

ĐẠI HỌC KINH TẾ THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ  
VIỆN ĐỔI MỚI SÁNG TẠO



BÁO CÁO ĐỒ ÁN MÔN HỌC  
ĐỀ TÀI: XÂY DỰNG GAME PLATFORMER 2D  
THỂ LOẠI METROIDVANIA - “WINTER KNIGHT”

Giảng viên hướng dẫn: ThS. Huỳnh Việt Thám

Môn học: Lập Trình Ứng Dụng

Nhóm sinh viên thực hiện: ClassOverFlow

Lớp: 25C1TEC55002503

HỌ VÀ TÊN – MSSV

CAO HÀ NHI - 31241020243

PHAN LÊ HIẾU - 31241020557

ĐINH SỸ HOÀNG QUÂN - 31241024009

TRẦN TRUNG NGHĨA - 31241024610

HUỲNH GIA TÍN - 31241020795

TP. Hồ Chí Minh, ngày 24 tháng 12 năm 2025

## BẢNG PHÂN CÔNG NHIỆM VỤ

HỌ VÀ TÊN	MSSV	NHIỆM VỤ	Đánh giá mức độ hoàn thành (%)
Cao Hà Nhi (NHÓM TRƯỞNG)	31241020243	<ul style="list-style-type: none"> <li>– Quản lý tiến trình làm game và link Github cho nhóm</li> <li>– Thiết kế hình ảnh (BACKGROUNDS)</li> <li>– Thiết kế và code giao diện, chỉnh sửa thuật toán, animation cho KNIGHT</li> <li>– Thiết kế giao diện và chỉnh sửa thuật toán để vật phẩm (ICON) xuất hiện ngẫu nhiên</li> <li>– Viết báo cáo và chỉnh sửa sơ đồ UML</li> </ul>	100%
Huỳnh Gia Tín	31241020795	<ul style="list-style-type: none"> <li>– Thiết kế và code đối tượng trực tiếp tương tác với người chơi (ENEMY THINGS)</li> <li>– Thiết kế FONT chữ cho game</li> </ul>	100%

		<ul style="list-style-type: none"> <li>- Thiết kế giao diện và thuật toán của các màn chơi trong game (LEVELS)</li> <li>- Thiết kế và chỉnh sửa giao diện Menu (STRINGS)</li> <li>- Viết báo cáo và chỉnh sửa sơ đồ UML</li> </ul>	
Đinh Sỹ Hoàng Quân	31241024009	<ul style="list-style-type: none"> <li>- Thiết kế và code đối tượng trực tiếp tương tác với người chơi (ENEMY THINGS &amp; ENEMY SKELETON)</li> <li>- Chỉnh sửa và thiết kế ngôn ngữ (Package LANGUAGE)</li> <li>- Thiết kế TILES cho game</li> <li>- Thiết kế và chỉnh sửa trạng thái hiển thị màn hình (SCREENS)</li> <li>- Viết báo cáo và chỉnh sửa sơ đồ UML</li> </ul>	100%
Trần Trung Nghĩa	31241024610	<ul style="list-style-type: none"> <li>- Thiết kế và code đối tượng trực tiếp tương tác với người chơi (ENEMY SKELETON)</li> </ul>	100%

		<ul style="list-style-type: none"> <li>- Chính sửa và thiết kế hiệu ứng âm thanh cho game (AUDIO)</li> <li>- Thiết kế TILES cho game</li> <li>- Thiết kế và chỉnh sửa trạng thái hiển thị màn hình (SCREENS)</li> <li>- Viết báo cáo và chỉnh sửa sơ đồ UML</li> </ul>	
Phan Lê Hiếu	31241020557	<ul style="list-style-type: none"> <li>- Thiết kế và tìm kiếm tài nguyên cho BOSS</li> <li>- Thiết kế và code đối tượng trực tiếp tương tác với người chơi (ENEMY SKULL)</li> <li>- Thiết kế và chỉnh sửa giao diện trò chơi (GUI)</li> <li>- Thiết kế giao diện và thuật toán của các màn chơi trong game (LEVELS)</li> <li>- Viết báo cáo và chỉnh sửa sơ đồ UML</li> </ul>	100%

# Mục Lục

<b>CHƯƠNG 1: GIỚI THIỆU .....</b>	<b>11</b>
1.1. Lý do chọn đề tài .....	11
1.2. Giới thiệu kỹ thuật lập trình hướng đối tượng .....	12
1.3. Mục tiêu của dự án.....	13
1.4. Phạm vi dự án .....	14
<b>CHƯƠNG 2: PHÂN TÍCH HỆ THỐNG .....</b>	<b>17</b>
2.1. Biểu đồ Use Case.....	17
2.2. Đặc tả Use Case .....	17
2.2.1. Danh sách Actor.....	17
2.2.2. Danh sách Use Case .....	18
2.2.3. Use case Language.....	18
2.2.4. Use Case New Game.....	19
2.2.5. Use Case Load Game .....	20
2.2.6. Use case Credits .....	21
2.2.7. Use case Options .....	21
2.2.8. Use case Exit.....	22
<b>CHƯƠNG 3: THIẾT KẾ HỆ THỐNG .....</b>	<b>23</b>
3.1. Cấu trúc tệp dữ liệu.....	23
3.1.1. Thư mục src: .....	23
3.1.2. Thư mục resources .....	24
3.1.3. Các thư mục tài liệu .....	24
3.2. Hệ thống các packages.....	24
3.3. Thiết kế chi tiết lớp (UML) .....	28
3.3.1. Phía Resources:.....	28
3.3.1.1 Audio: .....	28
3.3.1.2. Sprites: .....	29
3.3.1.3. Language: .....	35
3.3.1.4. Fonts: .....	39

3.3.1.5. Levels: .....	39
3.3.2 Phía src:.....	45
<b>CHƯƠNG 4: HIỆN THỰC ỨNG DỤNG .....</b>	<b>66</b>
4.1. Công nghệ sử dụng: .....	66
4.2. Kết quả của chương trình minh họa:.....	66
4.3. Tính năng tương tác trò chơi: .....	67
4.4. Giao diện chương trình: .....	70
4.5. Nhân vật chính và hình nền: .....	76
4.5.1. Nhân vật chính .....	76
4.5.2. Hình nền .....	80
4.6. Kẻ địch trong trò chơi: .....	85
4.6.1 Kiến trúc hệ thống và Mô hình hướng đối tượng .....	85
4.6.2. Cơ chế vận hành vật lý.....	85
4.6.3. Phân loại và Đặc tả hành vi Kẻ địch .....	86
4.6.4. Thiết kế Trùm cuối.....	88
4.7. Kiểm thử các chức năng thực hiện .....	89
<b>CHƯƠNG 5: BÁO CÁO QUÁ TRÌNH SỬ DỤNG AI .....</b>	<b>91</b>
<b>CHƯƠNG 6: THẢO LUẬN VÀ ĐÁNH GIÁ .....</b>	<b>92</b>
6.1. Kết quả đạt được .....	92
6.2. Tồn tại và hạn chế .....	93
6.3. Hướng phát triển của dự án .....	94
6.4. Đánh giá tổng thể tính chất OOP .....	95
6.5 Kết luận.....	96
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>97</b>
<b>PHỤ LỤC.....</b>	<b>98</b>

# DANH MỤC HÌNH ẢNH

Hình 1: Biểu đồ Use Case.....	17
Hình 2: Cấu trúc tệp dữ liệu .....	23
Hình 4: Cursor.png .....	30
Hình 5: death.png .....	30
Hình 6: death.blue.png.....	30
Hình 7: gui.png .....	31
Hình 9: Icons.png .....	32
Hình 10: player. png .....	32
Hình 11: Skeleton.png .....	33
Hình 12: Skull.png.....	33
Hình 13: thing.png.....	34
Hình 14: Boss.png .....	34
Hình 15: tiles.png .....	35
Hình 16: Tutorial.png .....	39
Hình 17: Save01.png .....	40
Hình 19: Level01.png .....	40
Hình 20: Level02.png .....	41
Hình 21: Level03.png .....	41
Hình 22: Level04.png .....	42
Hình 23: Level05.png .....	42
Hình 24: Level06.png .....	43
Hình 25: Level07.png .....	43
Hình 26: Level08.png .....	44
Hình 27: Level09.png .....	44
Hình 28: Boss01.png .....	45
Hình 29: Package Generics .....	46
Hình 30: Package gui.....	47
Hình 31. Package manager .....	48
Hình 32: Package resources.....	50
Hình 33: Package saves .....	53
Hình 34: Package scenes .....	54
Hình 35: Package screens .....	58
Hình 36: Package strings .....	61
Hình 37: File Game.java.....	63
Hình 38: File Main.java.....	65
Hình 38: Vật phẩm 1 .....	67
Hình 39: Vật phẩm 2 .....	67

Hình 40: Tính năng save game .....	68
Hình 41: Bị quái đánh.....	69
Hình 42: Bị nhận sát thương độc từ quái.....	69
Hình 43: Đánh boss và nhận vật phẩm double jump.....	70
Hình 44: Main menu .....	71
Hình 45: Cốt truyện game .....	72
Hình 46: Load game mà không có save data.....	72
Hình 47: Credits game .....	73
Hình 48: Options game.....	73
Hình 49: Thoát game .....	74
Hình 50: Pause game khi đang chơi .....	74
Hình 51: Nhân vật được buff sau khi nhặt vật phẩm.....	75
Hình 52: Nhân vật thấp máu.....	75
Hình 53: Game over sau khi máu về 0 .....	76
Hình 56: Nhân vật đi bộ .....	78
Hình 57: Nhân vật nhảy.....	78
Hình 58: Nhân vật tấn công (1) .....	78
Hình 59: Nhân vật tấn công (2) .....	78
Hình 60: Nhân vật đứng im .....	79
Hình 61: Nhân vật phòng thủ .....	79
Hình 62: Nhân vật mất máu.....	79
Hình 63: Nhân vật chết.....	79
Hình 65: Background mùa đông (1).....	80
Hình 68: Vật phẩm trong game .....	83
Hình 69: Logo game .....	84
Hình 70: Golem pack.....	86
Hình 71: Shuriken.....	87
Hình 73: Thiết kế boss .....	88
Hình 74: Boss triệu hồi quái .....	89

# **DANH MỤC BẢNG BIỂU**

Bảng 1: USE CASE ACTOR .....	18
Bảng 2: DANH SÁCH USE CASE.....	18
Bảng 3: USE CASE LANGUAGE.....	18
Bảng 4: USE CASE NEW GAME.....	19
Bảng 5: USE CASE LOAD GAME.....	20
Bảng 6: USE CASE CREDITS .....	21
Bảng 7: USE CASE OPTIONS .....	21
Bảng 8: USE CASE EXIT .....	22
Bảng 9: KIỂM THỬ CÁC CHỨC NĂNG ĐÃ THỰC HIỆN .....	89
Bảng 10: BÁO CÁO QUÁ TRÌNH SỬ DỤNG AI.....	91
Bảng 11: ĐÁNH GIÁ TỔNG THÊ TÍNH CHẤT OOP .....	95

# LỜI CẢM ƠN

Trước hết, nhóm em xin gửi lời cảm ơn sâu sắc tới giáo viên hướng dẫn - Thạc sĩ Huỳnh Viết Thám đã tạo điều kiện cho chúng em học và áp dụng kiến thức lập trình hướng đối tượng thông qua việc thiết kế trò chơi, qua đó kích thích trí tưởng tượng và tính sáng tạo của sinh viên.

Ý tưởng Game của nhóm em dựa trên trò chơi platformer 2D nổi tiếng “Super Mario”, trong đó người chơi điều khiển nhân vật vượt qua các chướng ngại vật, thu thập vật phẩm và tiêu diệt kẻ thù trong các màn chơi. Trò chơi có tên là “Winter Knight”, được đặt theo bối cảnh mùa đông, phù hợp với thời điểm đồ án được thực hiện khi mùa đông và Giáng sinh đang đến gần. Người chơi sẽ đóng vai 1 sinh viên UEH bước vào hành trình phiêu lưu trong khung cảnh mùa đông, khám phá các màn chơi để tìm kho báu nhằm trả nợ tín và vượt qua những thử thách khác nhau, tiến đến mục tiêu cuối cùng là đánh bại Boss để hoàn thành Game. Trong quá trình chơi, sẽ có những vật phẩm Giáng sinh được đặt ngẫu nhiên trên đường đi để người chơi có thể tăng sức mạnh và tăng cảm giác thú vị cho trò chơi.

Vì thời gian và trình độ hiểu biết về lĩnh vực lập trình Game của nhóm còn giới hạn, nên trong Project không tránh khỏi những thiếu sót. Nhóm em rất mong nhận được những góp ý và đánh giá để trò chơi có thể hoàn thiện hơn trong tương lai. Nhóm em xin chân thành cảm ơn!

Chúc thầy Giáng sinh an lành!

TP. HCM, ngày 24 tháng 12 năm 2025

Người viết lời cảm ơn

**Cao Hà Nhi**

# CHƯƠNG 1: GIỚI THIỆU

## 1.1. Lý do chọn đề tài

Việc nhóm quyết định phát triển dự án "Winter Knight" xuất phát từ sự kết hợp giữa niềm đam mê sáng tạo nội dung và mục tiêu cung cấp tư duy kiến trúc phần mềm chuyên nghiệp. Lựa chọn này được dựa trên ba trụ cột chính:

Thứ nhất, 2D Platformer là "môi trường mẫu" để làm chủ logic Game: Dòng game 2D Platformer không đơn thuần là một thể loại giải trí cổ điển, mà còn được coi là tiêu chuẩn vàng trong đào tạo lập trình game. Thể loại này đòi hỏi việc xử lý cực kỳ chính xác các bài toán vật lý như: trọng lực, gia tốc, lực nhảy và đặc biệt là hệ thống xử lý va chạm giữa nhân vật với địa hình phức tạp. Việc xây dựng một game Platformer từ con số không giúp nhóm hiểu sâu về Game Loop và cách tối ưu hóa hiệu suất xử lý trong thời gian thực.

Thứ hai, "Winter Knight" là vì đây là một dự án cân bằng lý tưởng giữa Thử thách Kỹ thuật và Tính Khả thi, đồng thời là môi trường tuyệt vời để áp dụng triệt để các nguyên tắc của Lập trình Hướng đối tượng (OOP). Game 2D Platformer cung cấp môi trường hoàn hảo để chúng em triển khai kiến trúc game dựa trên đối tượng: mỗi thực thể trong game như nhân vật, kẻ thù, vật phẩm,... được thiết kế như một Class riêng biệt, ứng dụng nguyên lý Đóng gói để quản lý các thuộc tính (HP, ATK) và hành vi (Di chuyển, Tấn công) một cách có tổ chức. Hơn nữa, chúng em có thể minh họa rõ ràng nguyên lý Kế thừa và Đa hình thông qua việc tạo ra một Class cơ sở chung cho các thực thể trong trò chơi, từ đó phát triển các đối tượng cụ thể như Player và Enemy, giúp mã nguồn dễ quản lý và mở rộng hơn.

Thứ ba, việc lựa chọn chủ đề Mùa đông và Giáng sinh không chỉ đơn thuần là một quyết định về mặt sở thích thẩm mỹ, mà còn là một chiến lược quan trọng trong việc xây dựng ngôn ngữ thiết kế và quản lý phạm vi dự án. Bối cảnh tuyết trắng và không gian lạnh giá đặc trưng của mùa đông cung cấp một tone màu tập trung, giúp tạo

ra sự nhất quán tối đa về mặt thị giác cho phong cách đồ họa Pixel Art. Sự đồng nhất này đóng vai trò then chốt trong việc xây dựng trải nghiệm người dùng, giúp người chơi dễ dàng phân biệt giữa các lớp môi trường, các nền tảng có thể tương tác và các mối nguy hiểm từ kẻ thù. Việc duy trì một phong cách đồ họa xuyên suốt còn giúp nhóm tránh được sự rời rạc trong thiết kế, một lỗi thường gặp trong các dự án game độc lập. Quan trọng hơn, lựa chọn này thể hiện tư duy quản lý dự án thực tế khi biết cách cân bằng giữa sức sáng tạo và tính khả thi. Đồng thời, cấu trúc mã nguồn dựa trên OOP chặt chẽ kết hợp với một chủ đề rõ ràng sẽ tạo tiền đề vững chắc cho việc bảo trì hoặc phát triển thêm các màn chơi mới trong tương lai mà không làm phá vỡ cấu trúc tổng thể của trò chơi.

## 1.2. Giới thiệu kỹ thuật lập trình hướng đối tượng

Lập trình Hướng đối tượng (Object-Oriented Programming - OOP) là một mô hình lập trình đóng vai trò nền tảng trong phát triển phần mềm hiện đại, đặc biệt là trong lĩnh vực lập trình game. Khác biệt cốt lõi của OOP là việc thay thế tư duy lập trình tuân tự bằng cách tổ chức chương trình xoay quanh các **Đối tượng** (Objects). Các đối tượng này mô phỏng các thực thể trong thế giới thực, nơi mỗi đối tượng là một thể hiện cụ thể của một Lớp (Class), bao gồm cả dữ liệu trạng thái (thuộc tính) và các hành vi có thể thực hiện (phương thức). Mô hình này tập trung vào việc quản lý dữ liệu và hành vi đi kèm, tạo ra một cấu trúc mã nguồn có tính module hóa cao.

OOP được xây dựng dựa trên bốn trụ cột chính, mang lại tính hiệu quả và độ tin cậy vượt trội:

- **Tính Đóng gói (Encapsulation):** Đây là nguyên lý then chốt giúp bảo mật và quản lý dữ liệu. Tính Đóng gói cho phép gom các thuộc tính và phương thức xử lý chúng vào trong cùng một Lớp, đồng thời ẩn đi các chi tiết cài đặt nội bộ phức tạp. Dữ liệu của đối tượng chỉ có thể được truy cập hoặc thay đổi thông qua các phương thức được định nghĩa công khai, đảm bảo tính toàn vẹn và ngăn ngừa sự can thiệp không kiểm soát từ bên ngoài.

- **Tính Kế thừa (Inheritance):** Kế thừa là cơ chế tái sử dụng mã nguồn mạnh mẽ. Nó cho phép định nghĩa một Lớp mới (Lớp con) dựa trên một Lớp đã có (Lớp cha), thừa hưởng mọi thuộc tính và phương thức của Lớp cha. Điều này không chỉ giúp giảm thiểu việc lặp lại mã (Code Duplication) mà còn tạo ra một hệ thống phân cấp logic và dễ mở rộng.
- **Tính Đa hình (Polymorphism):** Mang ý nghĩa "nhiều hình thức," Đa hình cho phép các đối tượng thuộc các Lớp khác nhau có thể phản ứng theo cách riêng biệt với cùng một lời gọi phương thức. Ví dụ, phương thức \$render()\$ được gọi có thể tạo ra hoạt ảnh khác nhau tùy thuộc vào đối tượng đang được gọi (Player, Enemy, hoặc Item), giúp mã nguồn trở nên linh hoạt và dễ dàng mở rộng chức năng mới.
- **Tính Trừu tượng hóa (Abstraction):** Trừu tượng hóa tập trung vào việc hiển thị chỉ những thông tin cốt lõi và cần thiết cho người dùng hoặc các thành phần khác của chương trình, trong khi che giấu các chi tiết phức tạp không liên quan. Điều này giúp đơn giản hóa giao diện lập trình, cho phép nhà phát triển làm việc với các khái niệm cấp cao mà không cần quan tâm đến cách thức chúng được cài đặt chi tiết.

=> Việc áp dụng OOP mang lại nhiều lợi ích thiết thực như: nâng cao khả năng tái sử dụng mã nguồn (Code Reusability), giảm thiểu thời gian bảo trì (Maintenance) nhờ tính đóng gói, và tăng cường khả năng mở rộng (Scalability) của hệ thống. Chính vì những ưu điểm vượt trội này, mô hình OOP đã được nhóm em lựa chọn làm nền tảng kiến trúc chính để phát triển dự án game "Winter Knight", đảm bảo code game được tổ chức logic, dễ quản lý và có tiềm năng phát triển tính năng mới trong tương lai.

### 1.3. Mục tiêu của dự án

Dự án game Winter knight được thực hiện với mục tiêu chính là vận dụng, đặc biệt là các nguyên lý của Lập trình hướng đối tượng (Object-Oriented Programming - OOP). Thông qua việc xây dựng một trò chơi 2D hoàn chỉnh ở mức cơ bản. Thông qua dự án, nhóm mong muốn áp dụng các trụ cột cốt lõi của OOP như đóng gói, kế thừa, đa

hình và trừu tượng hoá vào việc phân tích, thiết kế và cài đặt các thành phần trong game, từ đó giúp mã nguồn được tổ chức khoa học, dễ quản lý và có khả năng mở rộng.

Bên cạnh đó, dự án còn hướng đến việc rèn luyện tư duy phân tích và thiết kế hệ thống phần mềm cho sinh viên, giúp người học hiểu rõ hơn mối liên hệ giữa lý thuyết lập trình và ứng dụng thực tiễn. Việc xây dựng game Winter Knight cũng tạo điều kiện để nhóm nâng cao kỹ năng lập trình thực hành, làm quen với quy trình phát triển một sản phẩm. Đồng thời, dự án góp phần phát triển kỹ năng làm việc nhóm, khả năng phân công nhiệm vụ và phối hợp giữa các thành viên trong điều kiện thời gian giới hạn của một học phần. Kết quả cuối cùng là một sản phẩm game 2D có thể vận hành ổn định, đáp ứng các yêu cầu cơ bản về chức năng gameplay, phục vụ cho mục đích đánh giá và chấm điểm cuối kỳ của học phần Lập trình ứng dụng.

#### 1.4. Phạm vi dự án

– **Phạm vi gameplay:** Game triển khai lối chơi đi cảnh kết hợp khám phá bản đồ (Metroidvania), trong đó người chơi điều khiển nhân vật hiệp sĩ mùa đông (Winter Knight) di chuyển, nhảy, tấn công kẻ địch và vượt qua các chướng ngại vật. Hệ thống mở khóa khu vực mới được thiết kế ở mức đơn giản, phù hợp với quy mô dự án học phần.

– **Phạm vi kỹ thuật lập trình:**

Trong package winterknight.generics, các nguyên lý lập trình hướng đối tượng (OOP) được áp dụng một cách có chọn lọc và bài bản, phản ánh tư duy thiết kế rõ ràng của tác giả.

**Đóng gói (Encapsulation):** là nguyên lý được thực hiện triệt để nhất - tất cả thuộc tính đều được khai báo là private và chỉ được truy cập thông qua các phương thức getter và setter công khai, đảm bảo tính toàn vẹn dữ liệu và kiểm soát nghiêm ngặt trạng thái của đối tượng. Điều này thể hiện rõ qua lớp GameRect, nơi vị trí và kích thước được bảo vệ và chỉ thay đổi thông qua các phương thức được định nghĩa rõ ràng.

**Trùu tượng (Abstraction):** cũng được sử dụng khá hiệu quả thông qua việc mô hình hóa các khái niệm game như hình chữ nhật (GameRect), animation (GameSpriteAnimation) thành các lớp độc lập, ẩn đi các chi tiết phức tạp bên trong và chỉ cung cấp giao diện đơn giản để tương tác. Tuy nhiên, Ké thừa (Inheritance) lại gần như không được sử dụng; thay vào đó, thiết kế nghiêng về Thành phần hơn Ké thừa (Composition Over Inheritance) - một lựa chọn có chủ ý để tăng tính linh hoạt và giảm sự phụ thuộc giữa các lớp. Ví dụ, GameSpriteAnimation không kế thừa mà chứa một đối tượng GameRect bên trong, cho phép thay đổi hoặc mở rộng hành vi mà không cần sửa đổi cấu trúc kế thừa.

**Đa hình (Polymorphism):** việc áp dụng còn khá hạn chế, chủ yếu thể hiện qua việc sử dụng enum (như GameDamageType) để định nghĩa các hành vi khác nhau cho từng loại sát thương. Điều này cho thấy cách tiếp cận đơn giản, phù hợp với quy mô dự án, thay vì xây dựng một hệ thống đa hình phức tạp với nhiều lớp con và phương thức ghi đè. Nguyên lý Trách nhiệm duy nhất (Single Responsibility Principle - SRP) được tuân thủ rất tốt: mỗi lớp đều có một nhiệm vụ rõ ràng và tập trung, từ Camera chỉ quản lý góc nhìn, GameUtil cung cấp tiện ích hỗ trợ, đến GameRect xử lý va chạm và hình học.

**Ké thừa (Inheritance):** được sử dụng rộng rãi trong dự án nhằm tái sử dụng mã nguồn và tổ chức hệ thống theo cấu trúc phân cấp rõ ràng. Các lớp con kế thừa thuộc tính và phương thức từ lớp cha, đồng thời có thể ghi đè hoặc mở rộng hành vi để phù hợp với từng chức năng cụ thể. Cách tiếp cận này giúp hệ thống dễ quản lý, dễ mở rộng và phù hợp với phạm vi của đồ án môn học.

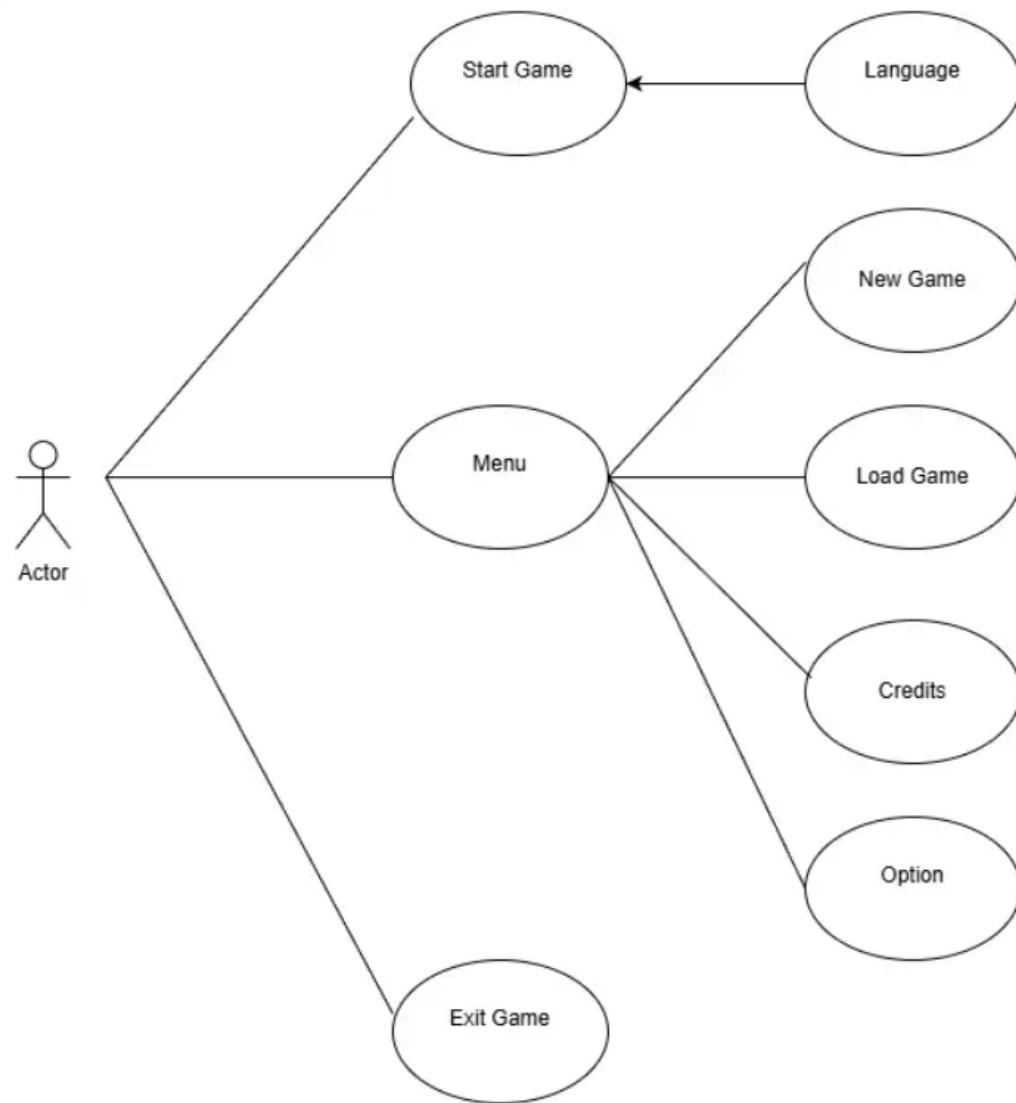
- **Phạm vi đồ họa và âm thanh:** Game sử dụng đồ họa pixel art 2D, lấy cảm hứng từ chủ đề Giáng Sinh và mùa đông, với các yếu tố như tuyết, băng, lâu đài, kẻ địch mang phong cách lễ hội. Âm thanh và nhạc nền được sử dụng ở mức cơ bản nhằm tăng trải nghiệm người chơi, không tập trung vào xử lý âm thanh nâng cao.

- **Phạm vi nội dung:** Dự án giới hạn ở một số màn chơi tiêu biểu, đủ để thể hiện ý tưởng, cơ chế gameplay và khả năng lập trình. Cốt truyện được xây dựng ngắn gọn, mang tính định hướng, không đi sâu vào yếu tố kể chuyện phức tạp.
- **Phạm vi người dùng:** Winter Knight hướng đến người chơi phổ thông, đặc biệt là người chơi yêu thích game pixel và phong cách Metroidvania cổ điển. Game được phát triển cho mục đích học tập, trình diễn và đánh giá học phần, không nhắm mục tiêu thương mại hóa.



# CHƯƠNG 2: PHÂN TÍCH HỆ THỐNG

## 2.1. Biểu đồ Use Case



Hình 1: Biểu đồ Use Case

## 2.2. Đặc tả Use Case

### 2.2.1. Danh sách Actor

Bảng 1: USE CASE ACTOR

TÊN ACTOR	Ý NGHĨA
ACTOR	NGƯỜI CHƠI

### 2.2.2. Danh sách Use Case

Bảng 2: DANH SÁCH USE CASE

Tên Use Case	Ý nghĩa
Language	Cho phép người dùng lựa chọn ngôn ngữ hiển thị trong giao diện và nội dung game (Ví dụ: English, Vietnamese).
New Game	Khởi tạo và bắt đầu một tiến trình chơi mới từ đầu.
Load Game	Tải và tiếp tục tiến trình chơi đã được lưu trữ trước đó.
Credits	Hiển thị danh sách thông tin về người đóng góp cho dự án.
Options	Đưa người dùng các chức năng (Ví dụ: Tắt/Mở nhạc nền, Fullscreen/Window, Tắt/Mở hiển thị FPS).
Exit	Đưa người dùng thoát khỏi ứng dụng.

### 2.2.3. Use case Language

Bảng 3: USE CASE LANGUAGE

STT	Thực hiện	Hành Động
1	System	Hệ thống hiển thị màn hình chọn ngôn ngữ (English, Vietnamese)
2a	User	Người chơi chọn English

2b	System	Người chơi chọn Vietnamese
3a	System	(Nếu chọn English): Hệ thống tải toàn bộ giao diện, văn bản, hướng dẫn và thông báo bằng tiếng Anh
3b	System	(Nếu chọn Vietnamese): Hệ thống tải toàn bộ giao diện, văn bản, hướng dẫn và thông báo bằng Tiếng Việt
4a	System	(Nếu chọn English): Game bắt đầu với ngôn ngữ Tiếng Anh. Kết thúc Use Case
4b	System	(Nếu chọn Vietnamese): Game bắt đầu với ngôn ngữ Tiếng Việt. Kết thúc Use Case

#### 2.2.4. Use Case New Game

Bảng 4: USE CASE NEW GAME

STT	Thực hiện	Hành động
1	User	Người dùng nhấn vào nút New Game.
2a	System	(Nếu đã tồn tại dữ liệu lưu game): Hệ thống kiểm tra và phát hiện có dữ liệu lưu game trước đó.
3a	System	(Nếu đã tồn tại dữ liệu lưu game): Hệ thống hiển thị hộp thoại xác nhận với nội dung: “Start new game? Current progress will be lost.” và hiển thị hai lựa chọn Confirm và Cancel.
4a	User	(Nếu đã tồn tại dữ liệu lưu game): Người dùng chọn Confirm.
5a	System	(Nếu chọn Confirm): Hệ thống xóa hoặc ghi đè dữ liệu lưu game cũ, khởi tạo trạng thái game mới (dữ liệu mặc định) và đưa người chơi vào màn chơi đầu tiên. Kết thúc Use Case.

4b	User	(Nếu đã tồn tại dữ liệu lưu game): Người dùng chọn Cancel.
5b	System	(Nếu chọn Cancel): Hệ thống quay lại Menu chính. Kết thúc Use Case.
2b	System	(Nếu chưa có dữ liệu lưu game): Hệ thống không cần hiển thị xác nhận.
3b	System	(Nếu chưa có dữ liệu lưu game): Hệ thống khởi tạo trạng thái game mới và đưa người chơi vào màn chơi đầu tiên. Kết thúc Use Case.

### 2.2.5. Use Case Load Game

Bảng 5: USE CASE LOAD GAME

STT	Thực hiện	Hành Động
1	User	Người dùng nhấn vào nút Load Game.
2a	System	(Nếu đã có dữ liệu): Hệ thống kiểm tra và phát hiện có dữ liệu lưu game (save data) của người chơi.
3a	System	(Nếu đã có dữ liệu): Tải dữ liệu lưu game gần nhất và đưa người chơi đến checkpoint đã lưu. Kết thúc Use Case.
2b	System	(Nếu chưa có dữ liệu): Hệ thống kiểm tra và phát hiện không có dữ liệu lưu game nào tồn tại.
3b	System	(Nếu chưa có dữ liệu): Hệ thống hiển thị thông báo "No Data" (hoặc "Chưa có dữ liệu lưu") và hiển thị nút "Back".
4b	User	Người dùng nhấn Back
5b	System	Hệ thống quay lại Menu chính. Kết thúc Use Case

## 2.2.6. Use case Credits

Bảng 6: USE CASE CREDITS

STT	Thực hiện	Hành động
1	User	Người dùng nhấn vào nút Credits
2	System	Hệ thống mở bảng credits bao gồm những cá thể đã góp phần làm nên game
3	User	Người dùng nhấn vào nút Back
4	System	Hệ thống đưa về Main Menu

## 2.2.7. Use case Options



Bảng 7: USE CASE OPTIONS

STT	Thực hiện	Hành động
1	User	Người dùng nhấn vào nút Options từ Menu chính.
2	System	Hệ thống hiển thị màn hình Options với các thiết lập hiện tại của game (âm thanh, đồ họa, điều khiển, ...).
3	User	Người dùng thay đổi một hoặc nhiều thiết lập trong Options
4a	User	Người dùng nhấn nút Save để lưu các thay đổi.

5a	System	Hệ thống lưu các thiết lập mới, áp dụng các thay đổi và quay lại Menu chính. Kết thúc Use Case.
4b	User	Người dùng nhấn nút Back hoặc Cancel.
5b	System	Hệ thống không lưu thay đổi và quay lại Menu chính. Kết thúc Use Case.

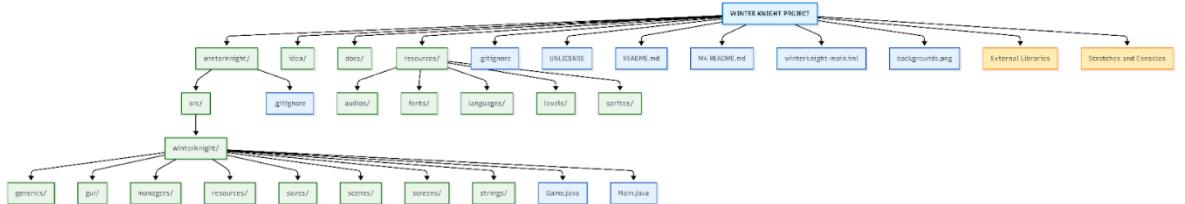
### 2.2.8. Use case Exit

Bảng 8: USE CASE EXIT

STT	Thực hiện	Hành Động
1	User	Người dùng chọn Exit trên màn hình chính của game
2	System	Hệ thống hiển thị hộp thoại xác nhận thoát game.
3a	User	(Nếu người dùng chọn Yes): Người dùng xác nhận thoát game.
4a	System	(Nếu người dùng chọn Yes): Hệ thống lưu các dữ liệu cần thiết (nếu có) và đóng game. Kết thúc Use case
3b	System	(Nếu người dùng chọn No): Người dùng huỷ thao tác thoát game.
4b	System	(Nếu người dùng chọn Yes): Hệ thống quay lại màn hình trước đó (Main Menu). Kết thúc Use Case.

# CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

## 3.1. Cấu trúc tệp dữ liệu



Hình 2: Cấu trúc tệp dữ liệu

Mã nguồn của Winter Knight được sắp xếp theo cấu trúc tách biệt giữa 3 phần gồm: mã nguồn, tài nguyên và tài liệu. Điều này nhằm giúp dễ quản lý và mở rộng trong quá trình phát triển game.

Các tệp dữ liệu chính bao gồm:

- **src/** Chứa toàn bộ mã nguồn Java của game. Mã nguồn được chia thành các package theo chức năng như quản lý giao diện, scene, tài nguyên, lưu game và logic xử lý chính.
- **resources/** Chứa các tài nguyên sử dụng trong game, bao gồm hình ảnh (sprites), âm thanh, font chữ, dữ liệu màn chơi và file ngôn ngữ.
- **docs/** Lưu trữ các tài liệu mô tả và ghi chú liên quan đến dự án.
- **Main.java:** Là điểm khởi động chương trình.
- **Game.java:** Chứa vòng lặp game và điều khiển logic chính.

**3.1.1. Thư mục src:** Chứa toàn bộ mã nguồn chính của game, bao gồm các package và lớp sau:

- **Package generics:** Chứa các lớp dùng chung cho toàn bộ game (lớp tiện ích, hằng số, xử lý cơ bản).
- **Package gui:** Chứa các lớp giao diện người dùng như menu, nút bấm, HUD hiển thị trong game.
- **Package managers:** Chứa các lớp quản lý vòng đồi game và tài nguyên, ví dụ quản lý scene, input, âm thanh.

- **Package resources:** Chứa các lớp dùng để tải và quản lý tài nguyên từ thư mục resources/.
- **Package saves:** Chứa các lớp xử lý lưu và tải trạng thái game.
- **Package scenes:** Chứa các lớp đại diện cho từng scene của game như menu scene, gameplay scene.
- **Package screens:** Chứa các lớp điều khiển các màn hình hiển thị trong game.
- **Package strings:** Chứa các lớp hoặc file quản lý chuỗi văn bản, phục vụ cho đa ngôn ngữ.
- **File Main.java:** Là lớp khởi động chương trình, gọi và khởi tạo game.
- **File Game.java:** Là lớp trung tâm điều khiển vòng lặp game và logic chính.

### 3.1.2. Thư mục resources

Chứa toàn bộ tài nguyên phục vụ cho game, bao gồm:

- **Package sprites/:** chứa các file hình ảnh, sprite nhân vật và vật thể
- **Package audios/:** chứa các file âm thanh, nhạc nền và hiệu ứng
- **Package fonts/:** chứa các file font chữ
- **Package levels/:** chứa các file dữ liệu màn chơi
- **Package languages/:** chứa các file ngôn ngữ

### 3.1.3. Các thư mục tài liệu

- **docs/:** tài liệu dự án
- **README.md:** mô tả tổng quan dự án
- **.gitignore:** cấu hình Git bỏ qua file không cần thiết

## 3.2. Hệ thống các packages

**Đối với resources:**

**Package audios:** Chứa các file âm thanh định dạng .wav phục vụ các sự kiện trong game, bao gồm: attack.wav (âm thanh tấn công),

boss.wav (nhạc nền khi đánh boss), game.wav (nhạc nền khi chơi game), game-over.wav (âm thanh khi game kết thúc), hit.wav (âm thanh trúng đòn chung), hit\_player.wav (âm thanh người chơi bị trúng đòn), jump.wav (âm thanh nhảy), lore.wav (nhạc khi đang phát cốt truyện/lời dẫn), menu.wav (nhạc nền menu), objects.wav (âm thanh tương tác với các đối tượng), và save.wav (âm thanh lưu game).

**Package fonts:** Chứa file font chữ định dạng .ttf (default.ttf), được sử dụng để định dạng và hiển thị văn bản trong giao diện game.

**Package languages:** Chứa các thư mục ngôn ngữ (english, vietnam) và trong đó có các file văn bản dịch thuật như screen.txt (dịch thuật các nội dung giao diện) và level.txt (dịch thuật các nội dung cốt truyện hoặc thông báo trong cấp độ) phục vụ cho tính năng chuyển đổi ngôn ngữ của ứng dụng.

**Package levels:** Chứa các file hình ảnh định dạng .png đóng vai trò là bản đồ hoặc màn hình của các cấp độ game, bao gồm: boss-01.png, level-01.png đến level-09.png, hai màn chơi checkpoint/cấp độ lưu trữ save-01.png, save-02.png, và màn chơi hướng dẫn tutorial.png.

**Package sprites:** Chứa các file hình ảnh định dạng .png đóng vai trò là giao diện của nhân vật hoặc vật phẩm trong game: backgrounds.png, boss.png, cursor.png, death.png, death-blue.png, golem.png, gui.png, icon.png, icons.png, player.png, skeleton.png, skull.png, thing.png, tiles.png, README.txt

### Đối với scr:

**Package generics:** Chứa các lớp tiện ích và cấu hình chung được sử dụng trong toàn bộ game, bao gồm Camera (quản lý khung nhìn), GameColors (định nghĩa bảng màu), GameDamage (tính toán sát thương), GameRect (Lớp hình chữ nhật nguyên dùng để xử lý va chạm và kiểm tra click trong game), GameRectEntity (Lớp hình chữ nhật thực lưu vị trí chính xác của các entity động trong game), GameSpriteAnimation (xử lý

hoạt ảnh), GameStatus (Enum trạng thái game), và GameUtil (tiện ích cung cấp phương thức tạo số ngẫu nhiên).

**Package gui:** Chứa các lớp liên quan đến giao diện người dùng của game, bao gồm lớp giao diện events/EventOnClick (xử lý sự kiện click), GameButton (tạo và quản lý nút bấm trong game với hiệu ứng nhấp/nhả và xử lý sự kiện click), GameText (hiển thị văn bản tĩnh), và GameTextRender (hiển thị văn bản, thường dùng cho hộp thoại hoặc chú thích trong game).

**Package managers:** Chứa các lớp quản lý logic chính của game, bao gồm thư mục con entities chứa thư mục enemies (có thư mục bosses) gồm GameManagerSpriteBoss (quản lý tất cả sprite và animation của boss), cùng với (các lớp quản lý sprite cho kẻ thù) như GameManagerSpriteSkeleton, GameManagerSpriteSkull, GameManagerSpriteThing, GameManagerSpritePlayer (quản lý toàn bộ sprite và animation của nhân vật chính, hỗ trợ nhiều trạng thái khác nhau), GameManagerAudio (quản lý tập trung tất cả âm thanh của game, từ nhạc nền đến hiệu ứng hành động), và GameManagerSpriteDeath (quản lý hoạt ảnh, hiệu ứng khi thực thể bị tiêu diệt).

**Package resources:** Chứa các lớp tiện ích để truy cập và xử lý tài nguyên tĩnh của game, bao gồm GameAudio (truy cập file âm thanh), GameFont (truy cập file font chữ), Spritesheet (quản lý bảng spritesheet), và Translation (xử lý nội dung đa ngôn ngữ, tải và thay đổi bản dịch từ file text).

**Package saves:** Chứa các lớp liên quan đến chức năng lưu trữ và quản lý tiến trình chơi, bao gồm GameSave (định nghĩa cấu trúc dữ liệu cần lưu, lưu trữ dữ liệu của game như máu, sát thương, trạng thái, vật phẩm...) và GameSaveManager (quản lý việc lưu và tải game, xử lý file save và khôi phục trạng thái nhân vật).

**Package scenes:** Chứa các lớp cốt lõi để xây dựng môi trường và các thực thể tương tác trong game, bao gồm thư mục backgrounds nhằm

quản lý các backgrounds với Background, BackgroundCastle, BackgroundMoon, và BackgroundSpriteManagerDebut (dùng để debug để kiểm tra việc load file sprite sheet backgrounds và xác thực tọa độ cắt ảnh), thư mục entities chứa các thực thể như Boss (ở thư mục riêng, không thừa hưởng từ Enemy, boss chính của game với các kỹ năng đặc biệt: dịch chuyển, triệu hồi quái vật, và nhiều phase chiến đấu), Enemy (Lớp trừu tượng định nghĩa đối tượng quái vật cơ bản với máu, sát thương, AI di chuyển và xử lý va chạm), Skeleton, Skull, Thing (cả 3 đều là các loại quái vật cụ thể kế thừa từ lớp Enemy chung, mỗi loại có đặc điểm sprite và hành vi riêng biệt) trong thư mục enemies, và lớp Player (nhân vật chính của game với đầy đủ hệ thống di chuyển, tấn công, animation và xử lý trạng thái), thư mục levels chứa các định nghĩa cấp độ như Boss01, Level01 đến Level09, Save, SaveLeft, SaveRight, Tutorial, thư mục objects (chứa các đối tượng tương tác như GameObject, Life, Spawn, Sword, Teleport), và thư mục tiles (chứa các thành phần xây dựng bản đồ như Block, Floor, Tile, Wall), cùng với lớp quản lý tổng thể Scene.

**Package screens:** Chứa các lớp quản lý các màn hình và trạng thái giao diện người dùng khác nhau của game, bao gồm ConfirmMainMenu (xác nhận quay về Menu chính), ConfirmNewGame (xác nhận chơi lại màn mới từ đầu), Credits, Exit, GameOver, Lore, MainMenu, NoData (không phát hiện dữ liệu lưu game của user), OpeningScreen, Options, Pause, Screen (lớp cơ sở cho màn hình), SelectLanguage (chọn ngôn ngữ), và Transition (quản lý hiệu ứng chuyển cảnh).

**Package strings:** Chứa các lớp tiện ích để quản lý và truy cập các chuỗi văn bản cố định trong ứng dụng, bao gồm GameString (tạo ra một lớp bọc wrapper để quản lý và kiểm soát giá trị của một chuỗi ký tự (String) trong game), StringError (các thông báo lỗi), StringGame (lưu trữ chuỗi liên quan đến tên game), StringLevel (các chuỗi liên quan đến cấp độ), và StringScreen (các chuỗi hiển thị trên màn hình như menu, hộp thoại,...).

**Package core:** Chứa các lớp cốt lõi để khởi tạo và quản lý ứng dụng, bao gồm lớp Game (lớp chính điều khiển toàn bộ game loop, cửa sổ, input và quản lý trạng thái game) và lớp Main (lớp khởi động game và xử lý thoát chương trình với thông báo lỗi).

### 3.3. Thiết kế chi tiết lớp (UML)

#### 3.3.1. Phía Resources:

##### 3.3.1.1 Audio:

###### 1. attack.wav

Có chức năng cung cấp hiệu ứng âm thanh cho hành động tấn công trong game. Âm thanh này được phát ra mỗi khi nhân vật thực hiện đòn đánh, nhằm tăng tính phản hồi (feedback) cho người chơi và làm cho hành động tấn công trở nên rõ ràng, sống động hơn.

###### 2. boss.wav

Được sử dụng như một nhạc nền trong các tình huống đối đầu với boss của game. Âm thanh này được phát khi người chơi bước vào khu vực hoặc bắt đầu trận chiến với boss, nhằm tạo không khí căng thẳng và kịch tính hơn so với các phân đoạn chơi thông thường.

###### 3. game.wav

Dùng làm nhạc nền chính trong quá trình chơi game ở các màn hoặc khu vực thông thường, không bao gồm các tình huống đối đầu với boss. Âm thanh này giúp duy trì không khí chung của trò chơi, tạo cảm giác liền mạch và tránh sự đơn điệu khi người chơi di chuyển hoặc tương tác trong game.

###### 4. game-over.wav

File game-over.wav được sử dụng làm hiệu ứng âm thanh khi người chơi thua cuộc. Khác với các âm thanh mang tính nghiêm túc, file này mang phong cách hài hước, mang tính meme, nhằm giảm cảm giác thất vọng khi game kết thúc.

###### 5. hit\_player.wav

Hiệu ứng âm thanh khi nhân vật người chơi bị trúng đòn tấn công. Âm thanh này được phát ngay tại thời điểm va chạm, giúp người chơi nhận biết rõ ràng rằng nhân vật đã chịu sát thương.

#### 6. jump.wav

Hiệu ứng âm thanh cho hành động nhảy của nhân vật. Âm thanh này được phát mỗi khi nhân vật thực hiện thao tác nhảy

#### 7. lore.wav

File lore.wav được sử dụng làm nhạc nền trong game, âm thanh này giúp tạo không khí chung cho trò chơi, làm nền cho các khoảnh khắc chuyển cảnh.

#### 8. menu.wav

Nhạc nền khi người chơi mở menu của game. Âm thanh này là một bản nhạc theo xu hướng (hot trend) mùa Giáng Sinh, mang giai điệu nhẹ nhàng, lãng mạn.

### 3.3.1.2. Sprites:

#### 1. Backgrounds



Hình 3: Backgrounds.png

Nhóm sử dụng công cụ [Leshylabs](#) để trích xuất tọa độ và nhóm tất cả backgrounds vào một spritemap cố định. Điều này giúp Winter Knight có một hệ thống backgrounds vô cùng đa dạng và đẹp mắt. Đồng thời, việc nhóm tất cả background vào một spritemap cố định cũng giúp nhóm dễ dàng trong việc lập trình để đổi backgrounds một cách đa dạng nhất có thể giữa các màn chơi.

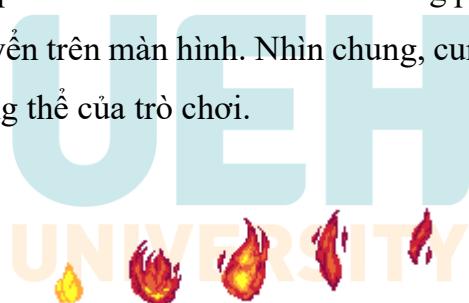
## 2. Cursor



Hình 4: Cursor.png

Cursor được thiết kế đơn giản với hình dạng rõ ràng, giúp người chơi dễ nhận biết vị trí trỏ chuột trong quá trình thao tác. Màu sắc tương phản tốt với background nên không bị chìm khi di chuyển trên màn hình. Nhìn chung, cursor gọn nhẹ, dễ sử dụng và phù hợp với giao diện tổng thể của trò chơi.

## 3. death.png



Hình 5: death.png

**Hiệu ứng khi kẻ địch bị tiêu diệt** được thiết kế với các frame lửa nhỏ tan dần, tạo cảm giác kẻ địch biến mất một cách rõ ràng và dứt khoát. Màu sắc nổi bật giúp người chơi dễ nhận biết thời điểm kẻ địch bị đánh bại. Hiệu ứng đơn giản nhưng hiệu quả, góp phần tăng cảm giác đã tay và phản hồi trực quan trong gameplay.

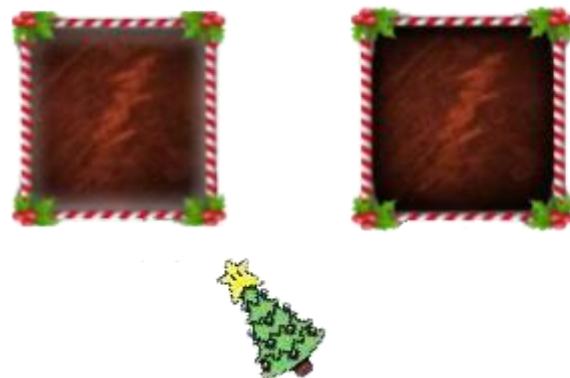
## 4. death.blue.png



Hình 6: death.blue.png

**Hiệu ứng khi người chơi mất hết máu và bị tiêu diệt** được thể hiện bằng các frame hiệu ứng màu xanh lam tan dần, tạo cảm giác lạnh và phù hợp với bối cảnh mùa đông của trò chơi. Chuyển động hiệu ứng diễn ra theo trình tự rõ ràng, giúp người chơi dễ nhận biết trạng thái thất bại. Hiệu ứng đơn giản nhưng đủ nổi bật, góp phần tạo phản hồi trực quan và tăng cảm xúc khi kết thúc lượt chơi.

### 5. gui.png



Hình 7: gui.png

**GUI** được thiết kế theo chủ đề Giáng sinh với viền kẹo gậy đỏ trắng và họa tiết lá xanh, tạo cảm giác vui mắt và đúng không khí lễ hội. Giao diện khá đơn giản, các ô hiển thị rõ ràng nên người chơi dễ nhìn và dễ sử dụng trong lúc chơi. Nhìn chung, GUI phù hợp với phong cách đồ họa pixel art của game và góp phần làm trò chơi sinh động hơn.

### 6. icon.png



Hình 8: Icon.png

**Logo game** được thiết kế một cách sinh động nhằm kỷ niệm môn học “Lập Trình Ứng Dụng” của nhóm với thầy Huỳnh Việt Thám. Ngoài ra, Logo game còn được thiết kế theo chủ đề Giáng sinh với chữ *Winter Knight* nổi bật, sử dụng tông đỏ và trắng giúp dễ nhận diện không khí Noel. Hình ảnh mũ Noel và các chi tiết trang trí xung quanh làm logo trông vui mắt, thân thiện và tạo cảm giác gần gũi với người chơi.

## 7. Icons.png



Hình 9: Icons.png

**Hệ thống vật phẩm** trong trò chơi được thiết kế theo chủ đề Giáng sinh với nhiều biểu tượng quen thuộc như mũ Noel, kẹo gậy, cây thông, bánh ngọt và người tuyết. Các vật phẩm có màu sắc tươi sáng, hình dáng rõ ràng nên người chơi dễ nhận biết trong quá trình chơi. Phong cách pixel art của vật phẩm phù hợp với tổng thể đồ họa của game, góp phần làm gameplay sinh động hơn và tăng không khí Noel.

## 8. player.png



Hình 10: player. png

**Hệ thống animation** của Winter Knight được xây dựng khá dày đú với nhiều trạng thái khác nhau như đứng yên, di chuyển, nhảy, tấn công và các hành động chiến đấu khác. Các frame được vẽ rõ ràng, chuyển động liền mạch nên khi đưa vào game sẽ tạo cảm giác nhân vật chuyển động tự nhiên. Việc chia animation theo từng hàng giúp dễ quản lý và thuận tiện khi xử lý logic nhân vật trong quá trình lập trình. Nhìn chung, hệ thống animation đáp ứng tốt yêu cầu gameplay hành động và phù hợp với phong cách pixel art của trò chơi.

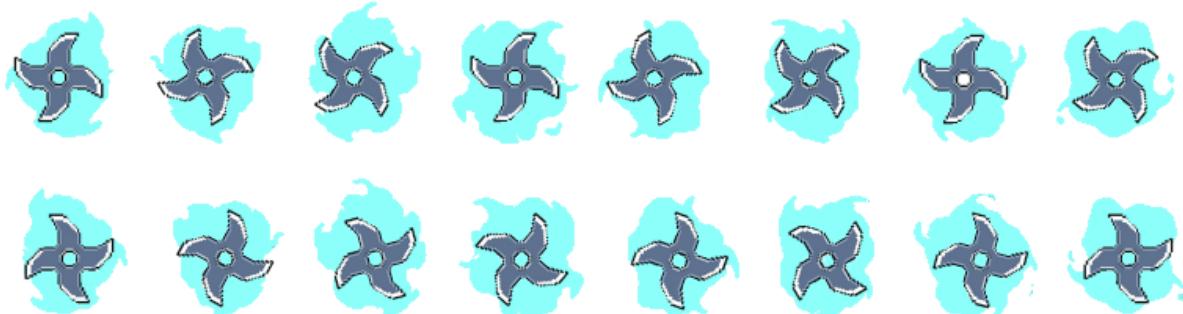
### 9. Skeleton.png



Hình 11: Skeleton.png

**Enemy** được thiết kế theo phong cách pixel art với tông màu xanh lạnh, phù hợp với bối cảnh mùa đông của trò chơi. Hình dạng nhân vật nhỏ gọn nhưng có chi tiết rõ ràng, giúp người chơi dễ nhận biết khi xuất hiện trên màn hình. Hệ thống sprite bao gồm nhiều frame cho các trạng thái khác nhau như đứng yên, di chuyển, bị đánh trúng và biến mất, tạo cảm giác chuyển động mượt và sinh động. Tổng thể thiết kế đơn giản nhưng hiệu quả, phù hợp để sử dụng làm kẻ địch thường trong gameplay.

### 10. Skull



Hình 12: Skull.png

**Sprite hiệu ứng** được thiết kế theo phong cách pixel art với tông màu xanh lam sáng, tạo cảm giác lạnh và huyền ảo, phù hợp với bối cảnh mùa đông của trò chơi. Hiệu ứng xoay tròn qua nhiều frame giúp chuyển động trông mượt mà và dễ nhận biết khi xuất hiện trong gameplay. Hình dạng đơn giản nhưng nổi bật, có thể dùng làm đạn kỹ năng, hiệu ứng phép thuật hoặc đòn tấn công tầm xa. Nhìn chung, sprite hiệu ứng rõ ràng, không quá rối mắt và hỗ trợ tốt cho việc truyền tải hành động trong game.

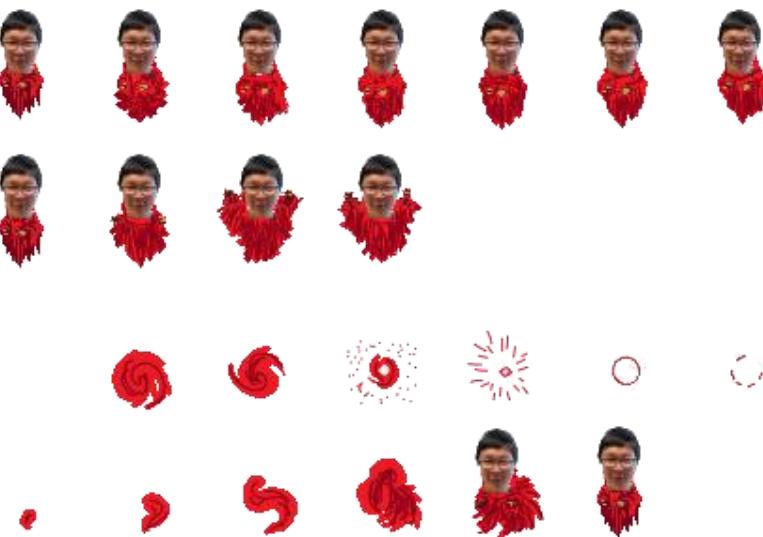
### 11. thing.png



Hình 13: thing.png

**Nhân vật kẻ địch** được thiết kế màu đỏ nổi bật, giúp dễ nhận biết trên nền background. Sprite có nhiều frame thể hiện các trạng thái di chuyển và hoạt động, tạo cảm giác sinh động khi xuất hiện trong gameplay. Thiết kế đơn giản nhưng rõ ràng, phù hợp để làm kẻ địch thường, tạo áp lực liên tục cho người chơi.

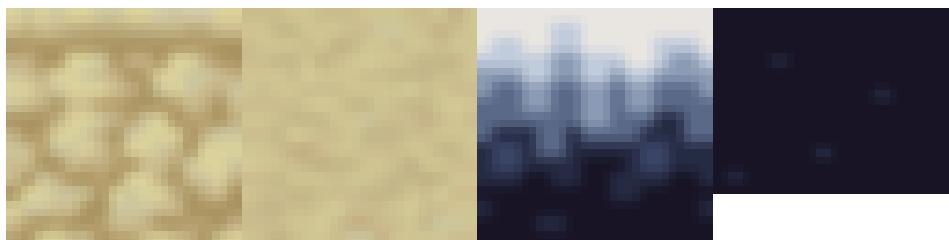
### 12. Boss.png



Hình 14: Boss.png

**Boss** được thiết kế với tạo hình nổi bật, sử dụng tông màu đỏ làm chủ đạo, tạo cảm giác nguy hiểm và khác biệt so với kẻ địch thường. Sprite boss có nhiều frame thể hiện các trạng thái như xuất hiện, tấn công, sử dụng kỹ năng và biến mất, giúp chuyển động trở nên sinh động và rõ ràng trong quá trình chiến đấu. Các hiệu ứng xung quanh boss được thiết kế đồng bộ với màu sắc và hình dạng, làm nổi bật sức mạnh và vai trò đặc biệt của boss trong game. Nhìn chung, boss tạo được cảm giác thử thách và là điểm nhấn quan trọng trong gameplay.

13. tiles.png



Hình 15: tiles.png

**Block băng** được thiết kế với tông màu trắng - xanh nhạt, tạo cảm giác lạnh và đúng với bối cảnh mùa đông của trò chơi. Bề mặt block có hiệu ứng mờ nhẹ, giúp phân biệt rõ với nhân vật và kẻ địch. Thiết kế đơn giản nhưng dễ nhận biết, phù hợp để sử dụng làm địa hình di chuyển và tạo thử thách cho người chơi trong gameplay.

### 3.3.1.3. Language:

#### - ENGLISH

+level:

Press F2 to enable/disable full screen

Press F3 to show/hide FPS

Press F4 to enable/disable music

Press P or ESC to pause the game

Press the A and D keys or the left and right arrow keys to move

Press Z, J or SPACEBAR to jump

Press X, K key, right click or left click to attack

Collect swords to increase your attack  
Collect the red weaklings to increase your maximum health  
Press enter to save the game  
Game saved  
Life restored  
Negative effects removed  
Double jump learned  
Press the jump button 2 times  
Thanks for playing!  
The game will soon have more levels

+screen:

Press F2 to enable/disable full screen

New Game

Load Game

Credits

Options

Exit



Do you want to leave the game?

Yes

No

No Data

Back

Do you want to start a new Game?

Your current progress will be lost

Go to menu?

Continue

Menu

Press enter to continue

Peace reigned under protection of the great empire

Humanity thrived  
But every empire must fall  
On a beautiful night, the Bloody Moon appeared  
Bringing with it the creatures of the night  
Despite the grandeur of the empire  
It succumbed within a few weeks  
It seemed like the end of humanity  
But then emerged the Order of Dyrvanias  
A group of skilled warriors  
After a long battle  
The Bloody Moon was sealed  
But that was long ago  
Today, for many, it's just a legend  
But recently...  
The Moon has been turning increasingly red  
And strange events have begun to occur

### - VIETNAMESE

+level:

Nhấn F2 để bật/tắt chế độ toàn màn hình  
Nhấn F3 để hiển thị/ẩn FPS  
Nhấn F4 để bật/tắt nhạc  
Nhấn P hoặc ESC để tạm dừng trò chơi  
Nhấn phím A và D hoặc mũi tên trái và phải để di chuyển  
Nhấn Z, J hoặc SPACEBAR để nhảy  
Nhấn X, phím K, chuột phải hoặc chuột trái để tấn công  
Thu thập kiếm để tăng sức tấn công  
Thu thập những kẻ yếu màu đỏ để tăng máu tối đa  
Nhấn Enter để lưu trò chơi

Trò chơi đã được lưu  
Máu đã được hồi phục  
Các hiệu ứng xáu đã được loại bỏ  
Đã học nhảy đôi  
Nhấn nút nhảy 2 lần  
Cảm ơn bạn đã chơi!  
Trò chơi sẽ sớm có thêm nhiều cấp độ

+Screen:

nhấn F2 để bật/tắt chế độ toàn màn hình  
trò chơi mới  
tải trò chơi  
danh đề  
tùy chọn  
thoát  
bạn có muốn rời khỏi trò chơi không?  
có  
không  
không có dữ liệu  
quay lại  
bạn có muốn bắt đầu trò chơi mới không?  
tiến trình hiện tại của bạn sẽ bị mất  
quay về menu?  
tiếp tục  
menu  
nhấn Enter để tiếp tục  
hòa bình tồn tại dưới sự bảo hộ của chế vĩ đại  
nhân loại phát triển thịnh vượng  
nhưng mọi chế rồi cũng sẽ sụp đổ  
vào một đêm đẹp trời, Huyết Nguyệt xuất hiện



mang theo nó những sinh vật của màn đêm  
bất chấp sự hùng mạnh của đế chế  
nó đã sụp đổ chỉ trong vài tuần  
tưởng chừng như đó là dấu chấm hết cho loài người  
nhưng rồi, Trật Tự Dyrvania xuất hiện  
một nhóm những chiến binh tài năng  
sau một cuộc chiến dài  
Huyết Nguyệt đã bị phong ấn  
nhưng đó là chuyện từ lâu  
ngày nay, với nhiều người, nó chỉ là truyền thuyết  
nhưng gần đây...  
mặt trăng đang dần chuyển sang màu đỏ  
và những sự kiện kỳ lạ bắt đầu xuất hiện

#### 3.3.1.4. Fonts:

Font chữ: Science Gothic Regular

#### 3.3.1.5. Levels:



Hình 16: Tutorial.png

Đây là màn chơi hướng dẫn (Tutorial) với thiết kế pixel tối giản, sử dụng nền đen và các đường kẻ tím để định hình địa hình. Level này giúp người chơi làm quen với cơ chế di chuyển cơ bản và tương tác với các vật phẩm (như kiếm hoặc mạng) trước khi bước vào hành trình chính thức.

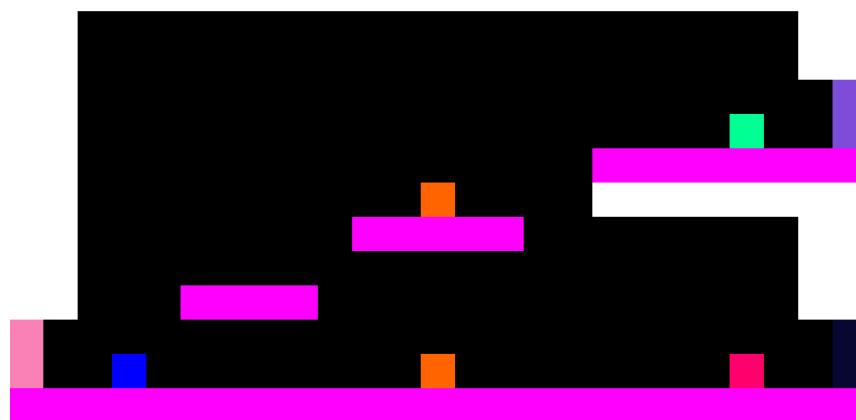


Hình 17: Save01.png



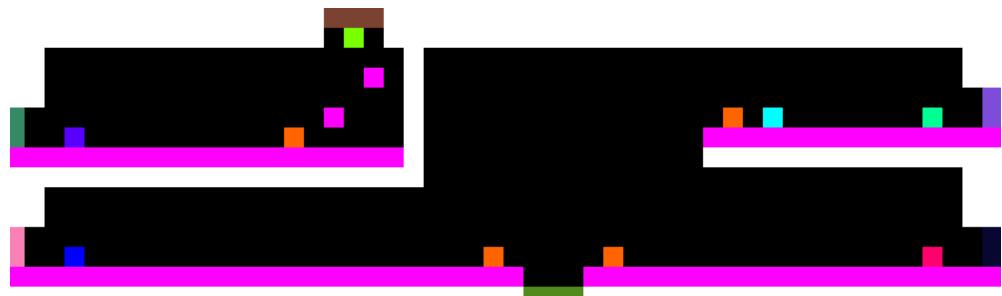
Hình 18: Save02.png

Các phòng nhỏ đặc biệt được thiết kế làm điểm dừng chân để lưu tiến trình game (**save game**).



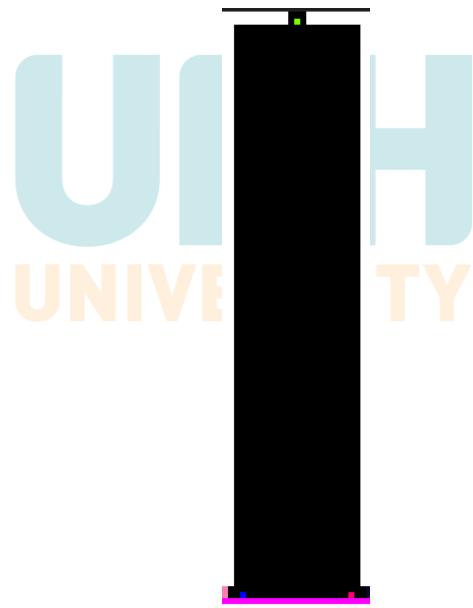
Hình 19: Level01.png

Đây là màn chơi khởi đầu (**Level 01**) của một tựa game platformer phong cách pixel tối giản. Người chơi điều khiển nhân vật vượt qua các bậc thang màu hồng để di chuyển từ trái sang phải và dần tiến lên cao. Mục tiêu chính là làm quen với thao tác nhảy cơ bản để chạm tới điểm đích hoặc thu thập các vật phẩm màu sắc trên màn hình.



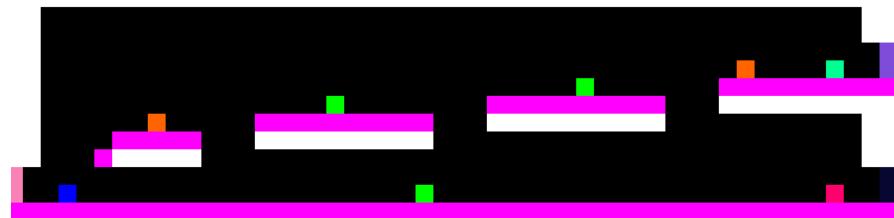
Hình 20: Level02.png

Đây là **Level 02**, mang đến thử thách phức tạp hơn với cấu trúc chia làm hai tầng được kết nối bởi một lối đi dọc. Người chơi phải điều hướng nhân vật di chuyển qua tầng dưới, leo lên tầng trên để thu thập các vật phẩm màu sắc nằm rải rác. Màn chơi này kiểm tra khả năng di chuyển trong không gian hẹp và tìm đường lắt léo hơn so với cấp độ đầu tiên.



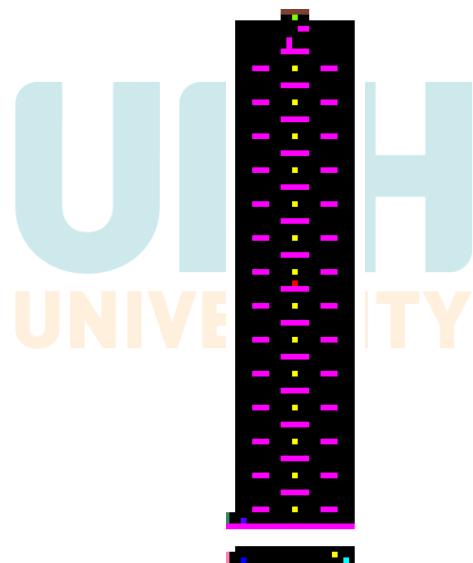
Hình 21: Level03.png

**Level 03** thay đổi hoàn toàn cấu trúc khi giới thiệu một tòa tháp đứng cao vút, buộc người chơi phải chuyển sang kỹ năng nhảy dọc thay vì di chuyển ngang. Bạn cần điều khiển nhân vật từ điểm xuất phát ở đáy tháp để leo dần lên phía trên qua một không gian hẹp. Điểm kết thúc màu xanh lá cây được đặt ở vị trí cao nhất, đòi hỏi sự chính xác tuyệt đối trong việc kiểm soát các cú nhảy.



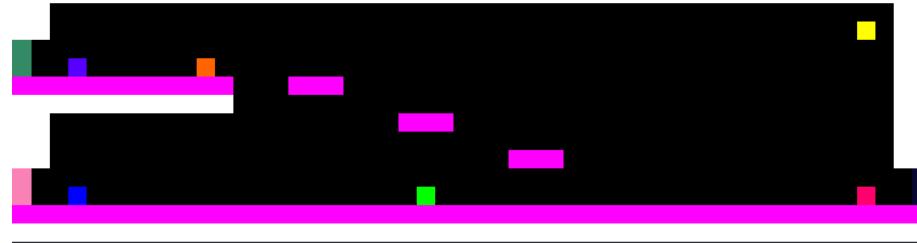
Hình 22: Level04.png

Đây là **Level 04**, quay lại bối cảnh nằm ngang nhưng giới thiệu thử thách mới với các nền nhảy lơ lửng rời rạc. Người chơi phải thực hiện những cú nhảy chính xác qua bốn bậc thang màu hồng-trắng để thu thập các vật phẩm xanh lá nằm ở cả phía trên và phía dưới các nền nhảy. Độ khó được nâng cao khi các khoảng trống xuất hiện nhiều hơn, đòi hỏi khả năng kiểm soát nhân vật tốt hơn để không bị rơi xuống sàn.



Hình 23: Level05.png

Đây là **Level 05**, một thử thách leo tháp đứng có độ phức tạp cao với hệ thống bậc thang dày đặc được bố trí thành hai cột song song. Người chơi phải thực hiện chuỗi nhảy liên tục để thu thập các vật phẩm màu vàng trải dài dọc theo trục giữa của tòa tháp. Màn chơi còn xuất hiện một vật phẩm màu đỏ ở vị trí trung tâm, đóng vai trò như một điểm mốc quan trọng trước khi người chơi chạm đến đích xanh lá ở đỉnh cao nhất.



Hình 24: Level06.png

Đây là **Level 06**, một màn chơi kết hợp giữa di chuyển ngang và nhảy chính xác trên các nền tảng nhỏ. Người chơi bắt đầu từ phía trên bên trái, phải nhảy qua các bậc thang màu hồng lơ lửng được sắp xếp theo hình bậc thang đi xuống để thu thập vật phẩm màu xanh lá ở dưới sàn. Thủ thách nằm ở việc kiểm soát đà nhảy để không bị rơi quá xa, đồng thời phải quay ngược lên hoặc tiến về góc phải để đạt tới điểm đích.



Hình 25: Level07.png

Đây là **Level 07**: Một không gian mở nằm ngang đơn giản, tập trung vào việc thu thập nhanh các vật phẩm đỏ và cam nằm紧跟 trực tiếp trên mặt sàn.



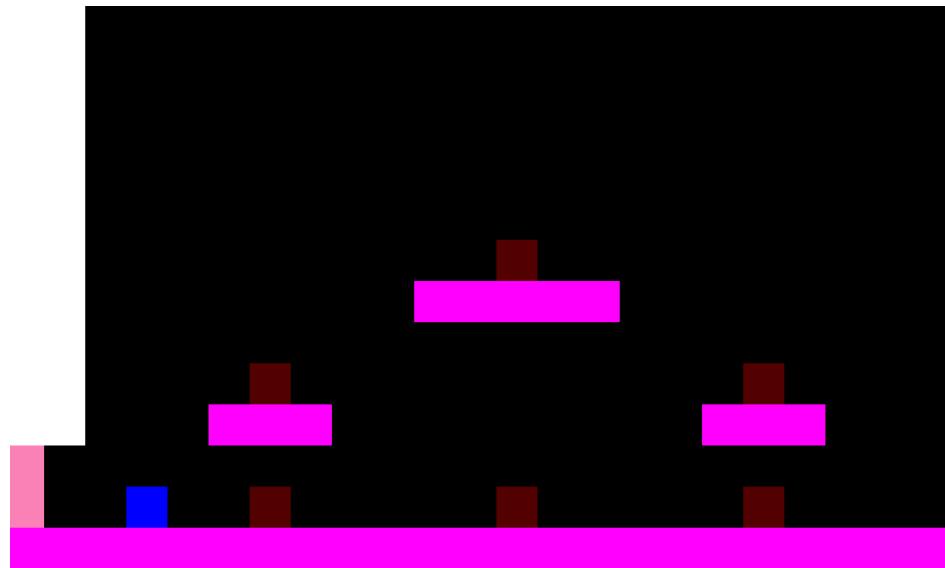
Hình 26: Level08.png

**Level 08:** Một màn chơi có cấu trúc hình chữ L ngược, buộc người chơi phải di chuyển từ lối vào bên phải, băng qua hành lang ngang rồi đi xuống khu vực hép phía dưới. Mục tiêu cuối cùng là thu thập vật phẩm cam và chạm đến điểm đích xanh lá nằm ở góc dưới cùng bên trái.



Hình 27: Level09.png

**Level 09:** Màn chơi này đưa người chơi trở lại một không gian mở tối giản với mặt sàn phẳng màu hồng. Thủ thách tại đây tập trung vào việc di chuyển thuận túy từ điểm xuất phát bên trái (vị trí vật phẩm xanh dương) để tiến về phía bên phải của màn hình.



Hình 28: Boss01.png

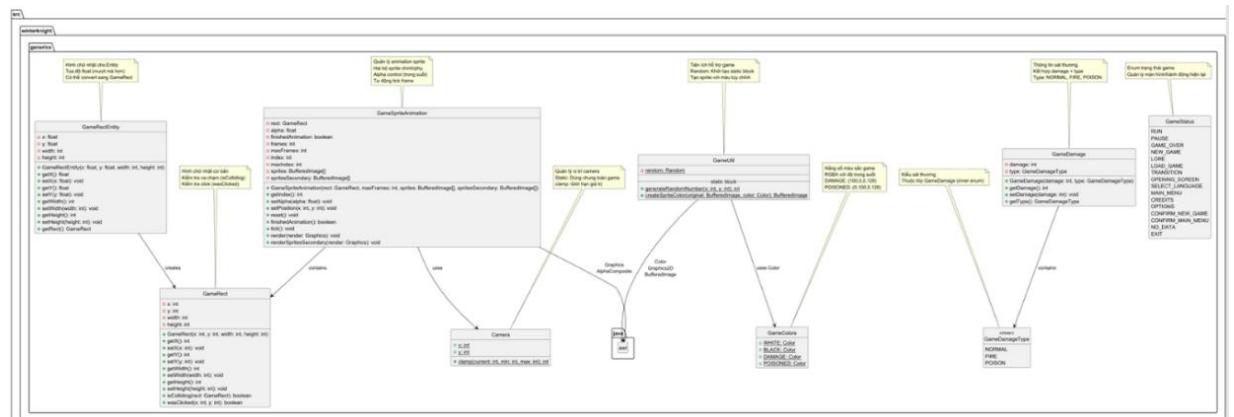
Đây là màn chơi **Boss 01**, đánh dấu sự xuất hiện của các chướng ngại vật nguy hiểm với mật độ cao. Không gian được chia thành nhiều tầng với ba nền tảng lơ lửng, mỗi nền tảng đều có các gai nhọn (vật phẩm màu đỏ) đặt ở cả phía trên và phía dưới. Người chơi phải cực kỳ khéo léo khi nhảy để vừa thu thập vật phẩm, vừa tránh chạm vào các gai này trong một bối cảnh đối đầu xứng đầy thử thách.

### 3.3.2 Phía src:

*Toàn bộ các sơ đồ UML có thể tìm thấy tại (Nên tải các file PDF trong Drive về trước khi xem để có thể xem sơ đồ ở chất lượng HD và rõ ràng hơn):*

[https://drive.google.com/drive/folders/1INGi6qZO9XA7uLKvA2YTvRzE\\_Fw4VJgR](https://drive.google.com/drive/folders/1INGi6qZO9XA7uLKvA2YTvRzE_Fw4VJgR)

\* Package generics



## Hình 29: Package Generics

- **Camera**: Lớp quản lý vị trí camera trong game, sử dụng các thuộc tính tĩnh x, y để điều khiển vùng hiển thị. Các thuộc tính x, y là static để dùng chung toàn game, phương thức clamp() giới hạn giá trị camera trong phạm vi cho phép. Thiết kế đơn giản nhưng hiệu quả cho game 2D side-scroller.

- **GameColors**: Lớp chứa các hằng số màu sắc được sử dụng trong game như màu mặc định, màu sát thương và trạng thái nhiễm độc. Bảng màu chuẩn hóa cho game. Định nghĩa các màu cơ bản (WHITE, BLACK) và màu đặc biệt cho hiệu ứng (DAMAGE với độ trong suốt 50%, POISONED cho trạng thái nhiễm độc). Giúp đồng bộ giao diện trên toàn game.

- **GameUtil**: Lớp tiện ích hỗ trợ game, cung cấp các hàm sinh số ngẫu nhiên và xử lý sprite với màu sắc tùy chỉnh. Có Random được khởi tạo trong static block, cung cấp phương thức sinh số ngẫu nhiên và tạo sprite với màu tùy chỉnh. Đóng vai trò như "toolbox" cho các xử lý chung.

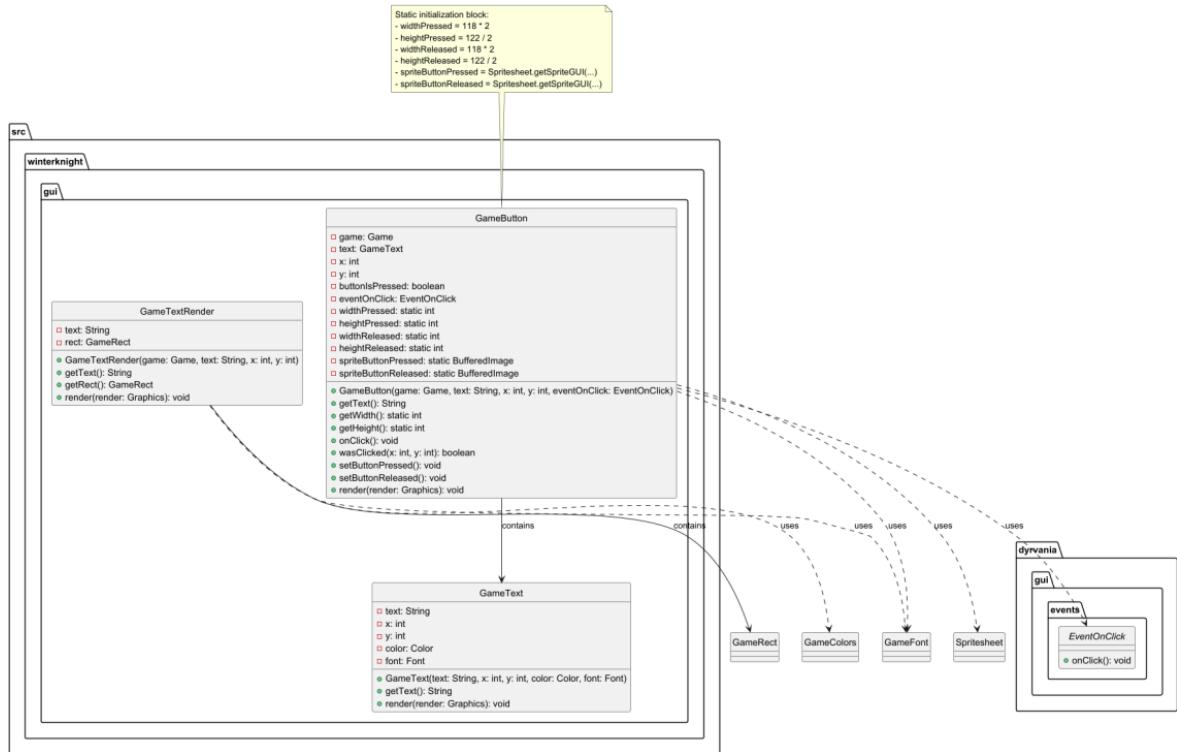
- **GameRect**: Lớp biểu diễn hình chữ nhật cơ bản với các thuộc tính tọa độ và kích thước, dùng để kiểm tra va chạm và click chuột. Lưu tọa độ nguyên (int) và kích thước, cung cấp phương thức isColliding() cho kiểm tra va chạm và wasClicked() cho xử lý sự kiện chuột. Là nền tảng cho physics engine.

- **GameRectEntity**: Lớp hình chữ nhật dành cho entity trong game, sử dụng tọa độ kiểu float để hỗ trợ chuyển động mượt hơn. Hình chữ nhật nâng cao dành cho Entity. Sử dụng tọa độ thực (float) cho chuyển động mượt mà, có thể chuyển đổi sang GameRect thông qua getRect(). Thiết kế phân biệt int/float là hợp lý cho game 2D.

- **GameDamage**: Lớp đại diện cho sát thương trong game, bao gồm giá trị sát thương và loại sát thương tương ứng. Hệ thống sát thương trong game. Kết hợp lượng damage (số nguyên) với loại damage (enum). Hỗ trợ nhiều loại sát thương khác nhau (NORMAL, FIRE, POISON) để mở rộng gameplay.

- **GameSpriteAnimation**: Lớp quản lý animation của sprite, lưu trữ các frame, trạng thái animation và xử lý render hình ảnh. Hệ thống animation cho sprite. Quản lý 2 bộ sprite (chính và phụ), hỗ trợ alpha blending, tự động chuyển frame qua tick(). maxFrames kiểm soát tốc độ animation, finishedAnimation() báo hiệu khi kết thúc.

### \* Package gui



Hình 30: Package gui

- **GameTextRender**: là lớp hỗ trợ hiển thị văn bản trong một vùng xác định của giao diện. Lớp này quản lý nội dung chữ và vùng hiển thị thông qua đối tượng `GameRect`, đồng thời cung cấp phương thức `render(Graphics)` để vẽ văn bản lên màn hình. Việc tách riêng lớp này giúp hệ thống kiểm soát tốt bối cảnh hiển thị và tăng khả năng tái sử dụng.

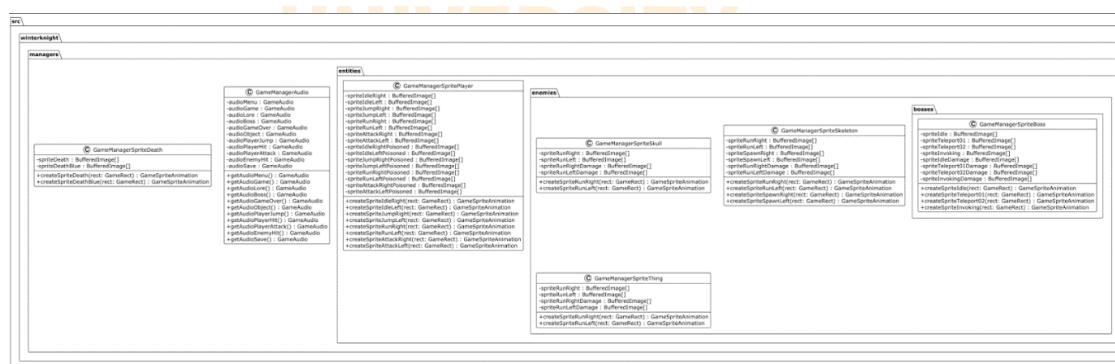
- **GameText**: đại diện cho một đối tượng văn bản trong giao diện trò chơi. Lớp lưu trữ nội dung chữ, vị trí hiển thị, màu sắc và font chữ, đồng thời chịu trách nhiệm trực tiếp cho việc vẽ chữ thông qua phương thức `render(Graphics)`. Lớp này chỉ tập trung vào hiển thị văn bản, không tham gia xử lý tương tác người dùng.

- **GameButton**: là thành phần trung tâm của hệ thống GUI, đại diện cho một nút bấm tương tác. Lớp quản lý vị trí, trạng thái nhấn/thả của nút và chứa một đối tượng GameText để hiển thị nội dung. Các thuộc tính static về kích thước và sprite của button được khởi tạo một lần duy nhất, giúp tối ưu tài nguyên. Việc xử lý click được thực hiện thông qua các phương thức kiểm tra va chạm, trong khi hành vi cụ thể khi click được ủy quyền cho đối tượng sự kiện.

- **Interface EventOnClick:** định nghĩa hành vi khi xảy ra sự kiện click thông qua phương thức onClick(). Việc sử dụng interface giúp tách biệt logic xử lý sự kiện khỏi lớp GameButton, cho phép mỗi button có thể gán các hành vi khác nhau và tăng tính linh hoạt của hệ thống.

- Các lớp như GameRect, GameColors, GameFont và Spritesheet cung cấp các chức năng chuyên biệt liên quan đến hình học, màu sắc, font chữ và tài nguyên đồ họa. Những lớp này được GameButton sử dụng thông qua quan hệ phụ thuộc, góp phần giảm sự phụ thuộc chặt chẽ và nâng cao tính mô-đun của thiết kế.

## \* Package manager



Hình 31. Package manager

- **GameManagerAudio**: là lớp quản lý tập trung tất cả các tài nguyên âm thanh trong trò chơi. Lớp này được thiết kế theo mô hình đơn thể ẩn (static class) để đảm bảo chỉ có một instance duy nhất trong toàn bộ ứng dụng. Nó chứa các biến static đại diện cho từng loại âm thanh cụ thể, bao gồm âm thanh menu, gameplay, cốt truyện, trùm cuối, game over, và các hiệu ứng âm thanh tương tác. Chức năng chính của lớp là cung

cấp các phương thức getter static như getAudioMenu(), getAudioGame(), getAudioPlayerJump() để các thành phần khác trong game có thể truy cập và sử dụng âm thanh một cách thống nhất mà không cần tải lại tài nguyên nhiều lần.

- **GameManagerSpriteDeath:** lớp này chuyên quản lý các hiệu ứng hình ảnh liên quan đến cái chết của các thực thể trong game. Nó lưu trữ hai bộ sprite animation chính: spriteDeath cho hiệu ứng chết thông thường và spriteDeathBlue cho hiệu ứng chết đặc biệt (thường dùng cho boss hoặc enemy đặc biệt). Lớp cung cấp các phương thức factory createSpriteDeath() và createSpriteDeathBlue() để tạo ra các đối tượng GameSpriteAnimation sẵn sàng sử dụng, đảm bảo tính nhất quán và tối ưu hiệu năng khi cần hiển thị hiệu ứng chết nhiều lần.

\* **Thư mục entities:** Thư mục này chứa các lớp quản lý sprite cho các thực thể có khả năng vận động trong game.

- **GameManagerSpritePlayer:** đây là lớp quản lý toàn diện nhất, chịu trách nhiệm lưu trữ và cung cấp tất cả các sprite animation cho nhân vật chính. Lớp này phân loại sprite theo 4 tiêu chí: hướng (trái/phải), trạng thái (đứng yên, chạy, nhảy, tấn công), và tình trạng (bình thường/trúng độc). Tổng cộng có 16 bộ sprite khác nhau, mỗi bộ được lưu trữ dưới dạng mảng Bufferedimage. Các phương thức factory như createSpriteIdleRight(), createSpriteRunLeftPoisoned() cho phép tạo animation phù hợp với trạng thái hiện tại của nhân vật, hỗ trợ hệ thống animation phức tạp với trạng thái trúng độc riêng biệt.

\* **Thư mục enemies:** thư mục con chứa các lớp quản lý sprite cho từng loại kẻ thù cụ thể.

- **GameManagerSpriteSkull:** lớp quản lý sprite cho enemy Skull (đầu lâu bay). Nó lưu trữ 4 bộ sprite: chạy sang phải/trái và chạy sang phải/trái khi bị damage. Điểm đặc biệt là sprite damage được tách riêng, cho phép hiển thị hiệu ứng nhấp nháy hoặc đổi màu khi enemy bị tấn công. Các phương thức createSpriteRunRight() và createSpriteRunLeft() tạo animation tương ứng với hướng di chuyển hiện tại.

- **GameManagerSpriteSkeleton:** lớp quản lý sprite cho enemy Skeleton (bộ xương). Ngoài các bộ sprite chạy và damage tương tự Skull, lớp này còn bổ sung thêm 2 bộ sprite đặc biệt cho quá trình spawn (xuất hiện): spriteSpawnRight và

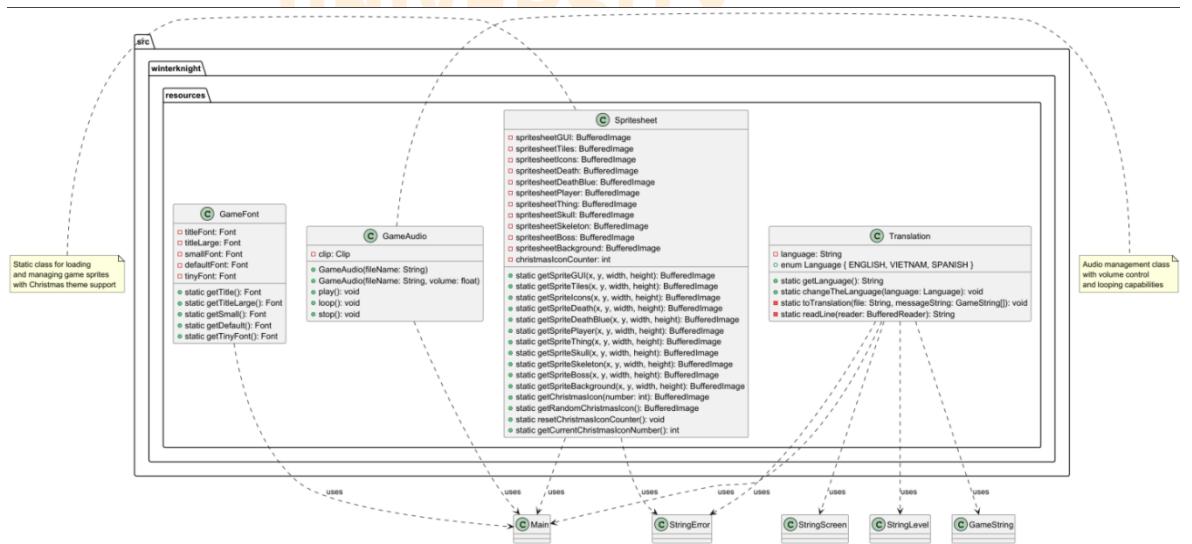
`spriteSpawnLeft`. Điều này cho phép Skeleton có hiệu ứng xuất hiện ấn tượng hơn, phù hợp với đặc tính "hồi sinh" của skeleton trong game. Phương thức `createSpriteSpawnRight()` được gọi khi enemy mới được tạo ra.

- **GameManagerSpriteThing**: lớp quản lý sprite cho enemy Thing (quái vật dị hình). Cấu trúc tương tự Skull với 4 bộ sprite cơ bản. Lớp này có thể được mở rộng thêm các hiệu ứng đặc biệt phù hợp với bản chất "dị hình" của enemy, như biến đổi hình dạng hoặc hiệu ứng độc.

\* **Thư mục bosses:** thư mục chuyên biệt cho quản lý sprite của boss.

- **GameManagerSpriteBoss:** lớp quản lý sprite cho boss cuối với hệ thống animation phức tạp nhất. Nó chứa 8 bộ sprite chia làm 2 nhóm: nhóm bình thường và nhóm damage. Mỗi nhóm bao gồm 4 trạng thái: spriteIdle (đứng yên), spriteTeleport01 và spriteTeleport02 (2 giai đoạn dịch chuyển), spriteInvoking (triệu hồi quái vật). Việc tách riêng sprite damage cho từng trạng thái cho phép boss hiển thị hiệu ứng bị tấn công ngay cả khi đang thực hiện kỹ năng đặc biệt, tăng tính trực quan và độ khó cho người chơi. Các phương thức factory tương ứng cho phép tạo animation phù hợp với hành vi hiện tại của boss.

## \* Package resources



Hình 32: Package resources

- **GameFont**: là lớp chịu trách nhiệm quản lý toàn bộ hệ thống phông chữ được sử dụng trong giao diện trò chơi. Được thiết kế theo mô hình đơn thể ẩn (static class),

lớp sử dụng các biến static để đảm bảo chỉ tồn tại một tập phông chữ duy nhất trong suốt vòng đời chương trình, tránh trùng lặp tài nguyên và tối ưu bộ nhớ. GameFont lưu trữ năm loại phông chữ chính, mỗi loại phục vụ một mục đích hiển thị khác nhau, bao gồm: titleFont cho tiêu đề chính, titleLarge cho tiêu đề lớn thường dùng trong màn hình chính, standardFont cho nội dung văn bản thông thường, defaultFont cho các thành phần UI phổ biến, và tinyFont kích thước nhỏ dùng cho thông kê hoặc thông tin phụ.

- Cơ chế truy cập phông chữ được thực hiện thông qua các phương thức getter static như getTitle(), getTitleLarge(), getDefault(), cho phép các thành phần giao diện truy cập trực tiếp mà không cần khởi tạo instance mới. Thiết kế này đảm bảo tính nhất quán về typography trong toàn bộ giao diện game, giảm chi phí bộ nhớ do không tạo nhiều bản sao Font, đồng thời dễ bảo trì vì việc thay đổi phông chữ chỉ cần thực hiện tại một nơi duy nhất.

- **GameAudio:** là lớp đại diện cho một đơn vị âm thanh có thể phát trong trò chơi, đóng vai trò như một lớp bao (wrapper) cho đối tượng Clip của Java Sound API. Lớp cung cấp một interface đơn giản hóa, giúp việc phát và quản lý âm thanh trở nên trực quan hơn trong game logic. GameAudio hỗ trợ hai constructor: constructor cơ bản nhận tên file âm thanh, dùng cho các âm thanh mặc định, và constructor nâng cao nhận thêm tham số volume để điều chỉnh âm lượng riêng cho từng âm thanh. Thiết kế này giúp linh hoạt trong việc xử lý nhiều loại âm thanh khác nhau như nhạc nền, hiệu ứng va chạm hay các âm thanh đặc biệt.

- Các phương thức chính của GameAudio bao gồm play() phát âm thanh một lần, loop() phát lặp liên tục thường dùng cho nhạc nền, và sing() phát âm thanh theo cơ chế riêng dành cho các hiệu ứng đặc thù. Lưu ý rằng GameAudio không quản lý toàn bộ âm thanh của trò chơi mà được sử dụng bởi lớp GameManagerAudio, chịu trách nhiệm điều phối và quản lý tập trung các âm thanh trong game.

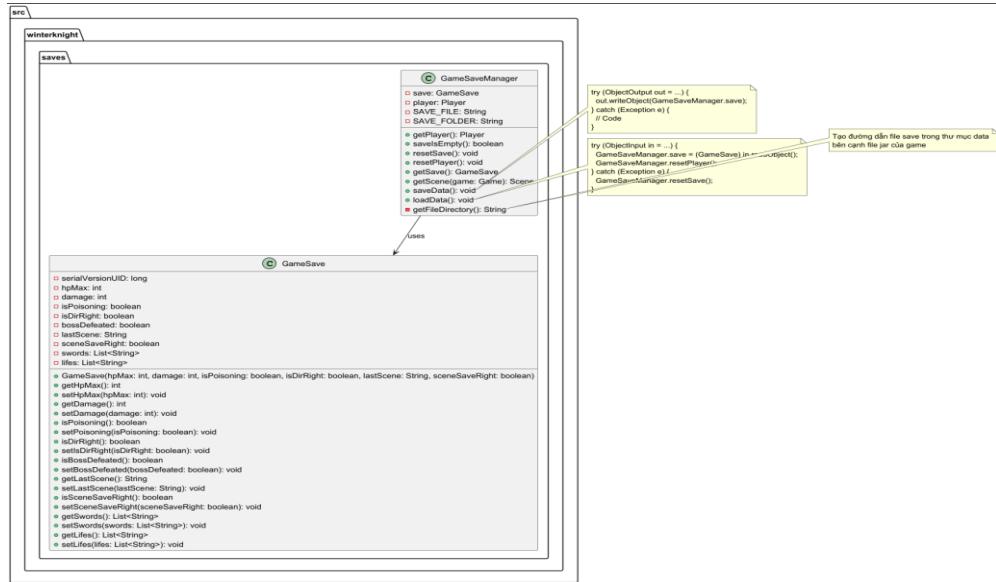
- **SpriteSheet:** là lớp quản lý trung tâm cho toàn bộ sprite sheet hình ảnh trong trò chơi. Lớp được thiết kế theo mô hình đơn thể với các biến static, đảm bảo mỗi bộ sprite sheet chỉ được tải một lần và tái sử dụng suốt quá trình chạy game. SpriteSheet định nghĩa chín biến static, mỗi biến đại diện cho một nhóm sprite sheet chuyên biệt, bao gồm: spriteSheetGUI (giao diện người dùng), spriteSheetTiles (các tile địa hình),

spriteSheetCores (vật phẩm cốt lõi), spriteSheetDashBluer (hiệu ứng dash), spriteSheetPlayer (nhân vật chính), spriteSheetThug (kẻ địch thông thường), spriteSheetStatus (trạng thái nhân vật), spriteSheetSeekIcon (biểu tượng tìm kiếm), spriteSheetClass (phân loại), và spriteSheetBackground (hình nền).

- SpriteSheet cung cấp các phương thức static để trích xuất sprite con từ sprite sheet lớn dựa trên tọa độ và kích thước, giúp tái sử dụng hình ảnh hiệu quả và giảm số lượng file đồ họa cần quản lý. Bên cạnh đó, lớp còn quản lý biến christmasIconCounter để đếm và xoay vòng các biểu tượng Giáng sinh đặc biệt, với các phương thức liên quan như: getRandomChristmasIcon(), resetChristmasIconCounter(), getCurrentChristmasIconNumber(). Cơ chế này cho phép thêm các yếu tố trang trí theo mùa mà không ảnh hưởng đến hệ thống sprite chính.

- **Translation:** là lớp chịu trách nhiệm quản lý hệ thống đa ngôn ngữ (internationalization – i18n) cho trò chơi. Lớp hỗ trợ việc chuyển đổi ngôn ngữ linh hoạt trong thời gian chạy mà không cần khởi động lại game. Lớp định nghĩa enum Language với ba giá trị: ENGLISH, VIETNAM, và SPANISH. Biến static language đại diện cho ngôn ngữ hiện tại của trò chơi. Translation cung cấp các phương thức như getLanguage() để trả về ngôn ngữ hiện tại, changeTheLanguage() để thay đổi ngôn ngữ khi đang chạy game, và toTranslation() chuyển một key thành mảng GameString[] chứa bản dịch cho tất cả ngôn ngữ. Phương thức readLinq() đảm nhiệm việc đọc và phân tích dữ liệu dịch từ nguồn bên ngoài như file cấu hình hoặc dữ liệu lưu trữ, tích hợp chặt chẽ với hệ thống GameString. Hệ thống Translation giúp chuyển đổi ngôn ngữ động, dễ dàng mở rộng thêm ngôn ngữ mới, và nâng cao trải nghiệm người chơi quốc tế.

## \* Package saves



Hình 33: Package saves

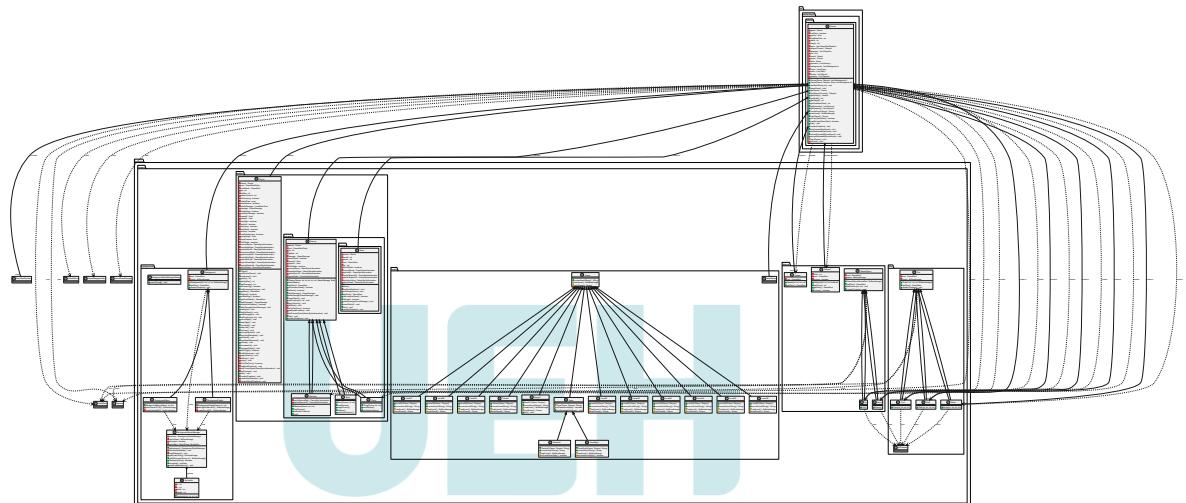
- Gói **saves** là thành phần chịu trách nhiệm quản lý việc lưu trữ và tải dữ liệu trạng thái của trò chơi. Gói này đảm bảo tiến trình của người chơi được bảo toàn qua các phiên chơi khác nhau thông qua cơ chế tuần tự hóa đối tượng. Cấu trúc gói được tổ chức thành các lớp chi tiết như sau:

- **GameSave:** Đây là lớp dữ liệu cốt lõi đóng vai trò lưu trữ toàn bộ thông tin trạng thái của người chơi tại một thời điểm nhất định. Lớp này định nghĩa một loạt các thuộc tính đa dạng như hpMax, damage để xác định chỉ số nhân vật, cùng các biến logic như isPoisoning, bossDefeated để ghi nhận tiến trình cốt truyện. Ngoài các kiểu dữ liệu nguyên thủy, lớp còn sử dụng các cấu trúc danh sách (List<String>) cho swords và lifes để quản lý kho đồ và tài nguyên hệ thống. Chức năng của lớp là cung cấp đầy đủ các phương thức getter và setter, cho phép các thành phần khác trong hệ thống truy xuất và cập nhật dữ liệu một cách an toàn.

- **GameSaveManager:** Lớp này đóng vai trò là bộ điều khiển trung tâm, chịu trách nhiệm thực thi các logic nghiệp vụ liên quan đến tệp tin và dữ liệu. Lớp định nghĩa các hằng số như SAVE\_FILE và SAVE\_FOLDER, kết hợp với phương thức

getFileDirectory() để xác định chính xác vị trí lưu trữ dữ liệu trong thư mục data bên cạnh file thực thi của trò chơi. Các phương thức saveData() và loadData() là hai tiến trình quan trọng nhất. Trong đó, dữ liệu được chuyển đổi giữa đối tượng bộ nhớ và tệp tin vật lý thông qua luồng ObjectOutputStream và ObjectInputStream. Hệ thống tích hợp các khối xử lý ngoại lệ (try-catch). Trong trường hợp xảy ra lỗi khi tải dữ liệu, phương thức resetSave() sẽ được kích hoạt để đưa trò chơi về trạng thái khởi tạo an toàn, tránh gây treo ứng dụng.

#### \* Package scenes



Hình 34: Package scenes

- Gói scenes là thành phần cốt lõi của hệ thống, chịu trách nhiệm quản lý toàn bộ ngữ cảnh hiển thị, vòng đời của trò chơi cũng như sự tương tác giữa các đối tượng. Cấu trúc gói được tổ chức thành các thư mục con và các lớp chi tiết như sau:

- **Backgrounds:** Thư mục này chứa các lớp chịu trách nhiệm quản lý và hiển thị hình ảnh nền cho các màn chơi, đảm bảo hiệu năng và tính thẩm mỹ.

+ **Background:** Đây là lớp trùu tượng đóng vai trò cơ sở cho tất cả các loại hình nền trong trò chơi. Lớp này định nghĩa các thuộc tính cơ bản như rect để xác định vị trí và kích thước, cùng với sprite để lưu trữ hình ảnh hiển thị. Chức năng chính của lớp là cung cấp phương thức render để vẽ hình nền lên màn hình đồ họa tại mỗi khung hình.

+ **BackgroundCastle:** Là một lớp con kế thừa từ lớp Background, lớp này chuyên biệt hóa việc hiển thị hình nền cho các màn chơi có bối cảnh lâu đài. Bên cạnh việc kế thừa các tính năng cơ bản, nó còn tích hợp các phương thức như createErrorBG

để tạo ra hình nền dự phòng trong trường hợp tải tài nguyên thất bại và debugGetMenuBG hỗ trợ quá trình kiểm thử giao diện.

+ **BackgroundMoon**: Tương tự như BackgroundCastle, đây là lớp con kế thừa từ Background nhưng phục vụ cho bối cảnh màn đêm hoặc ánh trăng. Lớp này chứa các phương thức xử lý đặc thù để tải các khung hình cụ thể và để khởi tạo nền đơn giản nhằm tối ưu hóa khi cần thiết.

+ **BackgroundSpriteManager**: Lớp này được thiết kế theo mẫu đơn thể để quản lý tập trung tài nguyên hình ảnh nền. Nhiệm vụ của nó là nạp và lưu trữ các mảnh hình ảnh từ SpriteSheet vào một cấu trúc dữ liệu ánh xạ, giúp tránh việc tải lại cùng một hình ảnh nhiều lần. Các phương thức như getSprite cho phép truy xuất nhanh hình ảnh qua tên định danh, trong khi loadIfNeeded đảm bảo tài nguyên chỉ được nạp khi thực sự cần thiết.

+ **BackgroundSpriteManagerDebug**: Đây là lớp hỗ trợ kỹ thuật, chứa phương thức main độc lập để chạy thử nghiệm và kiểm tra hoạt động của trình quản lý hình ảnh nền BackgroundSpriteManager mà không cần khởi động toàn bộ trò chơi.

- **Entities**: Thư mục này bao gồm các lớp đại diện cho các thực thể sống hoặc có khả năng vận động trong trò chơi, bao gồm nhân vật chính và các loại kẻ thù.

+ **Player**: Đây là lớp quan trọng nhất trong gói, đại diện cho nhân vật chính do người chơi điều khiển. Lớp này quản lý một loạt các chỉ số sinh tồn và trạng thái phức tạp như máu, sát thương, trạng thái trúng độc, và khả năng tạo khiên bảo vệ. Về mặt vật lý, lớp Player xử lý toàn bộ logic di chuyển, nhảy (bao gồm nhảy đôi), và chịu tác động của trọng lực. Ngoài ra, nó còn chịu trách nhiệm xử lý va chạm, tính toán sát thương gây ra hoặc nhận vào thông qua các phương thức dealDamage và takeDamage, đồng thời điều phối các hoạt ảnh tương ứng như chạy, nhảy, hoặc tấn công.

+ **enemies**: Thư mục con chứa các lớp liên quan đến kẻ thù.

+ **bosses**: Thư mục chuyên biệt cho các trùm cuối.

+ **Boss**: Lớp này đại diện cho đối thủ trùm cuối với AI phức tạp. Nó quản lý các trạng thái đặc biệt như dịch chuyển tức thời và triệu hồi quái vật thông qua các hoạt ảnh spriteTeleport và spriteInvoking. Phương thức tick của lớp này chứa logic xử lý hành vi tấn công theo chu kỳ, tạo ra thử thách lớn cho người chơi.

+ **Enemy**: Là lớp trừu tượng định nghĩa các hành vi chung cho mọi kẻ thù trong game. Lớp này thiết lập các thuộc tính cơ bản như máu, vùng va chạm và phương thức di chuyển tự động toMove. Nó cũng cung cấp khả năng kiểm tra va chạm với mặt đất để đảm bảo kẻ thù di chuyển đúng trên địa hình.

+ **Skeleton**: Lớp thực thi cụ thể cho loại quái vật golem. Nó cài đặt phương thức loadSprites để nạp bộ tài nguyên hình ảnh riêng biệt, phù hợp với hành vi và hình dáng của bộ xương.

+ **Skull**: Lớp đại diện cho loại quái vật shuriken bay. Điểm đặc biệt của lớp này là logic xử lý trọng lực và di chuyển applyGravity được tùy biến để phù hợp với đặc tính bay lượn thay vì đi bộ trên mặt đất.

+ **Thing**: Một loại quái vật độc. Lớp này tập trung vào việc xử lý vị trí hiển thị hình ảnh thông qua phương thức setSpritePosition để đảm bảo hình ảnh khớp chính xác với vùng va chạm thực tế.

- **Levels**: Thư mục này quản lý cấu trúc của từng màn chơi cụ thể và logic lưu trữ tiến trình.

+ **Boss01**: Lớp đại diện cho màn chơi đấu trùm. Nó chứa logic cập nhật riêng biệt trong phương thức tick để xử lý các sự kiện đặc thù chỉ xảy ra khi người chơi đối đầu với Boss.

+ **Level01 - Level09**: Các lớp này (từ Level01 đến Level09) kế thừa từ lớp cha Scene, mỗi lớp đại diện cho một màn chơi cụ thể trong chuỗi nhiệm vụ. Nhiệm vụ chính của chúng là hiện thực hóa phương thức loadLevel để nạp bản đồ từ các file hình ảnh tương ứng và định nghĩa phương thức nextScene để chuyển hướng người chơi sang màn kế tiếp sau khi hoàn thành.

+ **Save**: Lớp trừu tượng định nghĩa logic cho các phòng lưu game. Nó cung cấp phương thức kiểm tra isSceneSaveRight để xác định vị trí và hướng của phòng lưu, đảm bảo người chơi được hồi sinh đúng điểm khi tải lại game.

+ **SaveLeft & SaveRight**: Hai lớp này kế thừa từ Save, đại diện cho các phòng lưu game nằm ở phía bên trái hoặc bên phải bản đồ. Chúng thực hiện việc nạp cấu trúc phòng lưu tương ứng và quản lý trạng thái an toàn cho nhân vật.

+ **Tutorial**: Lớp thiết kế riêng cho màn chơi hướng dẫn. Tại đây, cấu trúc bản đồ được tối ưu hóa để người chơi làm quen với các thao tác cơ bản thông qua phương thức nạp level đặc thù.

- **Objects**: Thư mục này chứa các đối tượng vật lý tương tác được, bao gồm vật phẩm hỗ trợ, điểm sinh ra và công dịch chuyển.

+ **GameObject**: Lớp cơ sở trừu tượng cho tất cả các vật thể trong game. Nó chuẩn hóa việc quản lý vị trí thông qua GameRect và hiển thị hình ảnh, cung cấp phương thức render để vẽ vật thể lên màn hình.

+ **Life**: Lớp đại diện cho vật phẩm hồi máu hoặc tăng mạng sống. Khi được khởi tạo, nó sẽ chờ sự tương tác từ người chơi để kích hoạt hiệu ứng hồi phục.

+ **Spawn**: Lớp này đóng vai trò như một điểm đánh dấu vô hình trên bản đồ. Nó xác định tọa độ xuất hiện ban đầu của người chơi hoặc quái vật khi màn chơi bắt đầu.

+ **Sword**: Lớp đại diện cho vật phẩm tăng sát thương cho kiếm. Khi người chơi thu thập đối tượng này, các chỉ số tấn công của nhân vật sẽ được nâng cấp.

+ **Teleport**: Lớp đại diện cho công dịch chuyển không gian. Nó sử dụng mã màu để định danh điểm đến và thuộc tính rect để xác định vùng kích hoạt. Khi người chơi bước vào vùng này, hệ thống sẽ chuyển cảnh sang bản đồ mới.

- **Tiles**: Thư mục này quản lý các khối gạch cấu thành nên kiến trúc bản đồ trò chơi.

+ **Block**: Lớp đại diện cho các khối gạch chướng ngại vật hoặc trang trí. Chúng có thể được đặt lơ lửng hoặc xếp chồng để tạo địa hình phức tạp.

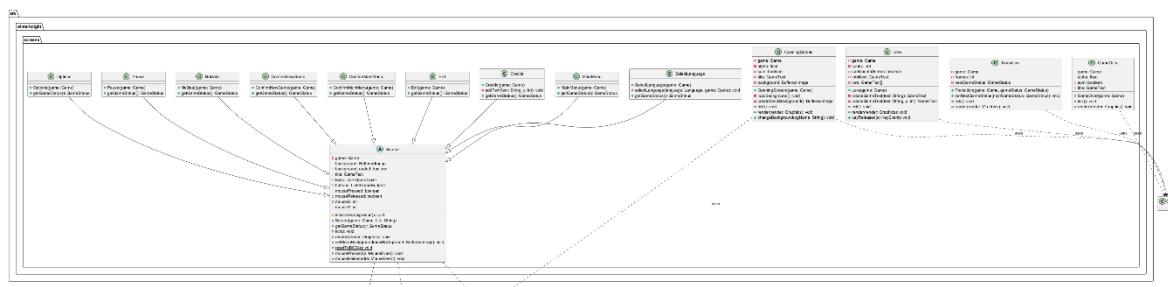
+ **Floor**: Lớp đại diện cho sàn nhà hoặc mặt đất. Chức năng chính của nó là tạo ra bề mặt vật lý để nhân vật và quái vật có thể đứng và di chuyển trên đó mà không bị rơi xuống.

+ **Tile**: Lớp cha của tất cả các loại gạch. Nó chịu trách nhiệm lưu trữ thông tin về vị trí và hình ảnh của từng ô gạch, đồng thời cung cấp phương thức render để vẽ toàn bộ bản đồ.

+ **Wall**: Lớp đại diện cho các bức tường thẳng đứng. Chức năng của nó là ngăn cản chuyển động ngang của nhân vật, tạo ra các giới hạn không gian cho màn chơi.

- **Scene:** đây là lớp quan trọng nhất đóng vai trò "Bộ điều khiển trung tâm" (Controller) cho mỗi màn chơi. Lớp này sở hữu và quản lý danh sách toàn bộ các đối tượng đang tồn tại như kẻ thù, vật phẩm, gạch, và nhân vật chính. Về mặt vận hành, nó cung cấp phương thức tick để cập nhật logic của toàn bộ thế giới game theo thời gian thực và phương thức render để vẽ tất cả các thành phần lên màn hình theo thứ tự lớp. Ngoài ra, lớp Scene còn cung cấp các tiện ích vật lý quan trọng như kiểm tra va chạm isFree để xác định xem một vị trí có bị chặn bởi tường hay không và phương thức gameSave để ghi lại tiến trình chơi.

#### \* Package screens



Hình 35: Package screens

- Đây là lớp trừu tượng đóng vai trò khuôn mẫu cho tất cả các màn hình trong trò chơi. Lớp này định nghĩa các thuộc tính cơ bản như game để tham chiếu đến đối tượng game chính, background và backgroundLoaded để quản lý hình nền, title để hiển thị tiêu đề màn hình, cùng với hai danh sách texts và buttons để chứa các thành phần giao diện. Lớp cũng quản lý trạng thái chuột thông qua các biến mousePressed, mouseReleased, mouseX, mouseY để xử lý sự kiện click. Chức năng chính của lớp bao gồm phương thức tick() để cập nhật logic, render() để vẽ giao diện, và các phương thức xử lý sự kiện chuột mousePressed() và mouseReleased(). Đặc biệt, lớp cung cấp phương thức trừu tượng getGameStatus() mà mọi lớp con phải triển khai để xác định trạng thái tương ứng. Các phương thức static initializeBackground(), setMenuBackground(), resetToBG04() cho phép quản lý hình nền toàn cục một cách linh hoạt.

- **OpeningScreen:** lớp này đại diện cho màn hình khởi động đầu tiên khi game được bật. Ngoài việc kế thừa từ Screen, nó bổ sung thêm các thuộc tính đặc biệt như

alpha để điều khiển độ trong suốt (hiệu ứng fade in/out), sum để xác định chiều thay đổi alpha (tăng/giảm), và background riêng biệt. Lớp có phương thức loadBackground() để tải hình nền từ BackgroundSpriteManager và createBlackBackground() để tạo nền đen dự phòng. Phương thức changeBackground() cho phép thay đổi hình nền động theo tên, hỗ trợ hiệu ứng chuyển cảnh mượt mà. Phương thức tick() được override để điều khiển hiệu ứng alpha, tạo ra hiệu ứng mờ dần hoặc hiện dần cho màn hình khởi động.

- **GameOver:** lớp đại diện cho màn hình kết thúc game khi người chơi thua. Tương tự OpeningScreen, lớp này sử dụng hệ thống alpha để tạo hiệu ứng visual nhưng với mục đích khác - thông báo kết thúc game một cách ấn tượng. Lớp tập trung vào việc hiển thị thông báo "Game Over" với hiệu ứng fade in, không chứa các nút bấm phức tạp, tạo cảm giác trang nghiêm phù hợp với ngữ cảnh thua cuộc. Phương thức render() được thiết kế đặc biệt để vẽ lớp phủ đen bao trùm suốt lên toàn bộ màn hình trước khi hiển thị thông báo.

- **Lore:** lớp chuyên biệt cho việc hiển thị cốt truyện (lore) của game. Lớp này quản lý một mảng lore chứa các đối tượng GameText đại diện cho từng dòng cốt truyện, cùng với biến control để điều khiển vị trí hiển thị hiện tại và canUpdateScreen để xác định khi nào có thể chuyển sang dòng tiếp theo. Lớp cung cấp hai phiên bản của phương thức createGameText() để tạo văn bản với vị trí Y mặc định hoặc tùy chỉnh. Phương thức keyReleased() được override để cho phép người chơi nhấn phím để chuyển qua các phần của cốt truyện, tạo trải nghiệm đọc tương tác. Phương thức tick() điều khiển tốc độ cuộn văn bản và logic hiển thị tuần tự.

- **Transition:** lớp quản lý hiệu ứng chuyển cảnh giữa các màn hình. Lớp này sử dụng biến frames để đếm số khung hình đã trôi qua và nextGameState để xác định trạng thái game tiếp theo sẽ chuyển đến. Phương thức setNextGameState() cho phép thiết lập đích đến động. Trong phương thức tick(), lớp đếm frames và tự động chuyển sang trạng thái tiếp theo khi đạt ngưỡng thời gian. Phương thức render() tạo hiệu ứng chuyển cảnh trực quan (thường là fade to black hoặc wipe) để che giấu quá trình tải màn hình mới, đảm bảo trải nghiệm mượt mà.

- **ConfirmNewGame:** lớp hiển thị hộp thoại xác nhận khi người chơi muốn bắt đầu game mới. Lớp này đảm bảo người chơi nhận thức được rằng tiến trình hiện tại sẽ

bị mất thông qua thông báo rõ ràng. Giao diện thường bao gồm hai nút "Yes" và "No" với layout đơn giản.

- **NoData**: lớp xử lý trường hợp không có dữ liệu game để tải. Lớp này thông báo cho người chơi rằng không tìm thấy file save và cung cấp các lựa chọn như quay lại menu hoặc tạo game mới.

- **ConfirmMainMenu**: lớp hiển thị hộp thoại xác nhận khi người chơi muốn quay về menu chính từ giữa gameplay. Lớp này ngăn chặn việc thoát nhầm và đảm bảo người chơi có cơ hội lưu tiến trình nếu cần.

- **Pause**: lớp quản lý màn hình tạm dừng game. Khi được kích hoạt, game ngừng cập nhật logic nhưng tiếp tục render hình nền với lớp phủ bán trong suốt. Lớp cung cấp các tùy chọn như tiếp tục, vào options, hoặc về menu chính

- **Options**: lớp quản lý màn hình cài đặt game. Lớp này chứa các controls để điều chỉnh âm lượng, độ sáng, điều khiển, và các tùy chọn hiển thị như fullscreen. Các thay đổi được áp dụng ngay lập tức hoặc sau khi xác nhận.

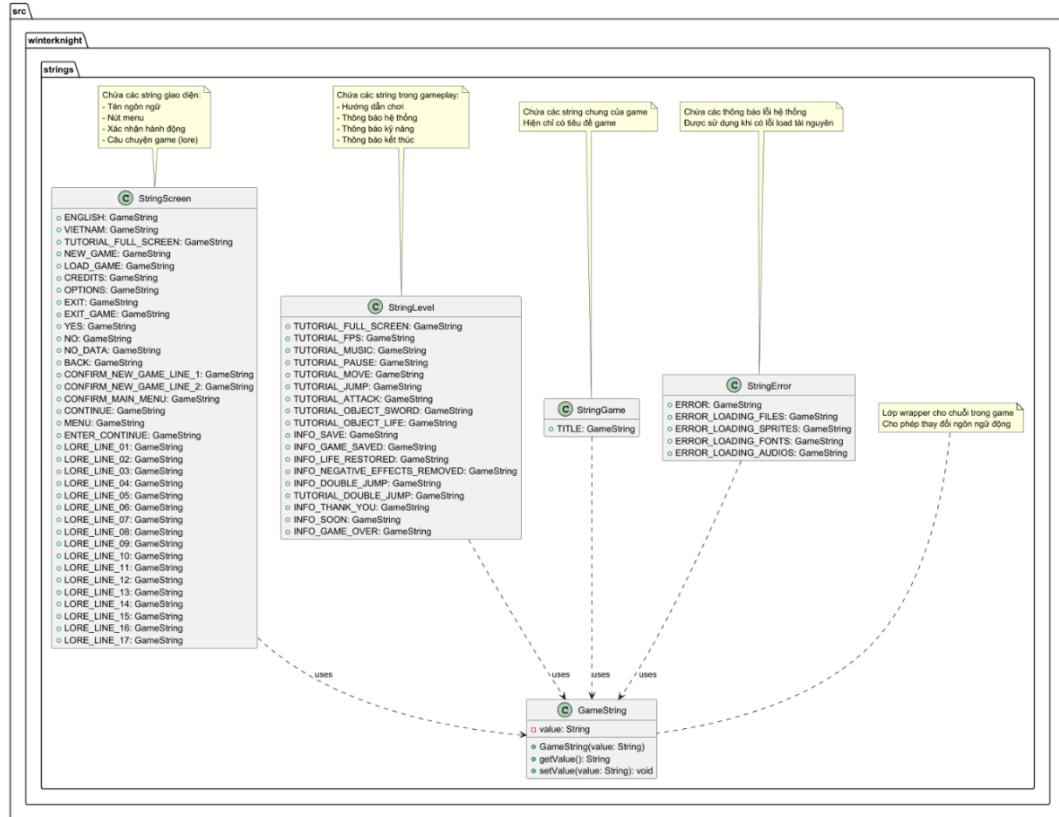
- **SelectLanguage**: lớp chuyên biệt cho việc lựa chọn ngôn ngữ game. Lớp này hiển thị danh sách các ngôn ngữ được hỗ trợ (English, Vietnam, Spanish) dưới dạng nút bấm. Phương thức selectLanguage() được gọi khi người chơi chọn một ngôn ngữ, kích hoạt hệ thống dịch thuật thay đổi toàn bộ giao diện.

- **MainMenu**: lớp đại diện cho menu chính của game - trung tâm điều hướng toàn bộ trò chơi. Lớp này chứa các nút chính: New Game, Load Game, Options, Credits, Exit với layout được thiết kế hấp dẫn trực quan, thường kết hợp với hình nền động hoặc hiệu ứng đặc biệt.

- **Credits**: lớp hiển thị thông tin về đội ngũ phát triển, công cụ sử dụng, và các đóng góp. Lớp sử dụng phương thức addText() để thêm từng dòng credit với vị trí Y được tính toán tự động hoặc chỉ định, tạo danh sách cuộn có thể điều hướng.

- **Exit**: lớp xử lý quá trình thoát game. Lớp này có thể hiển thị hộp thoại xác nhận và thực hiện các thao tác dọn dẹp tài nguyên trước khi đóng ứng dụng hoàn toàn.

## \* Package strings



Hình 36: Package strings

- Gói strings đóng vai trò quản lý toàn bộ hệ thống văn bản và thông báo trong trò chơi. Thay vì sử dụng chuỗi ký tự cứng nhắc được hard-coded, gói này tổ chức văn bản thành các đối tượng có cấu trúc nhằm hỗ trợ tính năng đa ngôn ngữ và giúp việc bảo trì, thay đổi nội dung trở nên dễ dàng và tập trung hơn.

- **GameString:** đây là lớp cơ sở quan trọng nhất trong gói, đóng vai trò là một lớp wrapper cho các chuỗi ký tự nguyên thủy trong Java. Thay vì lưu trữ dữ liệu dưới dạng String thông thường, hệ thống sử dụng đối tượng GameString để đóng gói nội dung văn bản thông qua thuộc tính value. Thiết kế này cho phép thay đổi nội dung văn bản động ngay trong thời gian chạy mà không cần khởi tạo lại đối tượng, là cơ chế cốt lõi để thực hiện tính năng chuyển đổi ngôn ngữ (ví dụ: Tiếng Anh sang Tiếng Việt) một cách tức thời trên toàn bộ giao diện người dùng.

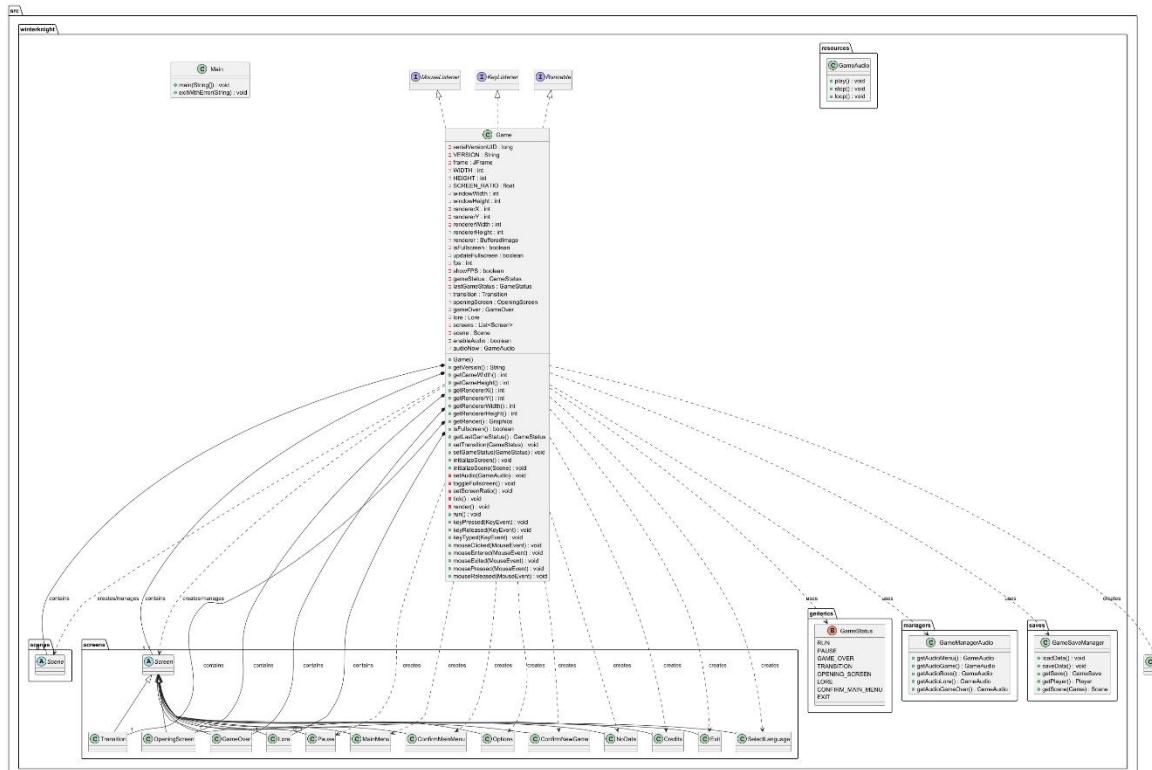
- **StringError:** lớp này đóng vai trò là kho lưu trữ trung tâm cho tất cả các thông báo lỗi kỹ thuật của hệ thống. Nó định nghĩa các biến tĩnh kiểu GameString đại diện cho các trạng thái lỗi phổ biến mà ứng dụng có thể gặp phải trong quá trình vận hành, điển hình là các lỗi liên quan đến việc nạp tài nguyên như không tải được tệp tin cấu hình, thất bại khi đọc dữ liệu hình ảnh (sprites), lỗi phông chữ hoặc lỗi âm thanh. Việc tập trung các chuỗi lỗi vào một lớp giúp dễ dàng quản lý và thống nhất cách thông báo lỗi tới người dùng cuối.

- **StringGame:** lớp này chịu trách nhiệm quản lý các chuỗi văn bản mang tính chất định danh chung và toàn cục của trò chơi. Hiện tại, công năng chính của nó là lưu trữ và cung cấp tên tiêu đề của ứng dụng thông qua biến tĩnh TITLE, được sử dụng để hiển thị trên màn hình chờ ban đầu.

- **StringLevel:** đây là lớp chuyên biệt dùng để quản lý các văn bản xuất hiện trong quá trình chơi, đặc biệt là các thông điệp tương tác trực tiếp với người chơi trong màn hình game. Lớp này cung cấp hệ thống văn bản hướng dẫn (tutorial) chi tiết như cách di chuyển, nhảy, tấn công hay sử dụng vật phẩm. Ngoài ra, nó còn chứa các thông báo trạng thái quan trọng phản hồi lại hành động của người chơi như thông báo đã lưu game thành công, thông báo hồi máu, gỡ bỏ hiệu ứng xấu, mở khóa kỹ năng double jump, hoặc thông báo kết thúc trò chơi (Game Over) và lời cảm ơn khi hoàn thành game.

- **StringScreen:** lớp này là thành phần lớn nhất trong gói, chịu trách nhiệm quản lý toàn bộ văn bản hiển thị trên giao diện người dùng và các yếu tố cốt truyện. Nó định nghĩa các chuỗi ký tự cho hệ thống Menu bao gồm các nút chức năng như New Game, Load Game, Options, Credits, Exit, cũng như các hộp thoại xác nhận quan trọng. Bên cạnh đó, lớp này còn chứa một lượng lớn các dòng văn bản dẫn truyện để xây dựng bối cảnh cốt truyện cho game và các chuỗi định danh ngôn ngữ (English, Vietnamese) để phục vụ cho menu cài đặt chuyển đổi ngôn ngữ.

## \*Package game



Hình 37: File Game.java

- **Main:** lớp khởi động của hệ thống, chứa phương thức `main(String[] args)`. Lớp này chịu trách nhiệm tạo đối tượng game và bắt đầu vòng đời của trò chơi. Vai trò của Main chỉ giới hạn ở việc khởi chạy, không tham gia vào logic xử lý game.

- **Game:** là lớp trung tâm điều phối toàn bộ hệ thống. Lớp này cài đặt các interface như `Runnable`, `KeyListener` và `MouseListener`, cho thấy nó trực tiếp quản lý vòng lặp game, xử lý bàn phím và chuột. Game lưu trữ các thông tin cấu hình quan trọng như kích thước màn hình, FPS, trạng thái fullscreen, cũng như tham chiếu đến `GameState`, `Scene` và các manager. Đây là lớp đóng vai trò “bộ não” của trò chơi.

- **Enum GameState:** `GameState` định nghĩa các trạng thái khác nhau của trò chơi như RUN, PAUSE, GAME\_OVER, TRANSITION, OPENING\_SCREEN, CONFIRM\_MAIN\_MENU và EXIT. Việc sử dụng enum giúp quản lý luồng game một cách rõ ràng, tránh lỗi logic và tăng tính dễ đọc của mã nguồn.

- **Scene:** đại diện cho một màn hình hoặc bối cảnh cụ thể trong trò chơi. Mỗi scene chịu trách nhiệm hiển thị và xử lý logic riêng biệt. Lớp Game quản lý việc chuyển đổi giữa các scene, cho thấy mối quan hệ điều phối thay vì phụ thuộc chặt chẽ.

- **Các lớp Scene cụ thể:** Hệ thống bao gồm nhiều scene con như OpeningScreen, GameOver, Pause, MainMenu, Options, ConfirmNewGame, ConfirmMainMenu, NoLife, Credits, Exit, SetLanguage và Transition. Mỗi lớp đại diện cho một trạng thái giao diện cụ thể của trò chơi, giúp tách biệt rõ ràng logic và giao diện của từng màn hình.

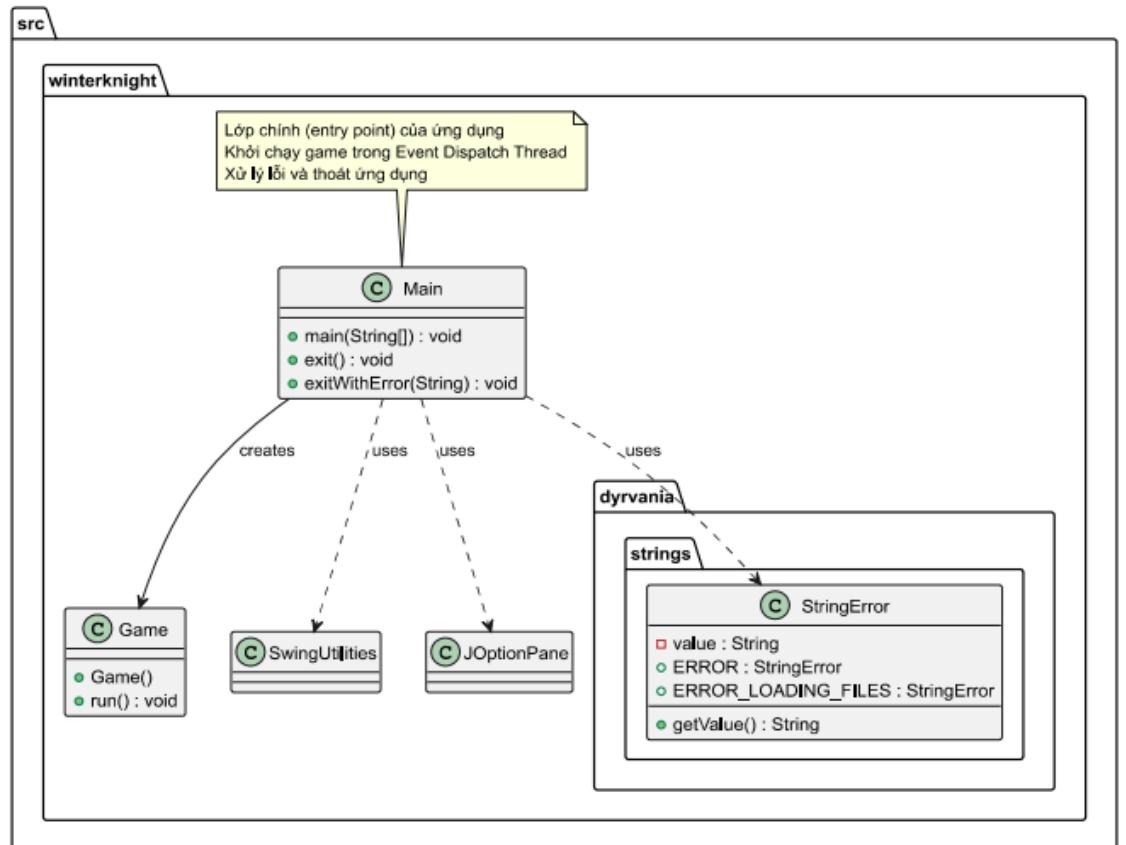
- **Lớp GameStateManager:** chịu trách nhiệm quản lý và cung cấp các trạng thái game tương ứng. Lớp này giúp tập trung hóa việc kiểm soát trạng thái, giảm sự phụ thuộc trực tiếp giữa Game và các scene cụ thể.

- **GameSceneManager:** quản lý việc tạo, lưu trữ và chuyển đổi giữa các Scene. Thiết kế này cho phép hệ thống mở rộng thêm scene mới mà không cần chỉnh sửa nhiều ở lớp Game.

- **Lớp GameAudio:** quản lý âm thanh trong trò chơi với các chức năng như phát, dừng và lặp âm thanh. Việc tách âm thanh thành một lớp riêng giúp hệ thống dễ bảo trì và thay đổi tài nguyên audio mà không ảnh hưởng đến logic game.

- **Các interface xử lý sự kiện:** Việc cài đặt Runnable, KeyListener và MouseListener cho thấy hệ thống hỗ trợ xử lý bất đồng bộ và tương tác người dùng theo thời gian thực. Các interface này giúp chuẩn hóa cách tiếp nhận sự kiện và tích hợp với thư viện Java.

## \* Package main



**Hình 38: File Main.java**

- **Main:** Lớp chính (entry point) của ứng dụng. Phương thức main() là điểm bắt đầu chạy game và sử dụng SwingUtilities.invokeLater() để đảm bảo game được khởi chạy trong Event Dispatch Thread. Lớp Main cung cấp các phương thức exit() và exitWithError() để xử lý việc thoát game và hiển thị thông báo lỗi thông qua JOptionPane.

- **Game:** Lớp đại diện cho game chính, được khởi tạo bởi lớp Main. Phương thức run() chịu trách nhiệm khởi động vòng lặp game và quản lý toàn bộ logic chính của ứng dụng. Đây là trung tâm điều khiển các hoạt động gameplay trong trò chơi.

- **StringError:** Lớp thuộc package winterknight.strings, dùng để quản lý các thông báo lỗi được chuẩn hóa. Cung cấp các hằng số lỗi như ERROR, ERROR\_LOADING\_FILES, giúp đảm bảo tính nhất quán khi hiển thị thông báo lỗi trên toàn hệ thống.

# CHƯƠNG 4: HIỆN THỰC ỨNG DỤNG

## 4.1. Công nghệ sử dụng:

1. Phần mềm lập trình Java: IntelliJ IDEA
2. Phần mềm thiết kế hình ảnh:
  - + Công cụ thiết kế: <https://www.leshylabs.com/apps/sstool/>
  - + Nhân vật và vật phẩm trong game: <https://itch.io/game-assets/>
3. Mọi tài nguyên khác như backgrounds và âm thanh được lấy chủ yếu trên trang web <https://itch.io/game-assets/> và Youtube

## 4.2. Kết quả của chương trình minh họa:

- Các kết quả đạt được trong quá trình phát triển phần mềm:
  - + Đạt được kỹ năng phân tích và thiết kế, phát triển biểu đồ cho đề tài.
  - + Củng cố thêm kiến thức về Java và lập trình hướng đối tượng qua việc áp dụng các nguyên lý OOP để xây dựng trò chơi.
  - + Phát triển làm việc nhóm và kỹ năng giao tiếp trong nhóm.
  - + Học được các kỹ năng thiết kế đồ họa cơ bản cho game.
- Các chức năng chính đã thực hiện:
  - + Game đã có giao diện cơ bản: Có màn hình bắt đầu, kết thúc (khi nhân vật chết), hoàn thành, màn hình dừng, màn hình Credits.
  - + Tuy game chưa thực sự hoàn thiện nhưng hiện đã có thể chơi được như một bản DEMO (9 levels).
  - + Âm thanh và backgrounds đa dạng.
  - + Thuật toán cho kẻ địch đuổi theo và tấn công người chơi, cho Boss có thể đánh tầm xa được hoàn thiện.
  - + Người chơi có thể tương tác với object cơ bản (vật phẩm, enemy).
  - + Thiết lập thành công những sự kiện cơ bản khi người chơi gặp và đụng phải enemy (trúng độc, chịu sát thương khi enemy đi qua)

### 4.3. Tính năng tương tác trò chơi:

#### 1. Vật phẩm tăng sức mạnh

Trong quá trình chơi, người chơi có thể thu thập các vật phẩm giúp tăng sức mạnh cho nhân vật như tăng sát thương, hồi phục hoặc gia tăng giới hạn máu. Khi nhặt vật phẩm, hiệu ứng sẽ được kích hoạt ngay lập tức và phản ánh trực tiếp lên chỉ số của nhân vật.



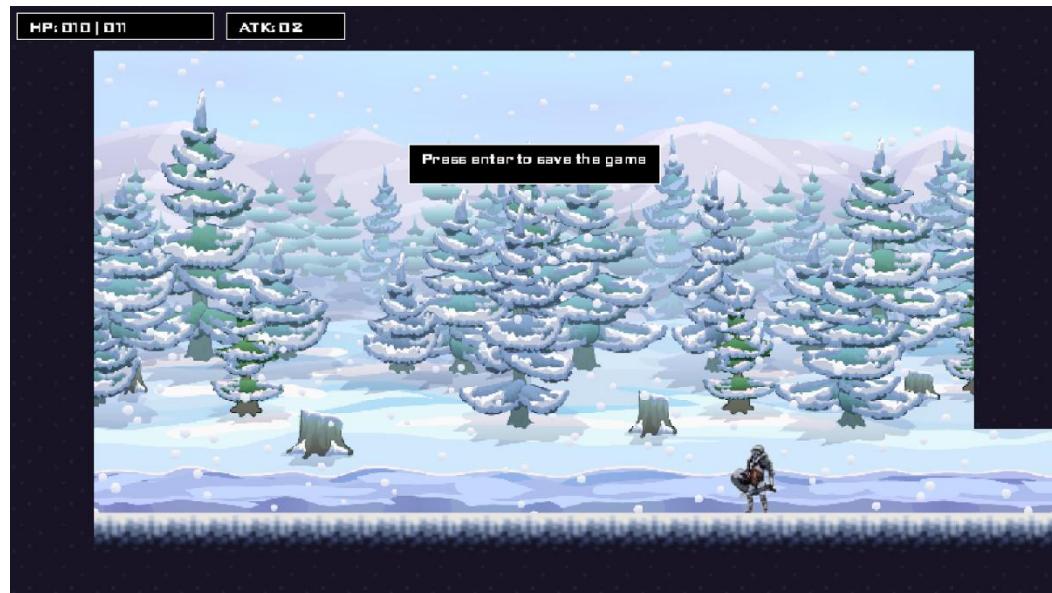
Hình 38: Vật phẩm 1



Hình 39: Vật phẩm 2

## 2. Phòng lưu tiến trình - Save Room

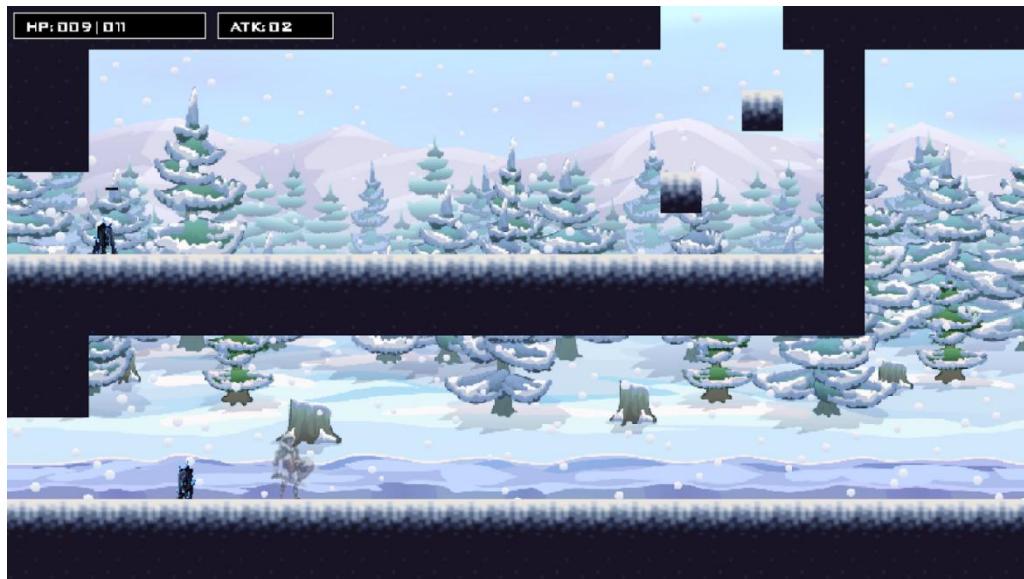
Save Room là khu vực đặc biệt cho phép người chơi lưu lại tiến trình chơi. Khi nhân vật tương tác với điểm lưu trong phòng, dữ liệu game sẽ được ghi lại, giúp người chơi tiếp tục từ vị trí đã lưu ở những lần chơi sau và giảm rủi ro khi thất bại.



Hình 40: Tính năng save game

## 3. Nhận sát thương khi bị quái tấn công

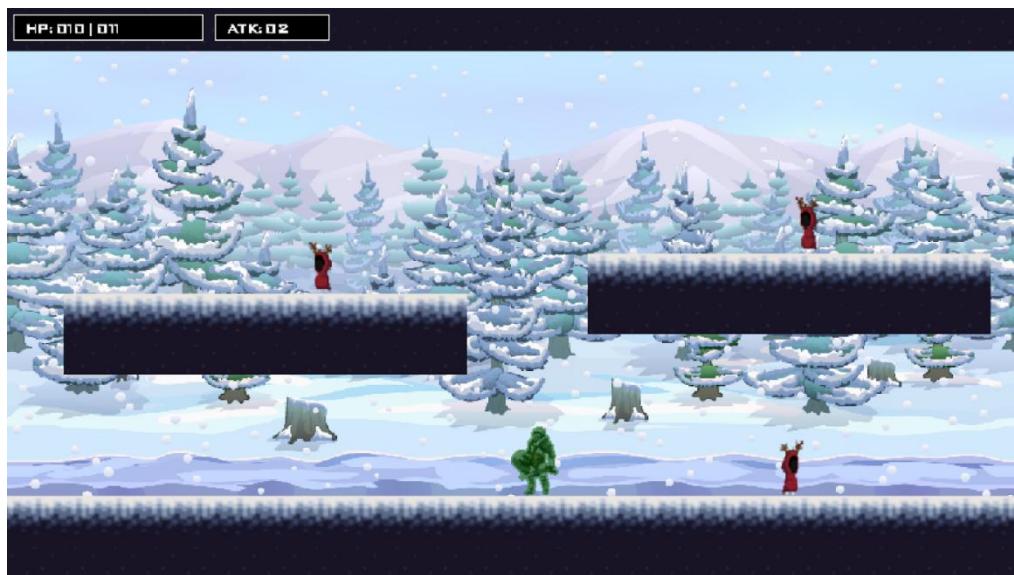
Khi nhân vật bị quái vật tấn công, chỉ số máu (HP) sẽ bị giảm tương ứng với lượng sát thương của kẻ địch. Trạng thái HP được cập nhật ngay trên giao diện, kèm theo hiệu ứng phản hồi giúp người chơi nhận biết và điều chỉnh chiến thuật né tránh hoặc phòng thủ phù hợp.



Hình 41: Bị quái đánh

#### 4. Nhận sát thương theo thời gian từ kẻ địch đặc biệt

Khi nhân vật bị tấn công bởi kẻ địch đặc biệt, người chơi sẽ không chỉ mất máu ngay lập tức mà còn tiếp tục bị trừ HP theo thời gian trong một khoảng ngắn. Hiệu ứng này buộc người chơi phải nhanh chóng né tránh, rời khỏi giao tranh hoặc sử dụng vật phẩm hồi phục để giảm thiểu thiệt hại và duy trì khả năng sinh tồn.



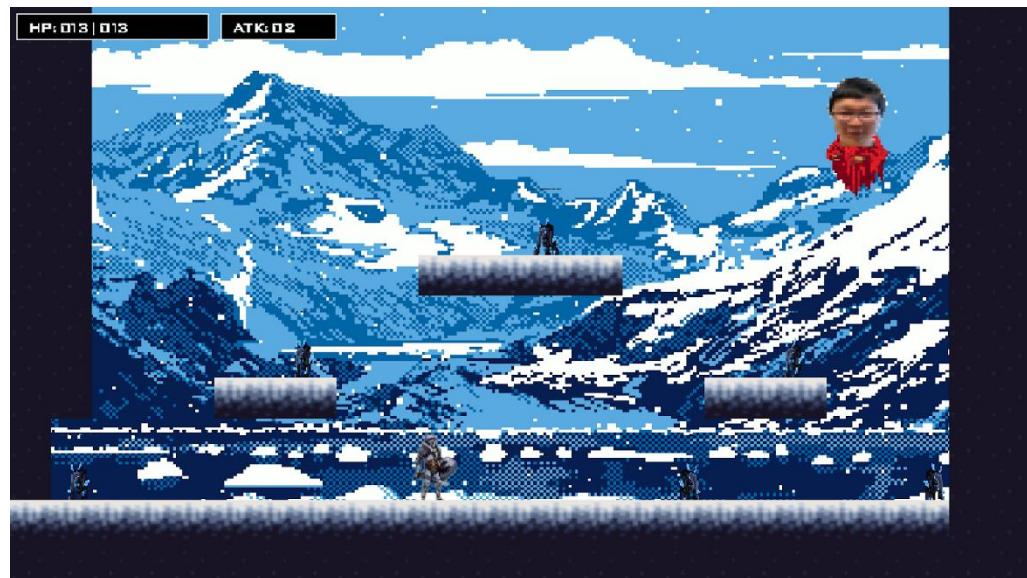
Hình 42: Bị nhận sát thương độc từ quái

## 5. Đánh boss

Trò chơi tích hợp các trận đánh boss với độ khó cao, yêu cầu người chơi quan sát hành vi tấn công và tìm ra quy luật di chuyển của boss. Mỗi boss sở hữu kỹ năng và cơ chế chiến đấu riêng, buộc người chơi kết hợp né tránh, tấn công hợp lý và tận dụng vật phẩm để giành chiến thắng.

## 6. Phần thưởng sau khi đánh Boss

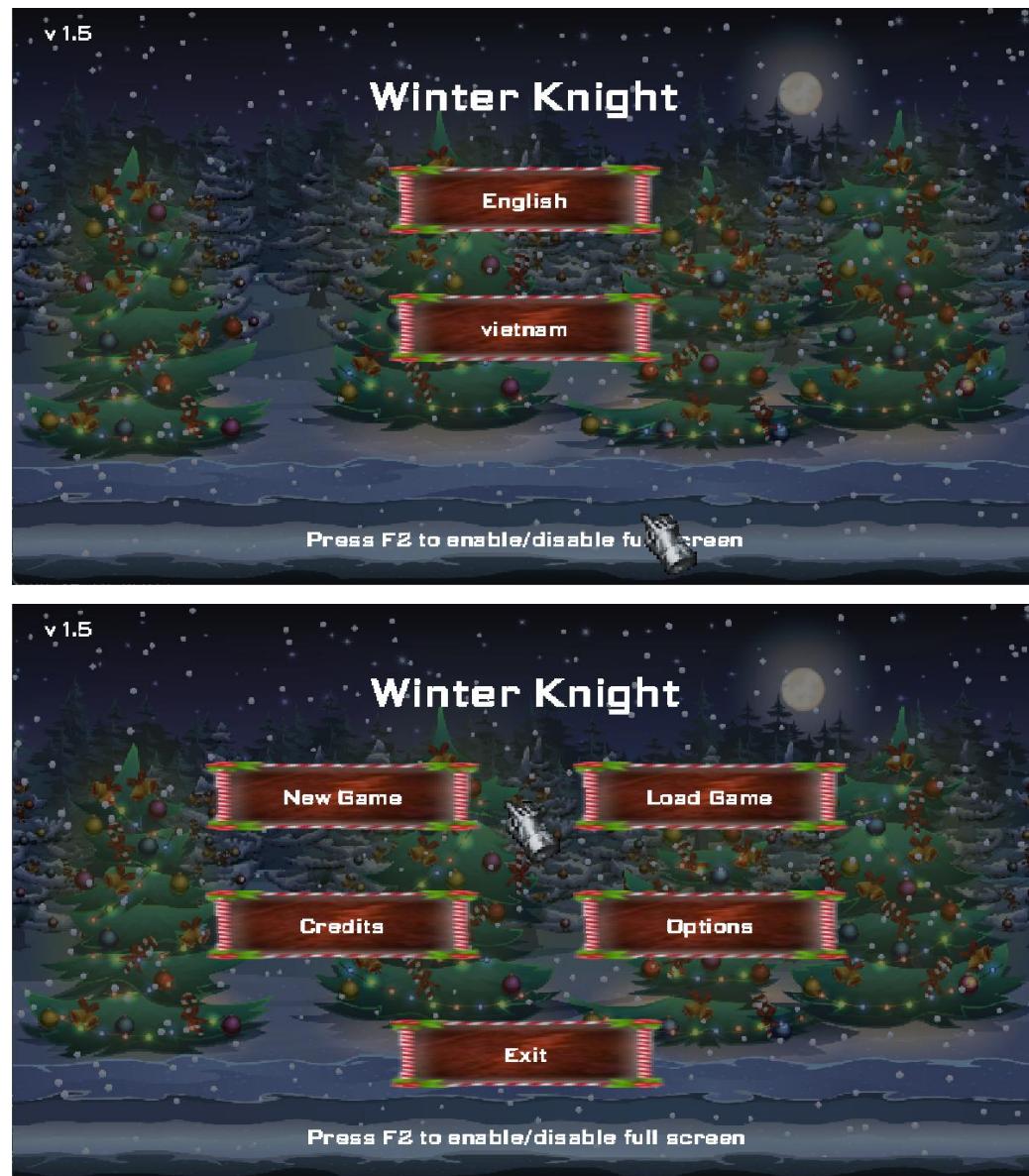
Sau khi đánh bại boss, người chơi sẽ mở khóa cơ chế **double jump**, cho phép nhân vật nhảy hai lần liên tiếp trên không. Cơ chế này giúp mở rộng khả năng di chuyển, tiếp cận các khu vực trước đó chưa thể tới và tăng tính khám phá trong lối chơi metroidvania.



Hình 43: Đánh boss và nhận vật phẩm double jump

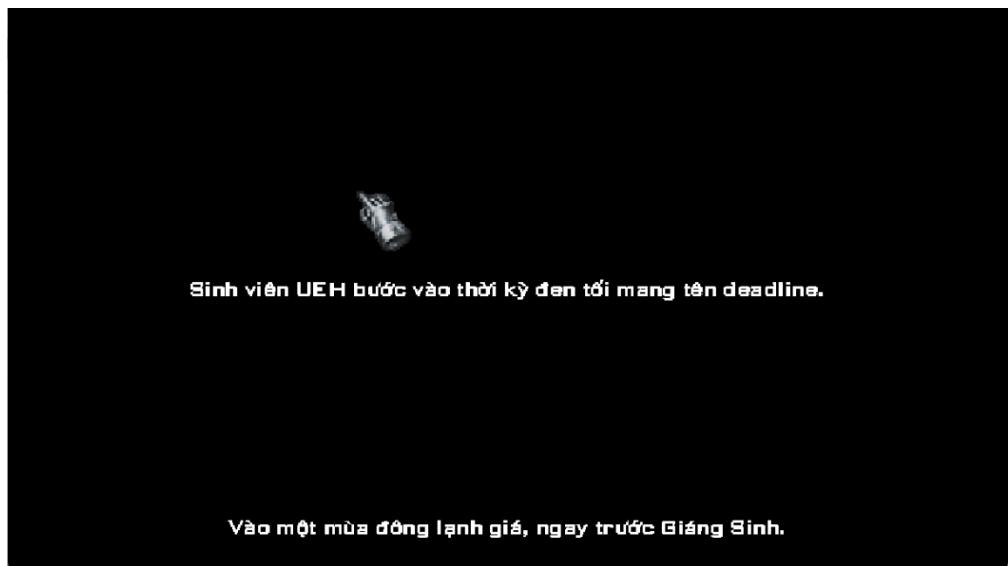
### 4.4. Giao diện chương trình:

Khi khởi chạy chương trình, hệ thống sẽ hiển thị màn hình khởi động của trò chơi. Tại đây, người chơi có thể lựa chọn ngôn ngữ hiển thị bằng cách nhấn vào tùy chọn English hoặc Vietnam. Sau khi hoàn tất việc chọn ngôn ngữ, người chơi tiếp tục lựa chọn các chức năng chính trên menu để tương tác với trò chơi.



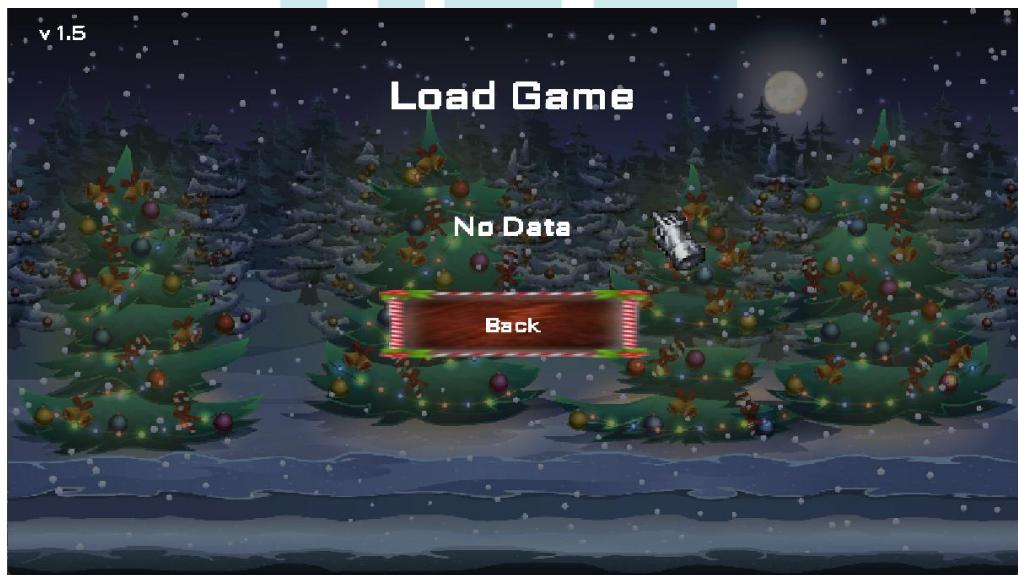
Hình 44: Main menu

Ấn Newgame: Cụ thể, khi người chơi chọn New Game, trò chơi sẽ khởi tạo một phiên chơi mới và bắt đầu quá trình trải nghiệm.



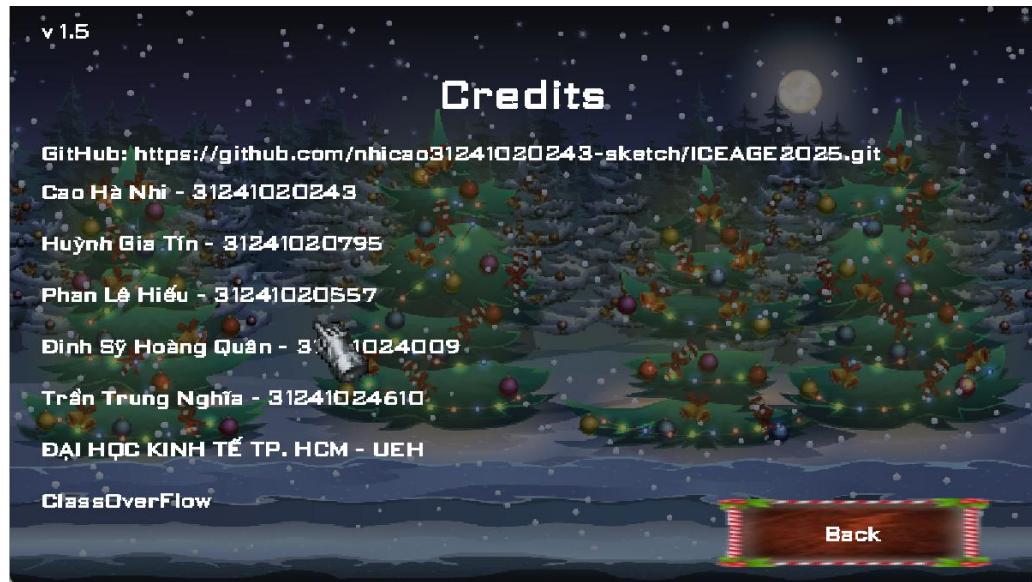
Hình 45: Cốt truyện game

Án Load game: Đối với chức năng Load Game, hệ thống chỉ hiển thị dữ liệu lưu trữ nếu người chơi đã từng chơi và có dữ liệu được ghi nhận trước đó; trong trường hợp chơi lần đầu, mục này sẽ không hiển thị thông tin.



Hình 46: Load game mà không có save data

Án Credits: Chức năng Credits cho phép người chơi xem thông tin về nhóm tác giả và những người tham gia phát triển trò chơi.



Hình 47: Credits game

Ấn Options: Ở mục Options, người chơi có thể tùy chỉnh các thiết lập như chế độ hiển thị toàn màn hình, số khung hình trên giây (FPS) và bật hoặc tắt âm thanh nền.



Hình 48: Options game

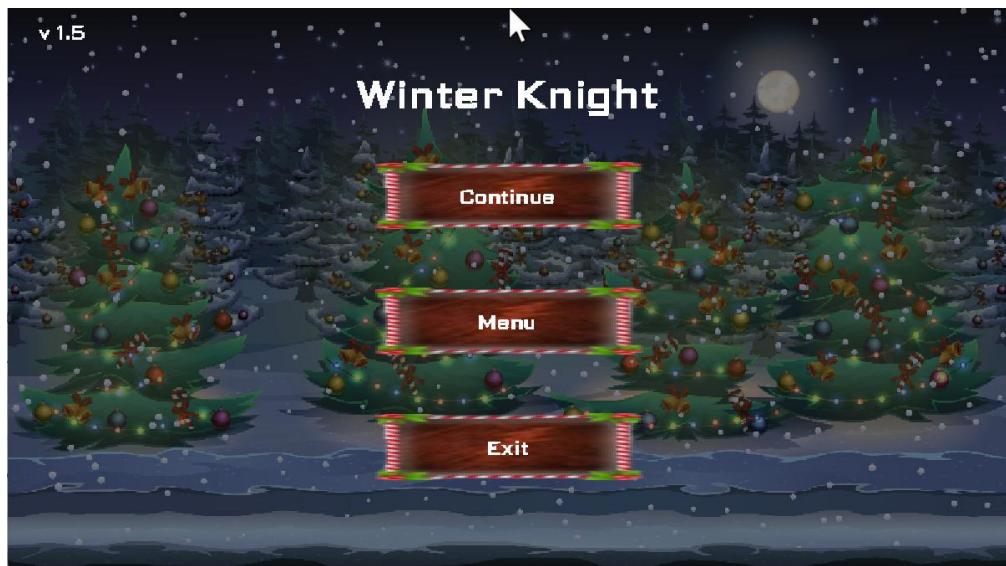
Ấn Exit: Khi lựa chọn Exit, hệ thống sẽ hiển thị hộp thoại xác nhận với hai tùy chọn Yes để thoát khỏi trò chơi và No để tiếp tục trải nghiệm.



Hình 49: Thoát game

Hướng dẫn chơi game rất đơn giản với những chức năng bao gồm di chuyển (ấn các phím W, A, S, D), tấn công (ấn phím Enter), hồi máu khi người chơi ấn phím Space khi di chuyển tới tượng (tính năng sẽ được nói rõ hơn ở phần tương tác).

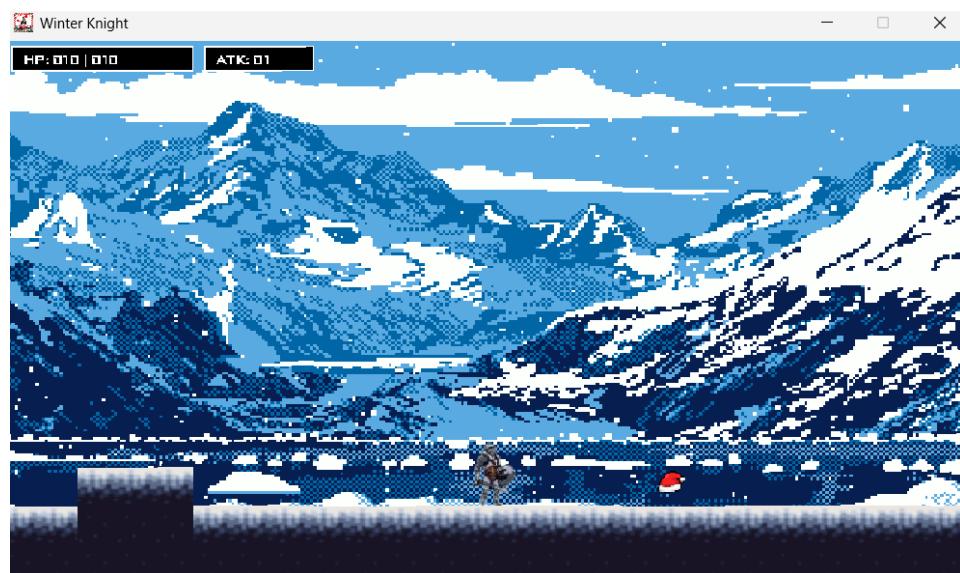
Để tạm dừng trò chơi, người chơi nhấn phím P hoặc ESC. Các chỉ số quan trọng như HP và ATK được hiển thị trực tiếp trên giao diện để người chơi dễ dàng theo dõi trạng thái nhân vật trong suốt quá trình chơi.



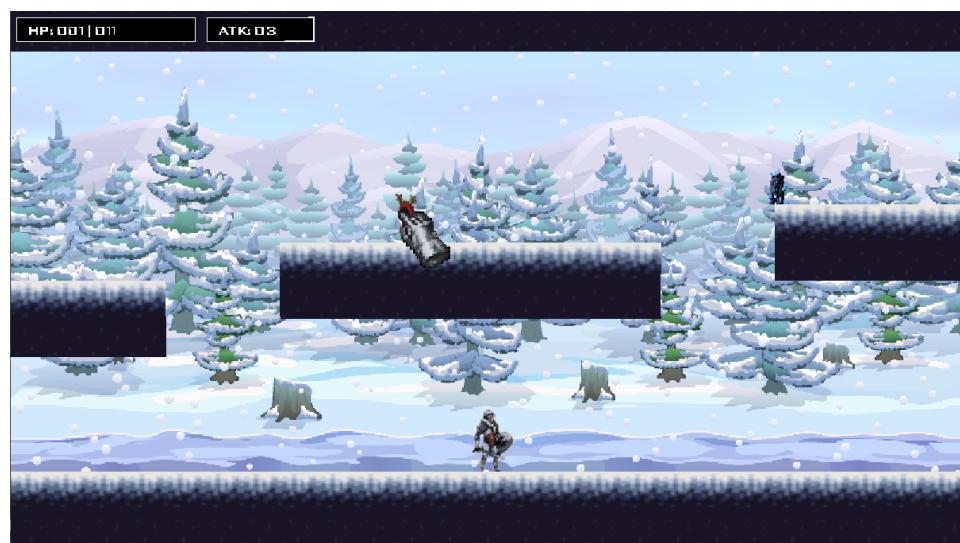
Hình 50: Pause game khi đang chơi

Người chơi điều khiển nhân vật di chuyển mũi tên trái/phải để sang trái và phải. n trong màn chơi bằng cách sử dụng phím A và D hoặc cách Để thực hiện thao tác nhảy, người chơi có thể nhấn phím Z, J hoặc Spacebar nhằm vượt qua địa hình và chướng ngại vật.

Trong quá trình chiến đấu, nhân vật có thể tấn công kẻ địch bằng cách nhấn phím X, K, hoặc sử dụng chuột trái / chuột phải. Chỉ số tấn công (ATK) của nhân vật sẽ được tăng lên khi người chơi thu thập các thanh kiếm xuất hiện trong màn chơi.



Hình 51: Nhân vật được buff sau khi nhặt vật phẩm



Hình 52: Nhân vật thấp máu

Khi lượng máu (HP) của nhân vật giảm về 0, hệ thống sẽ tự động kết thúc phiên chơi hiện tại và hiển thị màn hình *Game Over*. Tại thời điểm này, mọi hoạt động điều khiển của người chơi sẽ bị vô hiệu hóa, đồng thời trò chơi chuyển sang trạng thái kết thúc nhằm thông báo rằng người chơi đã thất bại trong màn chơi. Màn hình *Game Over* đóng vai trò cung cấp phản hồi trực quan về kết quả của quá trình chơi, giúp người chơi nhận biết rõ trạng thái thua cuộc và chuẩn bị cho các lựa chọn tiếp theo như chơi lại hoặc thoát khỏi trò chơi.



UNIVERSITY

Hình 53: Game over sau khi máu về 0

#### 4.5. Nhân vật chính và hình nền:

##### 4.5.1. Nhân vật chính

Trò chơi bao gồm 1 nhân vật chính là Winter Knight - một hiệp sĩ mặc giáp mang phong cách pixel art, tạo cảm giác cứng cáp và bền bỉ, phù hợp với môi trường lạnh giá trong Game:



Hình 54: Animation Winter Knight (I)

- Nhân vật sử dụng kiếm một tay làm vũ khí chính và khiên ở tay còn lại, cho phép vừa tấn công vừa phòng thủ trước các đòn đánh của kẻ địch.
- Dáng đứng vững vàng, tư thế sẵn sàng chiến đấu, thể hiện đây là một nhân vật thiên về cận chiến, tấn công kẻ địch ở hướng đối diện.
- Hiệp sĩ có khả năng:
  - + Di chuyển linh hoạt trên bản đồ theo điều khiển của người chơi.
  - + Tấn công cận chiến bằng kiếm, mỗi đòn đánh gây sát thương lên kẻ địch.
  - + Đỡ đòn (nhờ khiên) giúp giảm sát thương khi bị tấn công trực diện



Hình 55: Animation Winter Knight (2)

Winter Knight có 10 HP là lượng máu tối đa ban đầu. Khi bị tấn công:

- Bị kẻ địch “thường” đánh trúng: mất 1 HP.
- Bị đòn tấn công của BOSS: mất 2 HP.

Nhân vật có chỉ số ATK = 2, mỗi đòn tấn công thành công gây 2 sát thương lên kẻ địch.

- + Khi HP giảm về 0, Winter Knight sẽ gục ngã và trò chơi kết thúc.

+ Mục tiêu của người chơi là điều khiển Winter Knight tiêu diệt kẻ địch và đánh bại BOSS để hoàn thành trò chơi.

Dưới đây là hệ thống animation chính tạo nên một Winter Knight vô cùng sinh động và linh hoạt trong trò chơi:

- Di bộ:



Hình 56: Nhân vật đi bộ

- Nhảy:



Hình 57: Nhân vật nhảy

- Tấn công:



Hình 58: Nhân vật tấn công (1)



Hình 59: Nhân vật tấn công (2)

- Đứng im:



Hình 60: Nhân vật đứng im

- Tự vệ:



Hình 61: Nhân vật phòng thủ

- Nhân vật mất máu và chết:



Hình 62: Nhân vật mất máu



Hình 63: Nhân vật chết

=> Hệ thống animation đa dạng, bao gồm đứng yên, đi bộ, chạy, nhảy, tấn công, tấn công khi đang di chuyển, sử dụng vật phẩm và trạng thái bị đánh bại. Các chuyển động được xây dựng rõ ràng và mạch lạc, giúp người chơi dễ nhận biết từng trạng thái trong quá trình điều khiển.

#### 4.5.2. Hình nền

Background của trò chơi được thiết kế theo phong cách pixel art động, lấy bối cảnh vùng núi băng giá, tạo cảm giác lạnh lẽo, hoang vu và cô lập. Qua mỗi màn sẽ có 1 background khác nhau, nhằm tạo sự sinh động và đa dạng cho trò chơi. Winter Knight có 2 tệp hình nền chính:

##### 1. Mùa đông:

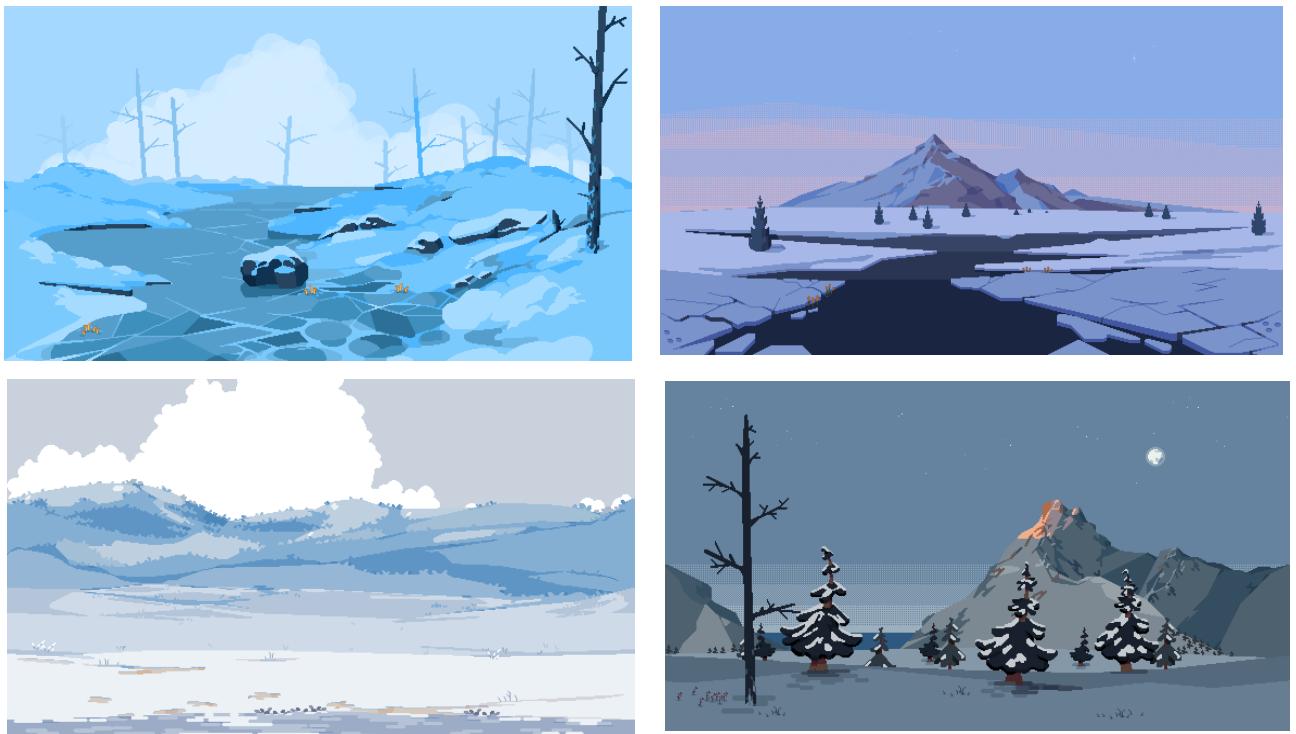


Hình 65: Background mùa đông (I)

Các vật thể trong background:

- Núi tuyết
- Mặt nước đóng băng
- Băng trôi
- Bầu trời mùa đông

=> Tông màu chủ đạo là xanh lam - trắng, các lớp nền được bố trí theo chiều sâu (parallax), mang cảm giác tĩnh lặng, làm nổi bật sự đơn độc của nhân vật.



Hình 66: Background Mùa đông (2)

Các vật thể trong các background:

- Núi tuyết ở hậu cảnh
- Đồi núi xa
- Đồng bằng phủ tuyết
- Bầu trời, mây
- Núi tuyết trung tâm
- Mặt băng / sông băng nứt
- Cây lá kim phủ tuyết
- Đồi tuyết
- Bầu trời mùa đông
- Núi tuyết
- Rừng thông phủ tuyết
- Cây khô
- Mặt trăng
- Bầu trời đêm
- Mặt băng phủ tuyết

- Tảng đá phủ tuyết
- Cây khô trơ trụi
- Đồi tuyết
- Bầu trời u ám

=> Các background được thiết kế với các lớp nền sắp xếp theo chiều sâu, bao gồm núi ở hậu cảnh, rừng, mặt băng và đồng bằng tuyết ở tiền cảnh, giúp không gian trở nên rộng và có chiều sâu. Sự phân lớp rõ ràng giữa các thành phần cùng với tông màu lạnh và ánh sáng nhẹ tạo nên khung cảnh vừa hùng vĩ, vừa tĩnh lặng và huyền bí, đồng thời làm nổi bật cảm giác lạnh giá và cô lập của môi trường trò chơi.

## 2. Ai cập:



Hình 67: Background Ai Cập

Các vật thể có trong các background:

- Tường đá cổ
- Cột đá chạm khắc
- Dây leo, cây bụi
- Thác nước nhỏ
- Bầu trời
- Kim tự tháp / đền bậc thang

- Cầu thang đá
- Cây cọ, cây bụi
- Tường đá cỗ
- Bầu trời
- Cổng đá cỗ
- Tường đá nút vỡ
- Dây leo
- Cây cọ, bụi cây
- Bầu trời
- Kim tự tháp đổ nát
- Khối đá lớn
- Cột đá chạm khắc
- Cây cọ
- Bầu trời hoàng hôn

=> Các background Ai Cập được sử dụng như bối cảnh phụ nhằm tăng sự đa dạng về hình ảnh cho trò chơi. Thiết kế mang phong cách pixel art với kiến trúc đá cỗ và cây cối giúp tạo điểm nhấn thị giác, trong khi vẫn giữ vai trò hỗ trợ, không làm mất đi trọng tâm bối cảnh chính là môi trường mùa đông.

### 3. Vật phẩm



Hình 68: Vật phẩm trong game

Hệ thống vật phẩm trong trò chơi được xây dựng theo chủ đề Giáng sinh, bao gồm nhiều loại vật phẩm quen thuộc và dễ nhận biết. Các vật phẩm chính gồm mũ Noel, kẹo gậy, bánh ngọt, bánh gừng, cây thông, quà Giáng sinh, người tuyết và chuông Noel. Một số vật phẩm được sử dụng để hồi máu, tăng chỉ số hoặc trang trí, giúp gameplay thêm đa dạng và sinh động. Nhìn chung, hệ thống vật phẩm có hình ảnh rõ ràng, màu sắc nổi bật, phù hợp với phong cách pixel art và góp phần tăng không khí lễ hội cho trò chơi.

#### 4. Logo game



Hình 69: Logo game

Logo game được thiết kế theo phong cách Giáng sinh với màu sắc tươi sáng và các chi tiết trang trí quen thuộc như mũ Noel, kẹo gậy, tạo cảm giác dễ thương và gần gũi. Kiểu chữ bo tròn, vui mắt giúp logo trông thân thiện, không quá nghiêm túc, phù hợp với không khí lễ hội của trò chơi. Tổng thể logo tạo ấn tượng nhẹ nhàng, vui vẻ và khiến người chơi cảm thấy thoải mái ngay từ cái nhìn đầu tiên.

## **4.6. Kẻ địch trong trò chơi:**

### **4.6.1 Kiến trúc hệ thống và Mô hình hướng đối tượng**

Hệ thống kẻ địch trong trò chơi được thiết kế dựa trên nguyên lý Đa hình (Polymorphism) và Ké thừa (Inheritance) triệt để, nhằm tối ưu hóa việc quản lý bộ nhớ và khả năng mở rộng của mã nguồn.

– Lớp cha trùu tượng (Enemy.java): Đây là lớp cơ sở định nghĩa toàn bộ các thuộc tính vật lý cốt lõi của một thực thể đối địch. Lớp này nắm giữ các tham số quan trọng như: Hộp va chạm (GameRectEntity), Máu (hp), Tốc độ di chuyển (speedX, speedY) và trạng thái hướng. Việc sử dụng lớp trùu tượng giúp nhóm chuẩn hóa logic di chuyển và va chạm cho tất cả quái vật mà không cần viết lại mã nguồn.

– Tích hợp với Vòng lặp Game :Ở phần Game.java, logic của kẻ địch được thực thi trong phương thức tick() của từng đối tượng. Hệ thống Game gọi scene.tick(), từ đó kích hoạt enemy.tick() cho từng quái vật đang tồn tại. Điều này đảm bảo trạng thái của kẻ địch được cập nhật đồng bộ với FPS của trò chơi (60 FPS).

### **4.6.2. Cơ chế vận hành vật lý**

Các hành vi vật lý của kẻ địch được cài đặt chi tiết trong lớp Enemy.java thông qua hai phương thức chủ đạo:

**1. Thuật toán xử lý trọng lực :** Nhóm không sử dụng thư viện vật lý có sẵn mà tự cài đặt logic kiểm tra va chạm điểm ảnh để tăng hiệu suất:

- Tại mỗi khung hình, hệ thống tạo một hộp va chạm ảo (hay dummy rect) nằm ngay bên dưới chân kẻ địch.
- Hàm scene.isFree() được gọi để kiểm tra vùng không gian đó.

+ Nếu giá trị trả về là True (Trống): Kẻ địch đang ở trên không, hệ thống cộng dồn trọng lực vào tọa độ Y, khiến kẻ địch rơi xuống.

+ Nếu giá trị trả về là False (Có vật cản): Kẻ địch được xác định là đang "đứng trên mặt đất", tọa độ Y giữ nguyên, cho phép thực hiện hành vi di chuyển.

### **2. Logic di chuyển tuần tra (Patrol AI Logic - toMove)**

Hành vi tuần tra, đi qua đi lại được hiện thực hóa bằng thuật toán “Dự đoán va chạm”:

Hệ thống tính toán vị trí dự kiến tiếp theo của kẻ địch dựa trên vận tốc hiện tại (speedX).

Trước khi thực sự di chuyển, một phép kiểm tra được thực hiện tại vị trí dự kiến đó.

Cơ chế quay đầu: Nếu vị trí dự kiến bị chặn bởi tường hoặc là vực thẳm, biến cờ isDirRight sẽ lập tức đảo giá trị. Điều này giúp kẻ địch tự động đổi hướng di chuyển một cách mượt mà mà không bị kẹt vào địa hình.

#### 4.6.3. Phân loại và Đặc tả hành vi Kẻ địch

##### A. Golem - Lính tuần tra mặt đất



Hình 70: Golem pack

- Golem là quái vật cơ bản của trò chơi, là những lính gác băng được Boss triệu hồi ở những nơi ngẫu nhiên trên bản đồ.
- Vai trò: Kẻ địch cơ bản kiểm tra kỹ năng phản xạ của người chơi.
- Logic khi Spawn : Khác với các đối tượng thông thường, Golem sở hữu một trạng thái ngay khi khởi tạo băng cách trôi từ mặt đất lên
- Trạng thái đầu tiên không phải là tấn công, mà là Spawn (Trôi lên từ đất).

- Trong hàm tick(), hệ thống kiểm tra finishedAnimation() của sprite Spawn. Chỉ khi hoạt ảnh này kết thúc, trạng thái mới chuyển sang di chuyển để Golem bắt đầu tuần tra.
- Tinh chỉnh hiển thị: Do kích thước sprite khác nhau giữa lúc Spawn và lúc Run, phương thức setSpritePosition() được ghi đè để điều chỉnh tọa độ vẽ, đảm bảo hình ảnh khớp hoàn hảo với hitbox.

## B. Shuriken (Phi tiêu bay) - Kẻ địch bay



- Shuriken là vũ khí được Boss chỉ định ở những nơi trọng yếu của bản đồ, có nhiệm vụ tăng thêm chướng ngại cho Knight ở những Map trò chơi.
- Cơ chế Vật lý đặc thù : Đây là ví dụ điển hình của tính Đa hình. Code của Shuriken được ghi đè lại hoàn toàn phương thức applyGravity() của lớp cha. Thay vì chịu lực hút xuống dưới, Skull di chuyển tự do theo trục dọc (Y).
- Thuật toán bay:
- Khởi tạo: Hướng bay ban đầu được chọn ngẫu nhiên
- Va chạm: Khi chạm trần hoặc sàn, vận tốc được nhân với -1 (speedY \*= -1), tạo ra quỹ đạo bay dích dắc liên tục.

### C. Thing (Nhân vật bí ẩn) - Kẻ địch được cho là phù thủy với sức mạnh độc tố

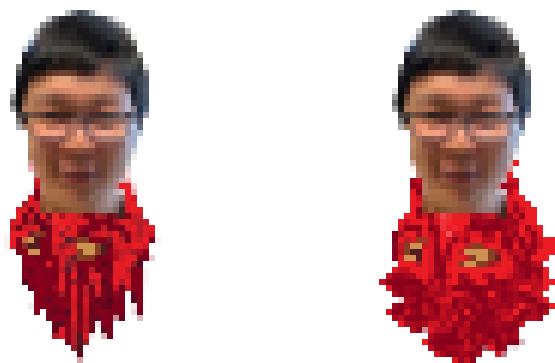


Hình 72: Pháp sư tuần lộc

- Thing là một nhân vật vô diện bí ẩn luôn khoác lên cho mình chiếc áo choàng ưu ám, đặc điểm nổi bật là chiếc sừng tuần lộc. Là tay sai được Boss cử đi để tiêu diệt Knight nhờ sức mạnh phép thuật gây độc cho người chơi
- Thông số kỹ thuật: Được thiết kế với chỉ số HP = 10 (gấp 3 lần Golem) và tốc độ di chuyển cực thấp.
- Cơ chế tấn công: Sử dụng loại sát thương độc tố gây DoT (Damaga Over Time). Điều này yêu cầu người chơi phải cẩn trọng hơn vì đính độc có thể gây hiệu ứng bất lợi kéo dài (thay đổi sprite nhân vật sang màu xanh).

#### 4.6.4. Thiết kế Trùm cuối

Boss được xây dựng như một thực thể độc lập với logic AI phức tạp nhất trò chơi, hoạt động dựa trên mô hình Máy trạng thái hữu hạn (Finite State Machine - FSM).



Hình 73: Thiết kế boss

#### Chu trình hoạt động:

1. **Trạng thái tĩnh (Chờ):** Boss đứng yên, hồi phục lại animation.

## 2. Logic khi Teleport (Dịch chuyển):

- Boss kích hoạt hoạt ảnh biến mất
- Ngay khi hoạt ảnh kết thúc, tọa độ của Boss được gán cứng luân phiên giữa hai vị trí chiến lược. Tại sao lại là 2? Việc thay đổi vị trí đột ngột này nhằm vô hiệu hóa khả năng "spam" chiêu thức của người chơi tại một điểm nhất định, nhằm mang đến lợi thế.
- Kích hoạt hoạt ảnh xuất hiện.

## 3. Logic triệu hồi Golem:

- Sau khi dịch chuyển, Boss chuyển sang trạng thái sử dụng phép thuật triệu hồi.



Hình 74: Boss triệu hồi quái

- Tại frame cuối của hoạt ảnh này, Boss tương tác trực tiếp với Scene để thêm mới các đối tượng Golem. Đây là cơ chế gia tăng độ khó theo thời gian thực.

## 4.7. Kiểm thử các chức năng thực hiện

Bảng 9: KIỂM THỬ CÁC CHỨC NĂNG ĐÃ THỰC HIỆN

Chức năng chọn ngôn ngữ	Hoạt động
Chức năng NewGame	Hoạt động
Chức năng Load Game	Hoạt động
Chức năng Credits	Hoạt động

Chức năng Options	Hoạt động
Chức năng Exit	Hoạt động
Chức năng đứng im của nhân vật	Hoạt động
Chức năng di chuyển của nhân vật	Hoạt động
Chức năng đánh của nhân vật	Hoạt động
Chức năng nhảy của nhân vật	Hoạt động
Chức năng chết của nhân vật	Hoạt động



## CHƯƠNG 5: BÁO CÁO QUÁ TRÌNH SỬ DỤNG AI

Bảng 10: BÁO CÁO QUÁ TRÌNH SỬ DỤNG AI

Nội dung	Đánh giá
Công cụ AI sử dụng	ChatGPT, Deepseek, Gemini, Claude
Mục đích sử dụng	Dò lỗi mã nguồn, phân tích nguyên nhân lỗi và các phương pháp chỉnh sửa để xuất để không làm hỏng game. Ngoài ra, nhóm cũng sử dụng AI để tìm các trang web hoặc công cụ có assets đồ họa phù hợp.
Đánh giá mức độ sử dụng AI	 1%
Prompt tiêu biểu	“Hãy dò lỗi của mã nguồn cho tôi và cho tôi biết lý do vì sao sai cũng như phương pháp sửa để không ảnh hưởng đến mã nguồn chính.”
Phạm vi sử dụng	Công cụ hỗ trợ tham khảo, không thay thế việc lập trình
Nhận xét	Nhóm luôn cố gắng hiểu bản chất của code để có thể chủ động tự sửa code, có sử dụng AI để hỗ trợ nhưng ở mức tối thiểu nhất có thể.

# CHƯƠNG 6: THẢO LUẬN VÀ ĐÁNH GIÁ

## 6.1. Kết quả đạt được

Sau quá trình thực hiện dự án, nhóm đã hoàn thành và xây dựng được trò chơi Winter Knight, một tựa game 2D thuộc thể loại metroidvania với chủ đề Mùa đông và Giáng sinh. Trò chơi đáp ứng được các mục tiêu đề ra ban đầu cả về mặt kỹ thuật lẫn trải nghiệm người chơi.

Nền tảng chức năng của trò chơi được xây dựng rất vững chắc với hệ thống điều khiển và chiến đấu cực kỳ nhạy bén. Sự kết nối giữa người chơi và nhân vật được đảm bảo thông qua các tương tác môi trường và đối đầu với Boss một cách trơn tru. Nhờ sự ổn định trong các cơ chế gameplay, *Winter Knight* mang lại cảm giác hành động không độ trễ, giúp dòng chảy của trò chơi luôn mạch lạc và lôi cuốn.

Về mặt thiết kế, *Winter Knight* tạo dựng một bản đồ liên thông đậm chất metroidvania, cho phép người chơi không ngừng khám phá và mở rộng thế giới thông qua việc thu thập kỹ năng và vật phẩm mới. Phong cách pixel art cùng hệ thống hiệu ứng và âm thanh được chau chuốt, hòa quyện với chủ đề mùa đông lạnh giá, qua đó góp phần định hình bầu không khí đặc trưng và bản sắc riêng biệt cho trò chơi.

Về mặt lập trình, nhóm phát triển đã tận dụng nguyên tắc lập trình hướng đối tượng (OOP) để tổ chức mã nguồn một cách hiệu quả, thông qua việc thiết kế các lớp riêng biệt cho nhân vật chính, kẻ địch, bản đồ cũng như hệ thống quản lý tổng thể của game. Nhờ đó, mã nguồn trở nên mạch lạc, dễ dàng mở rộng và thuận lợi cho việc bảo trì, đồng thời hỗ trợ tốt việc bổ sung tính năng mới trong tương lai.

Thông qua quá trình thực hiện dự án, nhóm đã củng cố kiến thức chuyên môn và nâng cao kỹ năng lập trình, đặc biệt là khả năng vận dụng lập trình hướng đối tượng vào việc xây dựng một sản phẩm game hoàn chỉnh. Đồng thời, dự án giúp nhóm rèn luyện kỹ năng làm việc nhóm, phân công nhiệm vụ và giải quyết vấn đề trong quá trình phát triển phần mềm. Trò chơi *Winter Knight* thể hiện kết quả học tập của nhóm

và là cơ sở để tiếp tục nghiên cứu, mở rộng và hoàn thiện trong các giai đoạn phát triển tiếp theo.

## 6.2. Tồn tại và hạn chế

### Tồn tại của hệ thống game

- Game đã được xây dựng theo hướng lập trình hướng đối tượng, với cấu trúc các lớp và package phân chia theo chức năng tương đối rõ ràng, giúp việc đọc hiểu và quản lý mã nguồn thuận lợi.
- Nhóm đã áp dụng các kiến thức OOP cơ bản như đóng gói, kế thừa và tái sử dụng mã nguồn, đặc biệt trong việc xây dựng hệ thống enemy và quản lý trạng thái game bằng enum.
- Với quy mô một game 2D nhỏ, cách tổ chức hiện tại giúp quá trình phát triển diễn ra nhanh, gameplay hoạt động ổn định và đáp ứng được các yêu cầu chức năng ban đầu.

### Hạn chế của hệ thống game

- Lớp trung tâm của game đang đảm nhiệm quá nhiều vai trò, từ xử lý logic chính đến điều phối các đối tượng, khiến code về lâu dài khó bảo trì và mở rộng.
- Hệ thống enemy phụ thuộc nhiều vào cơ chế kế thừa, dẫn đến nguy cơ trùng lặp mã nguồn và thiếu linh hoạt khi muốn bổ sung hành vi hoặc loại enemy mới.
- Một số lớp sử dụng biến và phương thức static để xử lý thuận tiện, nhưng điều này làm giảm khả năng mở rộng và gây khó khăn cho việc kiểm thử.
- Logic gameplay và phần hiển thị còn gắn chặt với nhau, cùng với việc nhiều thông số gameplay được hard-code trực tiếp trong code, khiến việc tinh chỉnh độ khó và phát triển thêm tính năng mới tốn nhiều thời gian và công sức.

### **6.3. Hướng phát triển của dự án**

Từ những hạn chế đã thấy khá rõ, nhóm xác định hướng phát triển tiếp theo là làm cho mã nguồn trở nên mềm hơn, đỡ cứng, dễ chỉnh sửa và mở rộng về sau, tránh tình trạng càng phát triển thì code càng rối và khó kiểm soát. Mục tiêu chung là vừa nâng cấp trải nghiệm gameplay cho bớt nhảm chán, vừa tối ưu lại quy trình phát triển để giảm bớt việc phải code tay những phần mang tính lặp lại.

Cụ thể, nhóm dự kiến bổ sung hệ thống vật phẩm và cơ chế nâng cấp nhân vật nhằm tăng chiều sâu cho lối chơi, chẳng hạn như thêm các loại giáp, khiên, vũ khí đặc biệt hoặc kỹ năng mới giúp nhân vật ngày càng mạnh hơn theo tiến trình chơi. Điều này không chỉ tạo cảm giác tiến triển rõ rệt mà còn giúp người chơi có thêm động lực tiếp tục khám phá game. Song song đó, nhóm sẽ tích hợp công cụ thiết kế màn chơi (như Tiled) để thay thế việc sắp xếp thủ công bằng mã nguồn, qua đó tiết kiệm thời gian phát triển và cho phép xây dựng các màn chơi lớn hơn, nhiều lớp hơn và có tính thẩm mỹ cao hơn.

Bên cạnh gameplay, trí tuệ nhân tạo của kẻ địch cũng sẽ được cải tiến để tránh tình trạng hành vi lặp lại và dễ đoán. Trong các phiên bản tiếp theo, quái vật sẽ được thiết kế với hành vi linh hoạt hơn, có khả năng phát hiện, truy đuổi, né tránh hoặc phối hợp tấn công, giúp các màn chơi trở nên thử thách và hấp dẫn hơn. Ngoài ra, nhóm hướng tới việc xây dựng hệ thống nhiệm vụ và cốt truyện nhằm chuyển trò chơi sang một hành trình có mục tiêu và nội dung rõ ràng, giúp người chơi cảm thấy gắn bó hơn với thế giới trong game và tăng giá trị chơi lại.

Trong tương lai xa hơn, nhóm cũng định hướng mở rộng dự án theo hướng tối ưu hiệu năng và nâng cấp giao diện, cải thiện trải nghiệm người dùng, đồng thời chuẩn hóa lại kiến trúc hệ thống để thuận tiện cho việc bảo trì và phát triển thêm các tính năng mới. Đây sẽ là nền tảng để game có thể tiếp tục được nâng cấp và hoàn thiện ở quy mô lớn hơn.

## 6.4. Đánh giá tổng thể tính chất OOP

**Bảng 11: ĐÁNH GIÁ TỔNG THỂ TÍNH CHẤT OOP**

NGUYÊN LÝ OOP	MỨC ĐỘ ÁP DỤNG CÁC NGUYÊN LÝ OOP (%)	NHẬN XÉT
Đóng gói (Encapsulation)	100%	Các lớp như Player, Enemy, Scene, Manager được thiết kế với phạm vi truy cập rõ ràng: “private, public, protected” đảm bảo dữ liệu và hành vi được kiểm soát chặt chẽ. Logic xử lý được tách biệt giữa các lớp, giảm sự phụ thuộc lẫn nhau và tăng tính an toàn của chương trình.
Kế thừa (Inheritance)	100%	Hệ thống Enemy và các đối tượng trong game được xây dựng theo mô hình kế thừa từ các lớp cha chung, giúp tái sử dụng mã nguồn hiệu quả, giảm trùng lặp và dễ dàng mở rộng thêm các loại đối tượng mới mà không ảnh hưởng đến hệ thống hiện có.
Đa hình (Polymorphism)	100%	Các phương thức chung như update(), render(), attack() được override linh hoạt ở các lớp con, cho phép game xử lý các đối tượng khác

		nhau thông qua cùng một interface hoặc lớp cha, đảm bảo tính linh hoạt và mở rộng của hệ thống.
Trừu tượng (Abstraction)	100%	Các lớp trừu tượng và interface được sử dụng nhằm tách phần định nghĩa hành vi khỏi phần cài đặt cụ thể. Nhờ đó, Scene và các Manager không phụ thuộc vào các lớp cụ thể mà chỉ làm việc với các kiểu trừu tượng, giúp giảm mức độ phụ thuộc giữa các thành phần. Thiết kế này cho phép dễ dàng thay thế hoặc mở rộng đối tượng mới mà không cần chỉnh sửa logic tổng thể của game.

## 6.5 Kết luận



Qua quá trình thực hiện đồ án, nhóm đã xây dựng được một trò chơi hoàn chỉnh dựa trên các nguyên lý lập trình hướng đối tượng. Hệ thống được tổ chức với cấu trúc tương đối rõ ràng, các lớp và package được phân chia theo chức năng, đồng thời vận dụng các khái niệm OOP cơ bản như đóng gói, kế thừa và tái sử dụng mã nguồn để đáp ứng các yêu cầu chức năng đã đề ra.

Bên cạnh những kết quả đạt được, hệ thống vẫn còn một số hạn chế về mặt thiết kế và khả năng mở rộng do phạm vi và thời gian thực hiện đồ án. Tuy nhiên, đồ án đã giúp nhóm củng cố kiến thức OOP và nâng cao kỹ năng phân tích, thiết kế hệ thống, tạo nền tảng cho việc phát triển game và áp dụng vào các dự án thực tế trong tương lai.

# TÀI LIỆU THAM KHẢO

*Toàn bộ sơ đồ UML ở chất lượng HD (Nên tải các file PDF về để có thể xem toàn bộ sơ đồ ở chất lượng cao nhất có thể): [LINK DRIVE SƠ ĐỒ UML \(CHẤT LƯỢNG HD\)](#)*

## 1. Font chữ:

License(s: <https://creativecommons.org/publicdomain/zero/1.0/>

<https://opengameart.org/content/public-pixel-font>

## 2. Hiệu ứng âm thanh:

attack.wav: [Jumping man sounds | OpenGameArt.org](#)

game-over.wav: [phải chịu - Instant Sound Effect Button | Myinstants](#)

jump.wav: [Jumping man sounds | OpenGameArt.org](#)

hit-player.wav: [5 Hit Sounds + Dying | OpenGameArt.org](#)

Nhạc nền game [Wham - Last Christmas \(8-bit\)](#)

Nhạc Boss: [O Christmas Tree](#)

## 3. Nhân vật:

Nhân vật chính: [KNIGHT](#)

Kẻ địch: [Skeleton](#)

Skull, Things, Boss (Do team tự thiết kế)

Vật phẩm trong trò chơi: [Free 20+ Christmas Icon Pack \(32x32\) by SODA](#)

## 4. Background:

Backgrounds Mùa Đông: [Free Winter Pixel Backgrounds by Free Game Assets \(GUI, Sprite, Tilesets\)](#)

Background Ai Cập: [Free Pixel Ancient Temple Backgrounds by Free Game Assets \(GUI, Sprite, Tilesets\)](#)

Background Giáng Sinh: <https://free-game-assets.itch.io/free-winter-holiday-2d-backgrounds>

## 5. Công cụ hỗ trợ:

Adobe photoshop; Paint 3D

[itch.io](https://itch.io); [leshy lab](https://leshylab.com)

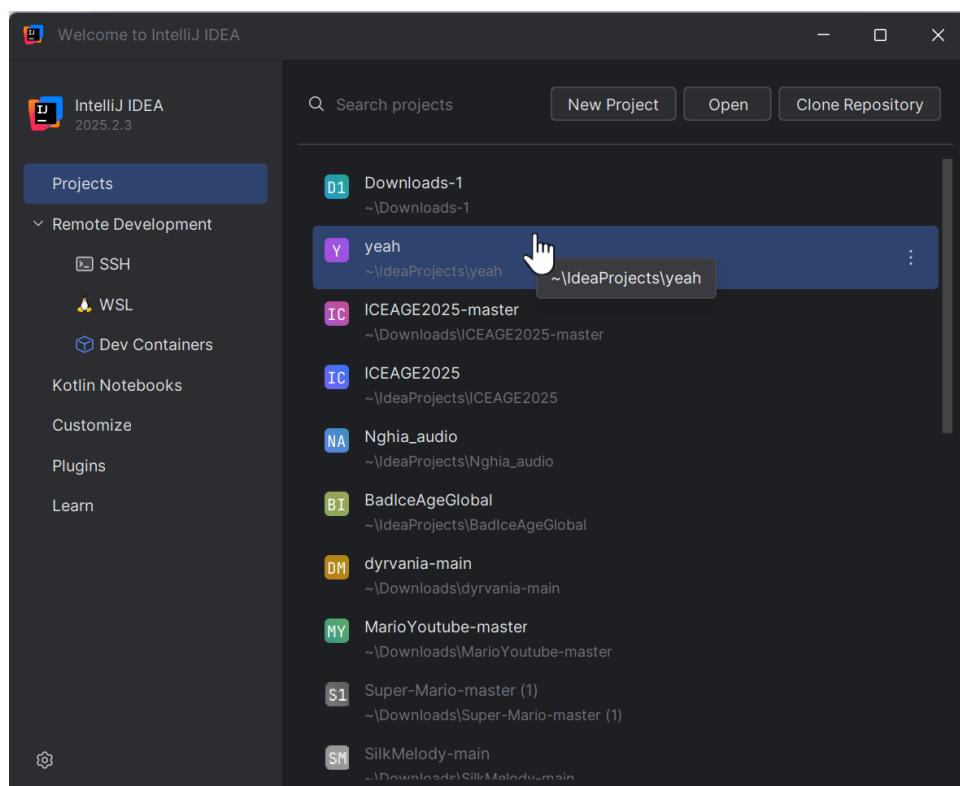
# PHỤ LỤC

## **Hướng dẫn cài đặt**

*Mã nguồn của dự án được lưu trữ và quản lý trên nền tảng GitHub tại địa chỉ:*  
<https://github.com/nhicao31241020243-sketch/ICEAGE2025.git>

### **Bước 1: Khởi động ứng dụng IntelliJ IDEA**

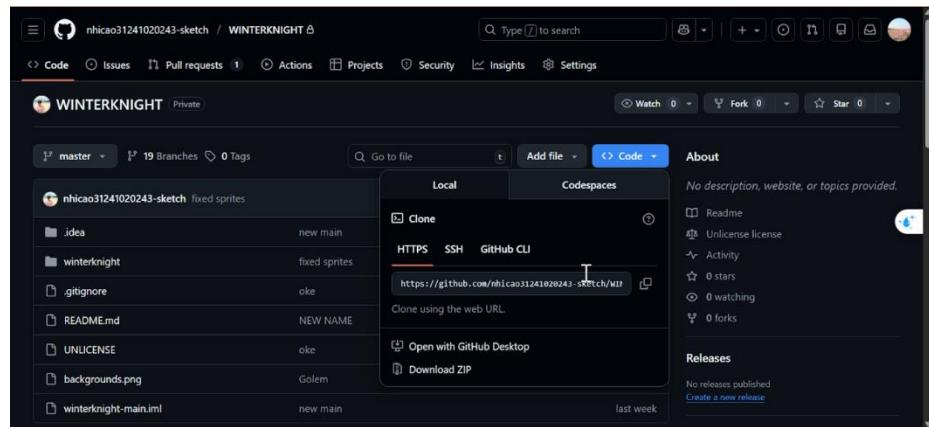
Người dùng tiến hành mở phần mềm IntelliJ IDEA, đây là môi trường phát triển tích hợp (IDE) được sử dụng để xây dựng và chạy dự án. Trước khi thực hiện các bước tiếp theo, cần đảm bảo rằng máy tính đã được cài đặt đầy đủ JDK phù hợp để chương trình có thể biên dịch và thực thi một cách ổn định.



### **Bước 2: Kết nối và lấy mã nguồn từ GitHub**

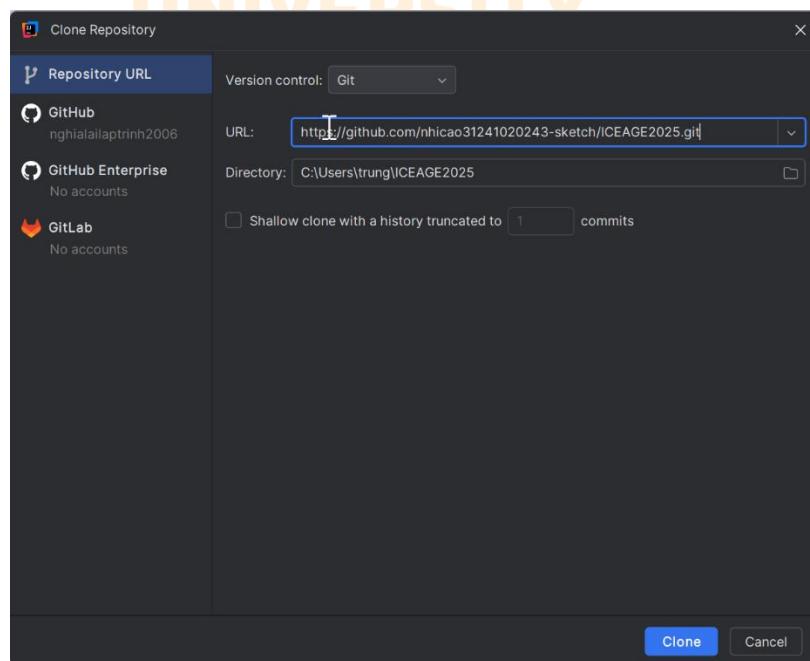
Tại giao diện khởi động của IntelliJ IDEA, người dùng chọn chức năng *Get from Version Control*. Sau đó, sao chép đường dẫn GitHub của dự án và dán vào ô *Repository*

*URL*. Bước này giúp IntelliJ IDEA kết nối trực tiếp với kho mã nguồn trực tuyến để chuẩn bị sao chép dữ liệu về máy.



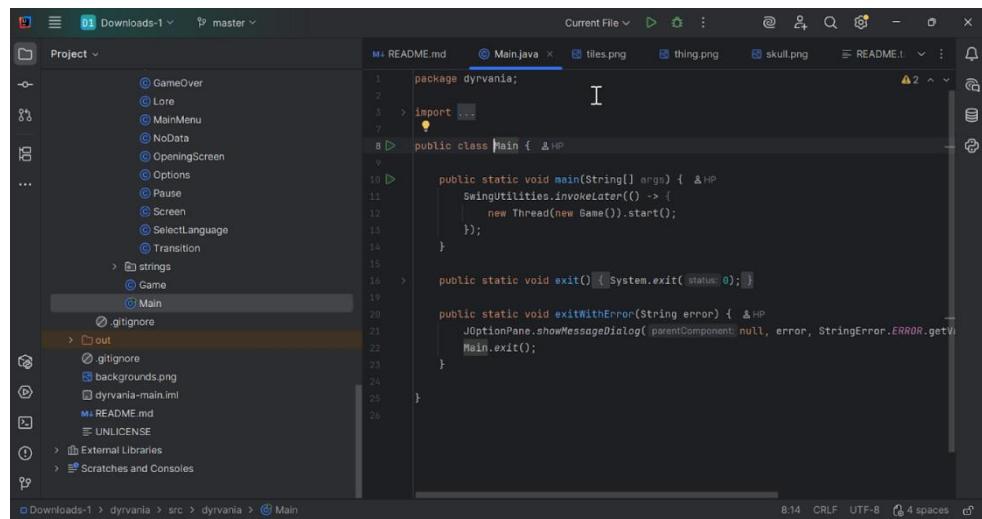
### Bước 3: Lựa chọn thư mục lưu trữ và sao chép dự án

Người dùng lựa chọn vị trí lưu trữ dự án trên máy tính tại mục *Path* sao cho phù hợp với cấu trúc thư mục cá nhân. Sau khi hoàn tất lựa chọn, nhấn nút *Clone* để bắt đầu quá trình tải toàn bộ mã nguồn của dự án về máy. Thời gian thực hiện bước này phụ thuộc vào tốc độ kết nối mạng và dung lượng dự án.



## Bước 4: Mở và kiểm tra cấu trúc dự án

Sau khi quá trình sao chép kết thúc, IntelliJ IDEA sẽ tự động mở dự án vừa được tải về. Người dùng cần kiểm tra lại cấu trúc thư mục, các package và lớp trong project để đảm bảo dữ liệu được tải đầy đủ và không phát sinh lỗi trong quá trình import.



## Bước 5: Chạy Game

Cuối cùng, người dùng xác định lớp chứa hàm main, đây là điểm bắt đầu của chương trình. Tiến hành chạy hàm main để khởi động trò chơi. Khi chương trình được thực thi thành công, giao diện game sẽ xuất hiện và người chơi có thể bắt đầu trải nghiệm.

