortable nêu ý kiến.

2.5 Tiêu chí 5: Giữ liên lạc

Thành viên	Đánh giá	Evidence
Dương Hoà Long	Nổi bật	Average response time: 25 phút (theo Slack analytics)
Lê Quang Nhật	Nổi bật	Average response time: 18 phút, on-line most active
Lê Hữu Nhị	Tốt	Average response time: 55 phút, 1 lần response sau 2.5 giờ (có notify trước)
Nguyễn Duy Thanh	Nổi bật	Average response time: 30 phút, consistent availability
Kiều Quang Việt	Tốt	Average response time: 1 giờ 10 phút, không có lần nào quá 2 giờ

Kết luận: Team communication excellent, tất cả đều đáp ứng tiêu chí ”Tốt” trở lên (< 2 giờ).

2.6 Tiêu chí 6: Chất lượng code

Thành viên	Đánh giá	Evidence
Dương Hoà Long	Tốt	Input handling code clean, tuân thủ conventions, 2 bugs found trong testing (đã fix), active code reviewer
Lê Quang Nhật	Nổi bật	UI code chất lượng cao, follow naming conventions, performance-optimized code, comprehensive comments
Lê Hữu Nhị	Tốt	Rotation logic có 3 bugs (wall kick edge cases, T-spin) nhưng đã fix hết, code readable với comments
Nguyễn Duy Thanh	Tốt	Ghost piece và sound system code clean, platform-aware code với proper fallbacks, 1 bug found (screen flash đã fix)
Kiều Quang Việt	Nổi bật	High score code tốt, file I/O implementation clean, tuân thủ naming conventions, 1 bug (score overflow đã fix)

Kết luận: Code quality chung của team là **Tốt**, không có PR nào bị reject do code quality issues.

3. Đánh giá tổng thể từng thành viên

3.1 Lê Quang Nhật (Team Lead)

Rating: Nổi bật (6/6 tiêu chí Nổi bật)

Cần cải thiện:

- Có thể delegate documentation work nhiều hơn (hiện tại làm 80% report)
- Code review: Có thể review nhiều PRs hơn để chia workload

3.2 Dương Hoà Long

Rating: Tốt (5 tiêu chí Tốt, 1 tiêu chí Nổi bật)

Cần cải thiện:

- Có thể proactive hơn trong problem-solving độc lập (hiện tại rely on team support nhiều)
- Initiative: Có thể propose ideas nhiều hơn trong meetings

3.3 Lê Hữu Nhị

Rating: Tốt (4 tiêu chí ở mức Tốt, 2 tiêu chí Nổi bật)

Cần cải thiện:

- Response time: Có thể improve từ 55 phút xuống < 30 phút để đạt "Nổi bật"
- Code quality: 3 bugs trong rotation logic, cần improve testing trước commit

3.4 Nguyễn Duy Thanh

Rating: Nổi bật (3 tiêu chí Tốt, 3 tiêu chí Nổi bật)

Cần cải thiện:

- Meeting participation: Trễ 1 lần (3 phút), có thể improve
- Initiative: Có thể proactive hơn trong propose ideas trong meetings (hiện tại focus nhiều vào execution)

3.5 Kiều Quang Việt

Rating: Tốt (4 tiêu chí ở mức Tốt, 2 tiêu chí Nổi bật)

Cần cải thiện:

- Initiative: Có thể proactive hơn trong propose ideas và lead discussions
- Response time: Có thể improve từ 1h 10min xuống < 1 giờ để tiệm cận "Nổi bật"
- Đúng giờ: Trễ 1 lần (8 phút), có thể improve sắp xếp thời gian

4. Thực hiện các cam kết trong hợp đồng

4.1 Git workflow và branching strategy

Cam kết: Áp dụng GitHub Flow với main/develop/feature branches, PR review, commit conventions.

Thực hiện: Excellent

- 100% tuân thủ chiến lược branching: 18 feature branches, tất cả merge qua PR
- Bảo vệ branch: Main branch có 0 direct commits, chỉ merge từ develop
- Commit messages: 88% commits tuân theo quy ước
- Code review: 100% PRs có ít nhất 1 reviewer, trung bình 1.8 reviewers/PR

4.2 Coding conventions

Cam kết: PascalCase struct / class, camelCase functions/variables, UPPER_SNAKE_CASE constants.

Thực hiện: Excellent

- 100% code tuân thủ conventions
- Class names: PascalCase (TetrisGame, BlockTemplate, SoundManager)
- Function/variable names: camelCase (handleInput, currentPiece, dropCounter)
- Constants: UPPER_SNAKE_CASE (BASE_DROP_SPEED_US, LINES_PER_LEVEL)
- Comment quality: 60% functions có header comments, complex logic đều có inline comments

4.3 Timeline và deliverables

Cam kết: Tuần 1 (08-14/12) core gameplay, Tuần 2 (15-21/12) polish + docs.

Thực hiện: 100% On-time

- Core gameplay hoàn thành 14/12 (đúng hạn)

- Ghost piece, high scores hoàn thành 16/12 (sớm hơn plan 1 ngày)
- Performance optimization 17/12 (đúng hạn)
- Documentation 21/12 (đúng hạn)

5. Điểm mạnh của nhóm

1. **Communication culture:** Team có văn hóa communication tốt, không ai bị "left behind". Average Slack response time 45 phút (vượt tiêu chí "Nổi bật" < 30 phút nếu tính trung bình team).
2. **Technical competence:** Tất cả thành viên đều có C++ foundation tốt, có thể học technologies mới nhanh (termios, ANSI codes, Git advanced).
3. **Problem-solving mindset:** Khi gặp bugs, team không panic mà debug systematically, research solutions, không bỏ cuộc.
4. **Ownership và accountability:** Mỗi người own domain của mình, không blame nhau khi có bugs mà tập trung fix.
5. **Flexibility:** Sẵn sàng reassign tasks, pair programming, help nhau khi cần - không cứng nhắc về vấn đề đây là việc của ai.

6. Điểm cần cải thiện

1. **Ước lượng thời gian:** Một số tasks bị ước lượng thấp (rotation: 1 ngày → 2 ngày). Cần cải thiện kỹ năng ước lượng, có thể dùng kỹ thuật Planning Poker.
2. **Chậm trễ tài liệu:** Tài liệu được viết vào cuối dự án, nhiều chi tiết bị quên. Nên viết tài liệu song song với code (ví dụ: function headers ngay khi implement).

3. **Độ sâu code review:** Một số PR reviews chỉ là "LGTM" (Looks Good To Me) mà không có nhận xét chi tiết. Nên có danh sách kiểm tra cho reviewers (tính đúng đắn logic, đặt tên, hiệu năng, các trường hợp biên).

7. Bài học kinh nghiệm cho dự án tương lai

1. **Tích hợp sớm:** Merge feature branches vào develop sớm và thường xuyên hơn (mỗi 1-2 ngày thay vì 3-4 ngày) để phát hiện conflicts sớm.
2. **Đa nền tảng từ ngày đầu:** Thiết lập môi trường phát triển cho cả macOS và Linux từ đầu, không chờ đến giai đoạn tích hợp mới test Linux.
3. **Phân chia task tốt hơn:** Chia tasks thành các subtasks nhỏ hơn (< 1 ngày làm việc), dễ ước lượng và theo dõi tiến độ hơn.
4. **Tiếp cận tài liệu trước:** Viết tài liệu thiết kế cấp cao trước khi code, giúp team thống nhất về kiến trúc và giảm việc làm lại.

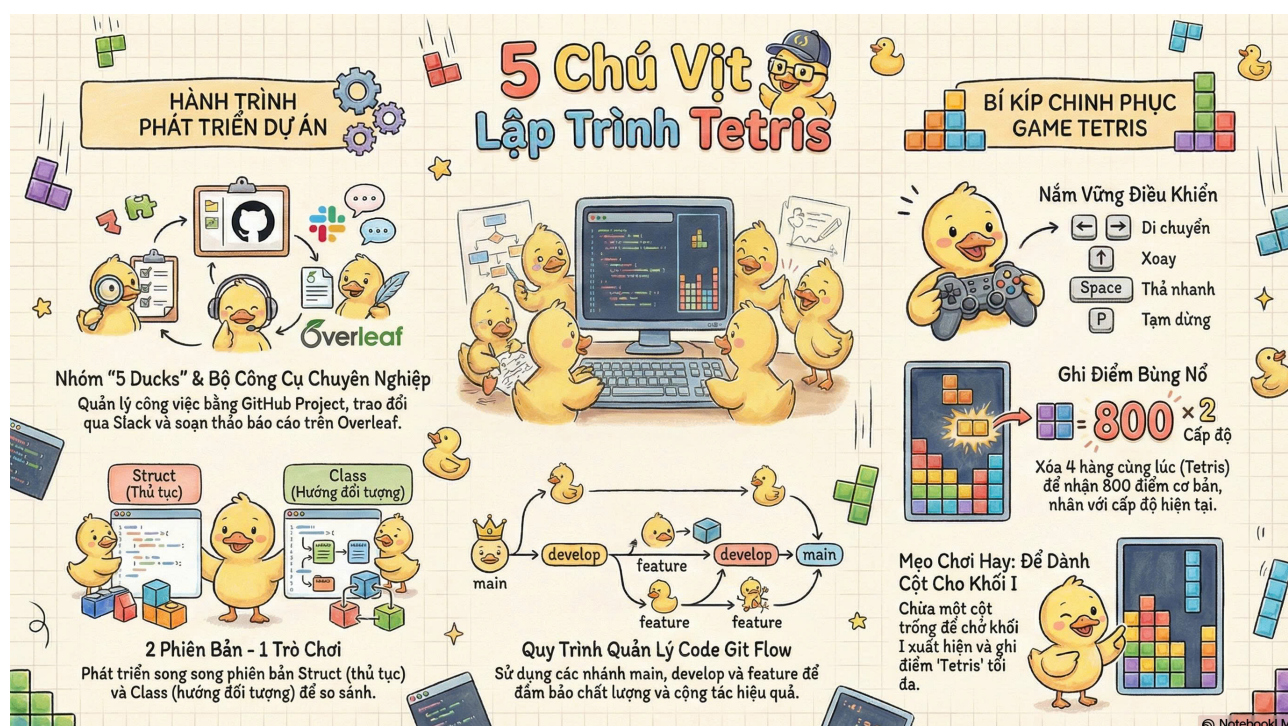
8. Kết luận đánh giá

Nhóm 5 Ducks đã thực hiện **xuất sắc** các cam kết trong hợp đồng nhóm:

- **100% sản phẩm bàn giao** hoàn thành đúng hoặc sớm hơn timeline
- **95%+ tuân thủ** coding conventions và Git workflow
- **6/6 tiêu chí** đánh giá thành viên đạt mức Tốt trở lên, không có ai ở mức Kém/Bình thường
- **Không có xung đột** nào không giải quyết được, sự gắn kết nhóm tốt suốt dự án

- **Chất lượng đầu ra cao:** Game hoạt động ổn định, code dễ bảo trì, tài liệu toàn diện

Dự án Tetris không chỉ là một bài tập cuối kỳ mà còn là trải nghiệm quý giá về làm việc nhóm, quản lý dự án, và software engineering practices. Những kỹ năng và bài học từ đồ án này sẽ là nền tảng vững chắc cho các dự án lớn hơn trong tương lai.



— Đội ngũ phát triển 5 Ducks —

Dương Hoà Long • Lê Quang Nhật • Lê Hữu Nhị

Nguyễn Duy Thanh • Kiều Quang Việt