TRƯỜNG ĐẠI HỌC TÀI CHÍNH – MARKETING

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC LẬP TRÌNH JAVA

Đề tài:

PHÂN TÍCH VÀ XÂY DỰNG CHƯƠNG TRÌNH GAME PACMAN

Giảng viên hướng dẫn: Ths. Nguyễn Thanh Trường

Sinh viên thực hiện 1: Nguyễn Trung Kiên

Sinh viên thục hiện 2: Đỗ Phương Anh

TPHCM, tháng 4 năm 2023

TRƯỜNG ĐẠI HỌC TÀI CHÍNH – MARKETING

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC LẬP TRÌNH JAVA

Đề tài:

PHÂN TÍCH VÀ XÂY DỰNG CHƯƠNG TRÌNH GAME PACMAN

Giáng viên hướng dẫn: ThS. Nguyễn Thanh Trường

Sinh viên thực hiện 1: Nguyễn Trung Kiên – 2121005132

Sinh viên thực hiện 2: Đỗ Phương Anh – 2121005070

Mã lớp học phần: 2311112005802

TPHCM, tháng 4 năm 2023

LỜI CẢM ƠN

Đầu tiên, chúng em xin chân thành cảm ơn tới giảng viên phụ trách học phần Lập trình Java – Thạc sĩ Nguyễn Thanh Trường. Cảm ơn thầy vì đã tận tình chỉ dạy cho chúng em ạ. Chúng em cũng xin gửi lời cảm ơn tới các anh chị, các bạn trong lớp đã hướng dẫn tận tình trong quá trình chúng em thực hiện đồ án. Bởi lẽ đây là một dịp để chúng em có thể tiếp cận được với thực tiễn công việc cũng như là kiểm chứng, vận dụng các kiến thức lý thuyết được học trên giảng đường vào trong các ngữ cảnh hoạt động của một số đơn vị.

Với vốn kiến thức cũng như kinh nghiệm còn khiêm tốn, ít ỏi và là bước đầu thực hành làm quen với các công việc nghiên cứu nên có thể kết quả đạt được sẽ không tránh khỏi những hạn chế nhất định. Chúng em rất mong muốn được thầy nói riêng, các giảng viên, những anh chị sinh viên đi trước nói chúng sẽ quan tâm, góp ý để chúng em có thể hoàn thiện hơn cho đồ án.

Xin kính chúc Thạc sĩ Nguyễn Thanh Trường cùng tất cả những người đã hỗ trợ và đóng góp ý kiến cho chúng em một lời chúc sức khỏe, hạnh phúc và thành đạt nhất.

NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 1

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

- Điểm số: .....
- Điểm chữ: .....

TPHCM, ngày ... tháng ... năm 2023

Giảng viên

Nguyễn Thanh Trường

NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 2

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

- Điểm số: .....
- Điểm chữ: .....

TPHCM, ngày ... tháng ... năm 2023

Giảng viên

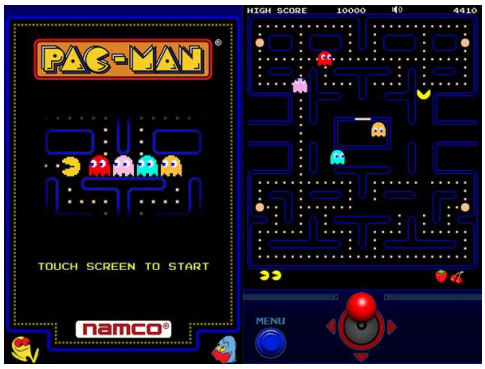
MỤC LỤC HÌNH ẢNH

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu PacMan:

PacMan, ban đầu được gọi là Puck Man, là một trò chơi điện tử hành động mê cung do Namco phát triển và phát hành đầu tiên tại Nhật Bản vào 22 tháng 5 năm 1980 cho các trò chơi arcade. Ở Bắc Mỹ, trò chơi được phát hành bởi Midway như một thỏa thuận cấp phép với Namco.



Hình 1. 1 Game Pacman nguyên bản

Kết hợp được những thể loại phổ biến của thời đại những năm 80, Pac – Man nhanh chóng thành công và được nhắc đến như một bước ngoặt lớn trong lịch sử trò chơi điện tử, cũng là một trong những trò chơi arcade nổi tiếng nhất mọi thời đại. Nó còn là trò chơi đoạt doanh thu cao nhất và bán chạy nhất, tạo ra doanh thu hơn 14 tỷ đô là (tính đến năm 2016) và tổng doanh thu 43 triệu đơn vị, đồng thời là một di sản văn hóa và thương mại lâu dài. Pac – Man là trò chơi tồn tại lâu nhất trong Kỷ nguyên vàng của trò chơi arcade, và là một trong ba trò chơi điện tử duy nhất có mặt tại viên Smithsonian ở Washington D.C.

Pac – Man là một trò chơi điện tử đuổi theo mê cung hành động, người chơi điều khiển các nhân vật cùng tên thông qua một mê cung kèm theo và ăn các chấm pac (pac – dots). Nếu người chơi ăn hết các chấm pac và đồng thời tránh được 4 con ma màu là Blinky (đỏ), Pinky (hồng), Inky (lục lam) và Clyde (cam) đang truy đuổi theo thì Pac – Man được đưa qua các cấp độ chơi mới. Các cấp độ được biểu thị bằng trái cây ở cuối màn hình. Nếu Pac – Man bị ma

bắt, anh ta sẽ mất mạng, và trò chơi sẽ kết thúc khi tất cả các mạng sống đều mất. Mỗi con trong số bốn con ma đều có những trí tuệ nhân tạo (AI) hoặc "tính cách riêng" như: Blinky đuổi theo trực tiếp Pac – Man, Pinky và Inky sẽ cố định vị mình trước Pac – Man, thường là bằng cách dồn anh ta vào chân tường; còn Clyde sẽ chuyển đổi giữa việc đổi theo Pac – Man và chạy trốn khỏi anh ta. Được đặt ở 4 góc của mê cung là những "viên năng lượng" nhấp nháy lớn. Ăn những thứ này sẽ khiến Pac – Man có thể ăn được những con ma trong thời gian ngắn để nhận được điểm thưởng.

1.2. Lý do chọn đề tài:

Hiện nay, các game điện tử nói chung và game Pac Man nói riêng đều được rất nhiều người quan tâm và sử dụng. Các thể loại game này không chỉ giúp cho chúng ta thư giãn đầu óc, mà còn có thể giúp rèn luyện trí não. Để tạo các trò chơi điện tử như vậy, ít nhất cần có những kiến thức về các ngôn ngữ lập trình như C++, C#.net, Java, ... Các ngôn ngữ lập trình này rất phù hợp cho các bạn sinh viên đang học lập trình, và có thể dễ dàng thử sức để sáng tạo ra các trò chơi điện tử không quá khó.

Trong đồ án này, nhóm chúng em sẽ nghiên cứu về đề tài "**Phân tích chương trình game PacMan"**. Trong đồ án này, chúng em sẽ thiết kế một phiên bản đơn giản, tinh gọn hơn của trò chơi này, chủ yếu xoay quanh phân tích code và cách vận hành game.

1.3. Mục đích và mục tiêu của đồ án:

- ☐ Phân tích và thiết kế hệ thống, hướng đối tượng Game PacMan gồm các lớp, tính năng hỗ trợ chạy game.
- ☐ Nghiên cứu quy trình lập trình, sự vận hành của game.
- ☐ Xây dựng được giao diện chơi dễ sử dụng, rõ ràng, mang lại cảm giác tốt khi chơi.

1.4. Nội dung đề tài:

Chương 1: Tổng quan về Game Pac – Man

Chương 2: Cơ sở lý thuyết

Chương 3: Phân tích và xây dựng chương trình

Chương 4: Thực hiện chương trình

Chương 5: Tổng kết

1.5. Đối tượng và phạm vi đề tài:

1.5.1. Đối tượng đề tài:

Nghiên cứu về game Pac - Man và phân tích, xây dựng chương trình gồm chức năng của game: di chuyển, tránh các kẻ thù, ăn các chấm pac, lên level và tạo giao diện game, các lớp hỗ trợ tính năng, …

1.5.2. Phạm vi đề tài:

Đề tài được nghiên cứu với ngôn ngữ Java là ngôn ngữ lập trình chính, phần mềm Netbeans 12.6 phát triển tích hợp (IDE) cho Java.

1.6. Phương pháp nghiên cứu:

Phương pháp phân tích và tổng hợp để đánh giá hiện trạng.

Phương pháp phân tích và xây dựng chương trình game.

1.7. Dự kiến kết quả đạt được

Đề tài "Phân tích chương trình Game Pac Man" được xây dựng nhằm đạt được những kết quả như sau :

- ☐ Nghiên cứu công cụ sử dụng, phân tích công cụ lập trình JDK 1.8, NETBEAN 12.6.
- ☐ Phân tích thiết kế hệ thống quản lý giao diện chương trình game.
- ☐ Lập trình các chức năng của chương trình.

- ☐ Xây dựng giao diện chương trình game.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Yêu cầu bài toán:

Đảm bảo game có đầy đủ các tính năng hỗ trợ chạy game:

- ☐ Tạo mê cung, các kẻ địch.
- ☐ Xử lý các sự kiện va chạm, ăn điểm, qua màn,…
- ☐ Thông báo "số mạng" của người chơi
- ☐ Thông báo thắng thua cho người chơi.

Các giao diện trò chơi:

- ☐ Giao diện chơi game: gồm mê cung, phía dưới bên phải là điểm của người chơi, phía dưới bên trái là số mạng của người chơi.
- ☐ Giao diện thông báo thắng thua

2.2. Giới thiệu và phân tích các đối tượng trong game:

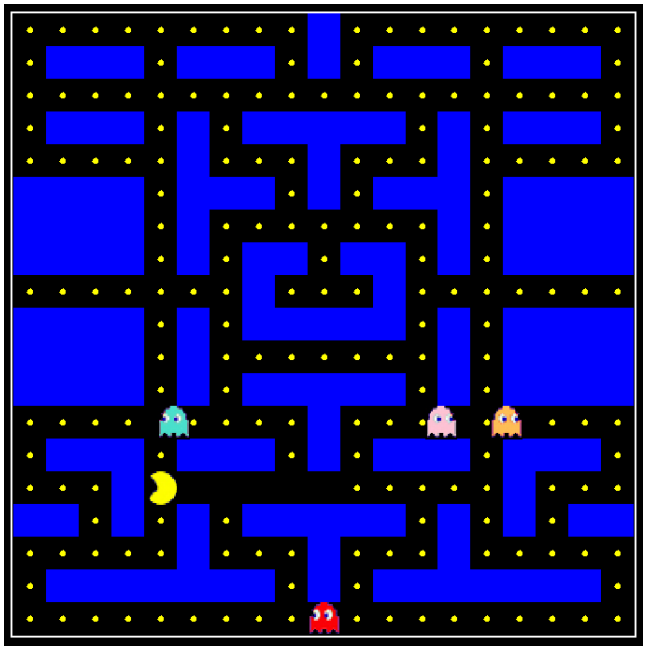
2.2.1. Màn hình chờ:

Màn hình chờ cung cấp các thông tin cơ bản về game cho người chơi.

Hình 2. 1. Màn hình chờ

2.2.2. Mê cung:

Mê cung là nơi PacMan di chuyển để ăn các điểm và tránh các kẻ thù ăn bản thân.



Hình 2. 2 Mê cung trong game

2.2.3. Kẻ địch:

Kẻ địch hay còn gọi là các con ma (Ghost) được tạo ra để tiêu diệt PacMan, ngăn cản PacMan ăn các điểm để lên cấp.

Trong trò chơi Pacman, có tổng cộng 4 con ma:

Blinky - màu đỏ

Pinky - màu hồng

Inky - màu xanh da trời

Clyde - màu cam

Hình 2. 3 Con ma – kẻ địch của Pacman (từ trái qua Inky, Pinky, Clyde, Blinky)

2.2.4. PacMan

Nhân vật chính trong trò chơi, được người chơi điều kiển với mục tiêu là ăn các điểm trong mê cung để tăng điểm và lên level, bình thường Pacman luôn sợ các con ma và luôn phải tránh né chúng.

Hình 2. 4 Pacman

2.2.5. Thông số:

Thông số cung cấp những thông tin, số liệu quan trọng cho người chơi trong một màn chơi bao gồm:

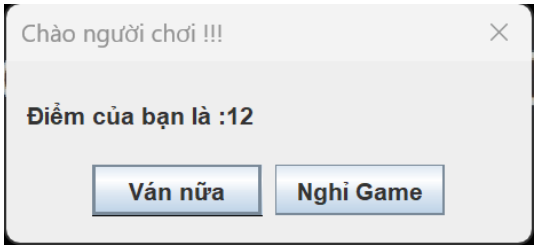
- ☐ Số mạng còn lại.
- ☐ Điểm của người chơi.

Hình 2. 5 Thông số của người chơi

2.2.6. Thông báo:

Khi người thắng hoặc thua, trò chơi sẽ kết thúc và hiện ra bảng thông báo kèm hai sự lựa chọn:

1. Ván nữa (bắt đầu chơi một ván game mới).
2. Nghỉ Game (Thoát khỏi trò chơi).

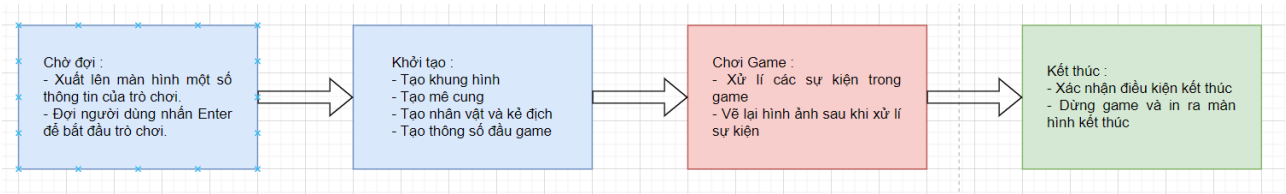


Hình 2. 6 Thông báo khi end game

2.3. Phân tích và thiết kế giải thuật:

2.3.1 Mô hình tổng quát:

Vấn đề thiết kế và xử lý chế độ khi chơi phải trải qua một số bước, có thể được tóm gọn theo mô hình:



Hình 2. 7 Mô hình tổng quát các bước xử lí trò chơi

2.3.1.1. Chờ đợi:

Ở giai đoạn này, các công việc xử lý bao gồm:

Thiết kế giao diện vào game: Ở đây màn hình được thiết kế để người chơi nhập nickname và chọn các lựa chọn chơi hoặc thoát (trường hợp để trống tên sẽ không được chấp nhận và thông báo lỗi).

Truyền các biến cần thiết: Các biến được truyền vào và được lưu lại sử dụng.

2.3.1.2. Khởi tạo:

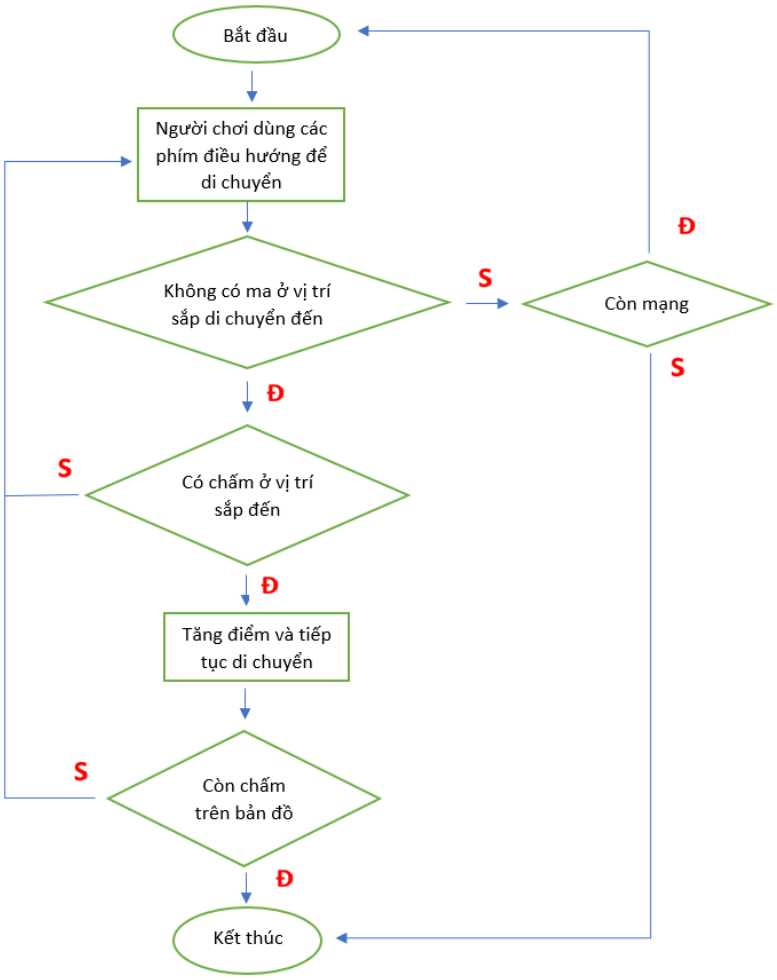
Các công việc trong giai đoạn này bao gồm:

- Khởi tạo một Frame là màn hình chơi.
- Vẽ bản đồ lên Frame đã tạo.
- Vẽ các nhân vật và kẻ thù lên bản đồ.
- Khởi tạo các giá trị, thông số bắt đầu như : tốc độ Pacman, số lượng ma, bản đồ, mạng, điểm,… sẽ được khởi tạo và nạp vào trò chơi.

2.3.1.3. Chơi game:

Các công việc trong giai đoạn này bao gồm:

- Nhận các sự kiện từ bàn phím sau đó thực hiện các chức năng tương tự như: di chuyển, bắt đầu, …
- Nhận các sự kiện khi các đối tượng tương tác nhau như: va chạm, di chuyển trong mê cung, chết, …



Hình 2. 8 Mô tả điều kiện kết thúc trò chơi

2.3.1.3. Kết thúc trò chơi:

Điều kiện kết thúc trò chơi là 1 trong 2 trường hợp :

- Thắng: khi người chơi đã ăn hết tất cả các điểm trong mê cung và số mạng vẫn còn lớn hơn 0.
- Thua: khi số mạng trở về 0 trước khi ăn hết tất cả các điểm trong mê cung.

Tips: cố gắng tránh né các kẻ thù và ăn thật nhiều điểm trước khi thua cũng có thể coi là một cách chiến thắng.

2.3.2 Tạo bản đồ:

Bảng đồ được tạo thông qua các bước sau:

- Tô đen toàn bộ nền của khung hình.
- Vẽ điểm PacMan cần ăn lên trên khung hình.
- Vẽ các khối màu xanh dương đại diện cho tường ở những ví trí và kích thước tương ứng.
- Vẽ Pacman và các con ma tại vị trí tương ứng.

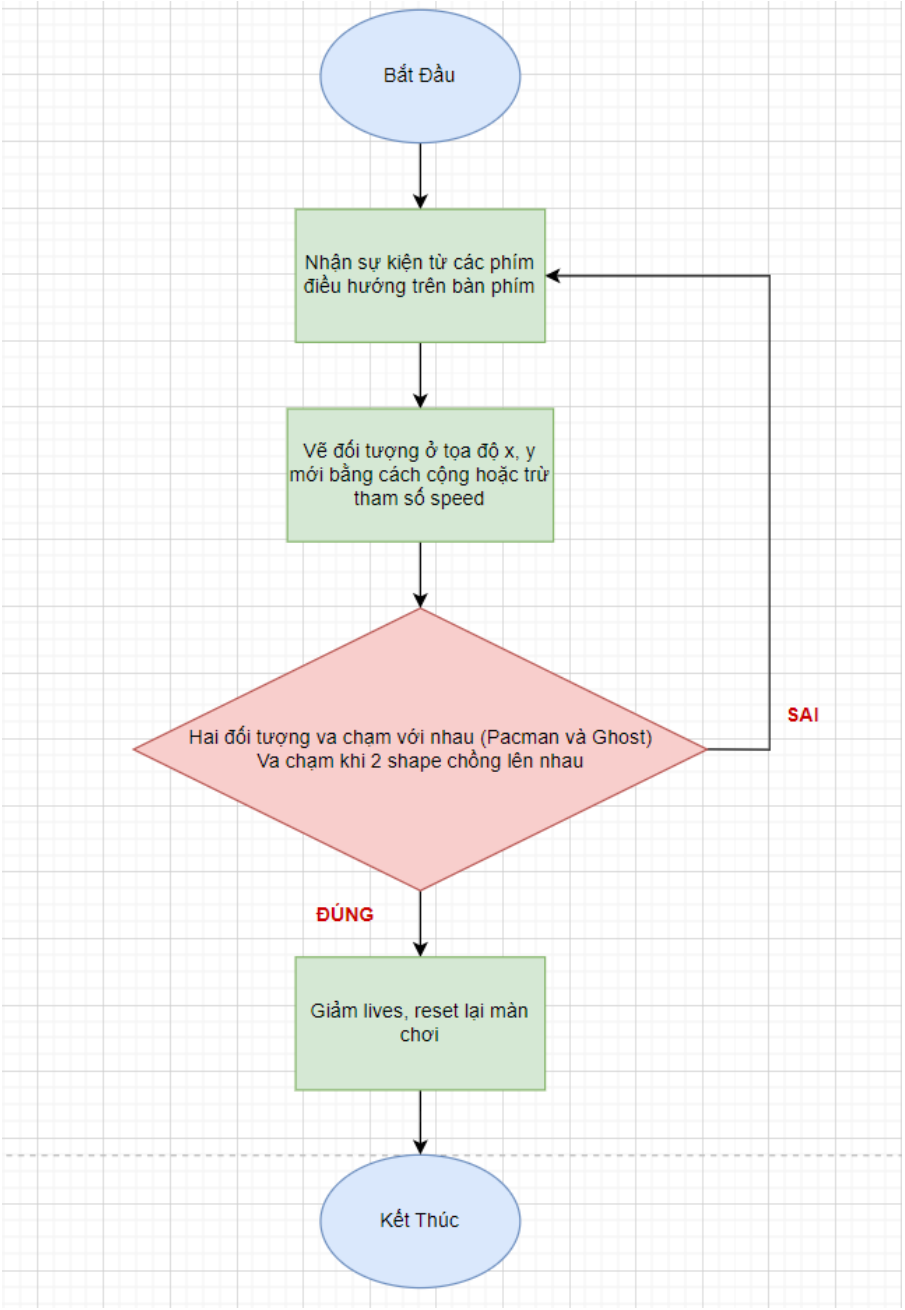
Việc tạo bảng đồ này nhằm mục đích giúp Pacman và ma cũng như các phương thức khác nhận biết được các vị trí trên background từ đó làm cho các đối tượng trong game không di chuyển hoặc xuất hiện ở những vị trí lộn xộn.

2.3.3 Xử lí va chạm

Xử lí va chạm là một trong những công việc quan trọng nhất khi lập trình game. Đối với game Pacman, va chạm xảy ra trong 3 trường hợp :

2.3.3.1 Pacman va chạm với ma:

Trong khi nhân vật Pacman di chuyển, các con ma cũng đồng thời di chuyển khắp bản đồ theo một hàm random số từ đó ngẫu nhiên lựa chọn các hướng lên, xuống, trái, phải mà không theo quy tắc nào, mục tiêu của những con ma nhằm cố gắng bắt nhân vật Pacman. Khi đó sau một thời gian sẽ xuất hiện trường hợp Pacman và ma va chạm với nhau. Trường hợp trên sẽ được xử lí theo lưu đồ sau :

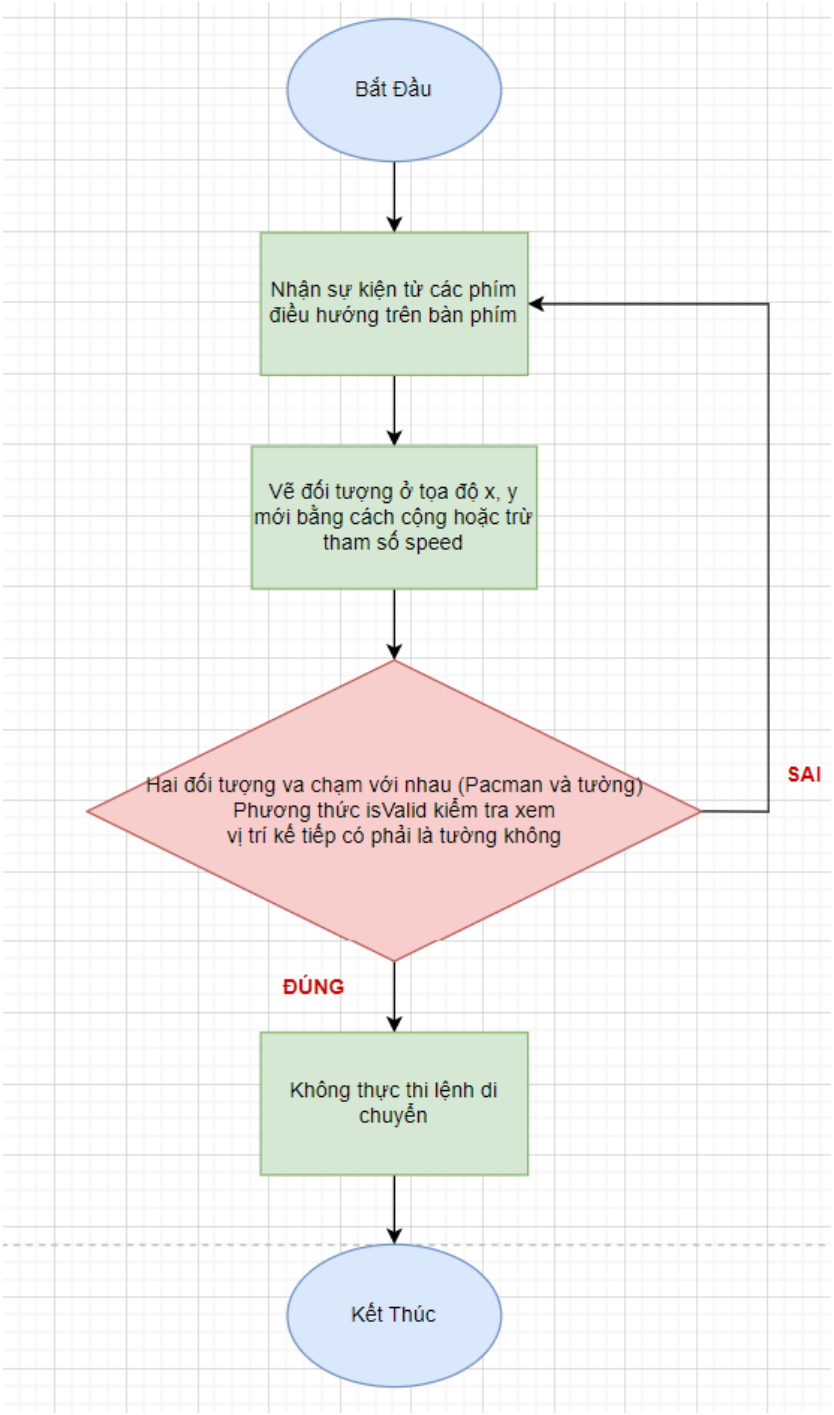


Hình 2. 9 Mô tả điều kiện khi Pacmna va chạm với ma

2.3.3.2 Pacman khi va chạm với tường:

Pacman sẽ di chuyển bên trong mê cung đã được tạo, để làm được việc đó chúng ta cần tạo một thuật toán nhằm xác định đâu là đường đi, đâu là tường nhờ đó đó dẫn lối cho có thể đi theo các lối đi đã được xây dựng sẵn, tránh trường hợp chạy lung tung khắp bản đồ.

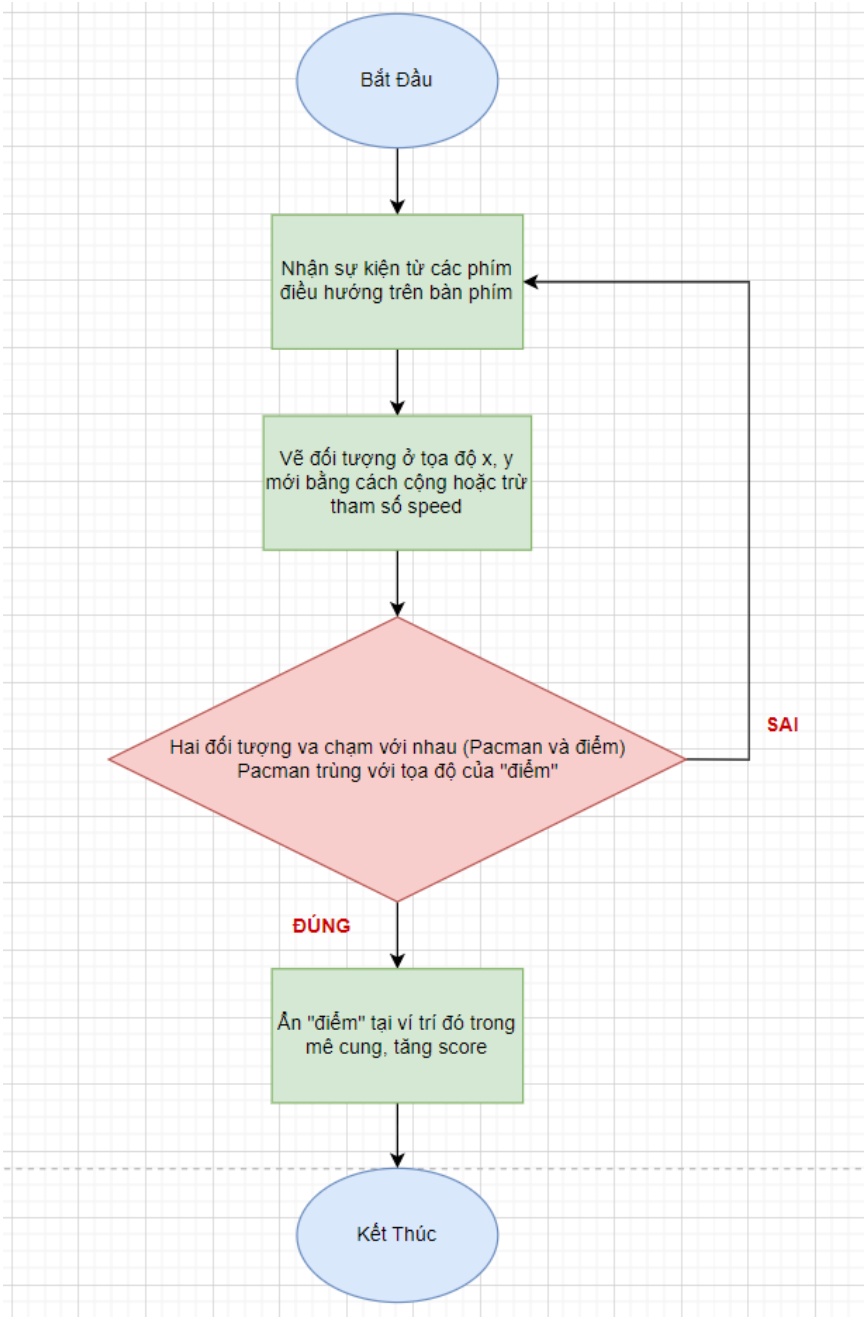
Thuật toán này cũng được sử dụng để con ma nhận biết đâu là tường đâu là đường đi.



Hình 2. 10 Mô tả điều kiện khi Pacman va chạm tường

2.3.3.3 Pacman khi va chạm với điểm:

Xảy ra khi Pacman di chuyển đến tọa độ x, y tiếp theo và ở đó là được xác định tồn tại một "điểm" (viên màu vàng trên bảng đồ) khi đó điểm số của người chơi sẽ được cộng 1 ứng với mỗi chấm Pacman ăn được. Sau khi ăn xong chấm đó sẽ mất đi khi quay lại vị trí đó sẽ không được cộng điểm nữa.



Hình 2. 11 Mô tả điều kiện khi Pacman va chạm với điểm

2.3.3.4 Thuật toán di chuyển của Ghost (Thuật toán random):

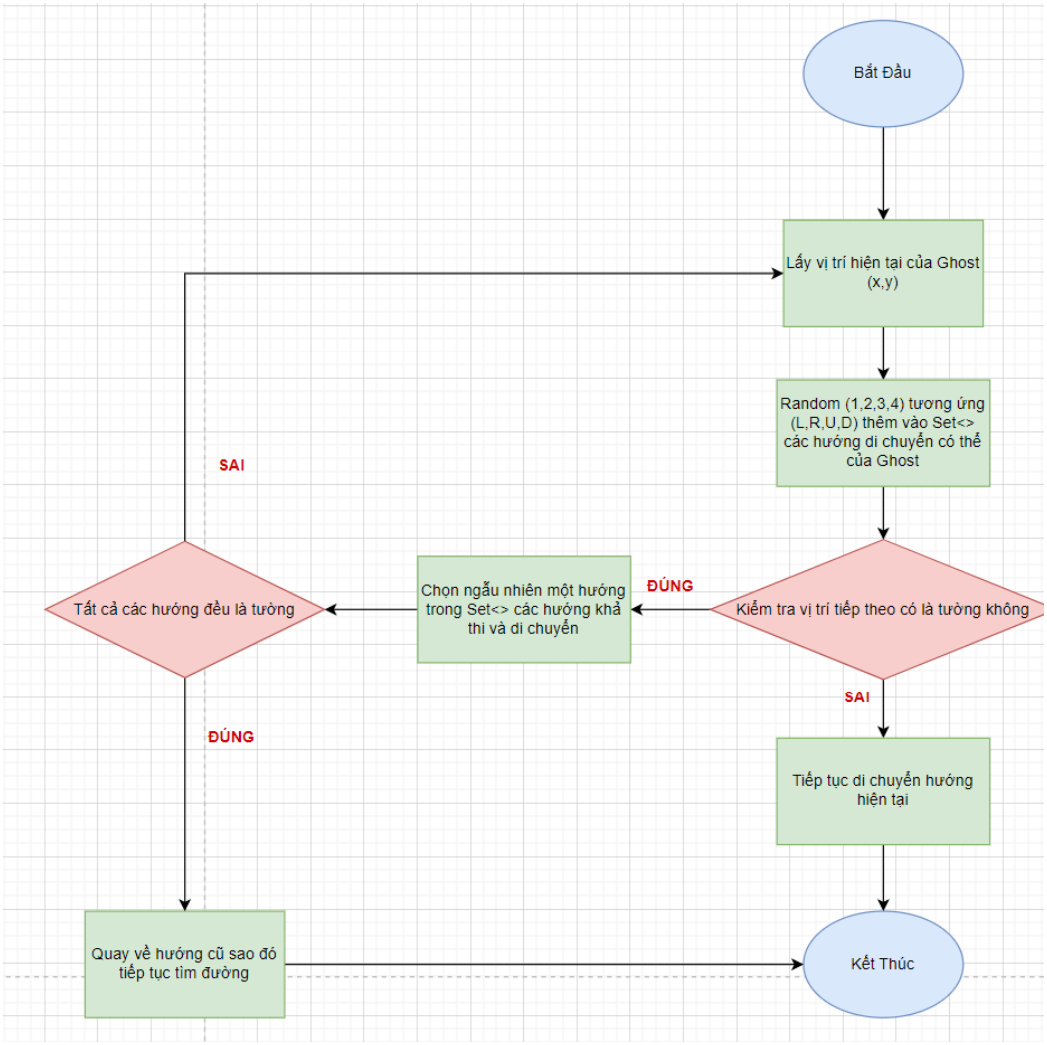
Mô tả: Thuật toán random là thuật toán đơn giản nhất để Ghost có thể tìm đường trong game Pacman mục đích của thuật toán này nhằm random vị trí tiếp theo Ghost sẽ di chuyển trong ma trận cho đến khi va chạm.

Sử dụng một Set<> sử dụng một set để lưu trữ các hướng di chuyển đã thử và chọn ngẫu nhiên một hướng di chuyển khác hướng đối diện với hướng hiện tại của ghost. Nếu ghost không thể đi được trong hướng đã chọn, phương thức sẽ lựa chọn hướng khác cho đến khi tìm được hướng di chuyển hợp lệ. Sau đó, phương thức trả về hướng di chuyển mới tìm được.

Phương thức khác sẽ dùng hướng di chuyển mới để di chuyển ghost một bước. Nếu ghost đang di chuyển theo một hướng và vẫn có thể di chuyển tiếp theo hướng đó thì ghost sẽ tiếp tục di chuyển. Nếu ghost đã đến rìa của một ô vuông và không thể di chuyển tiếp theo hướng đó thì hướng di chuyển mới sẽ được lựa chọn.

Sử dụng thuật toán sinh số ngẫu nhiên để xác định các hướng di chuyển (1,2,3,4) tương ứng với các kí tự chỉ hướng (L,R,U,D) đồng thời thiết lập giá trị speed phù hợp để di chuyển đến vị trí mới, rồi sau từ những hướng ngẫu nhiên đã được tạo chọn tiếp ngẫu nhiên một hướng đã được lưu trong Set<> và di chuyển theo hướng đó.

- Thuật toán được mổ tả như lưu đồ sau:



Hình 2. 12 Mô tả thuật toán tìm đường random

2.4. Giới thiệu ngôn ngữ lập trình Java:

Java là một ngôn ngữ lập trình hướng đối tượng (OOP). Nó là ngôn ngữ lập trình có mục đích chung cho phép các nhà phát triển ứng dụng viết một lần, chạy ở mọi nơi, nghĩa là mã Java đã biên dịch có thể chạy trên tất cả các nền tảng hỗ trợ Java mà không cần biên dịch lại. Thay vì biên dịch mã nguồn thành mã máy hoặc thông dịch mã nguồn khi chạy của phần lớn ngôn ngữ lập trình thông thường, Java được thiết kế để biên dịch mã nguồn thành bytecode, bytecode sau đó sẽ được môi trường thực thi (runtime enviroment) chạy.

Hình 2. 13 Ngôn ngữ lập trình Java

Cú pháp của Java tương tự như C và C++ nhưng có cú pháp hướng đối tượng đơn giản hơn và ít tính năng xử lý cấp thấp hơn. Do đó việc viết một chương trình bằng Java dễ hơn, đơn giản hơn, đỡ tốn công sức để debug hơn. Với thư việc chuẩn KFC, Java chỉ cần mất vài dòng là được, trong khi C phải tốn cả vài chục dòng để thực hiện yêu cầu đề bài với nhiều lỗi dễ xảy ra và khó sửa.

2.4.1. Lịch sử:

James Gosling, Mike Sheridan và Patrick Naughton khởi xướng dự án về ngôn ngữ Java trong tháng 6/1991. Họ được gọi là Green Team.

Thiết kế đầu tiên là cho các hệ thống nhỏ, có thể nhúng vào trong các thiết bị điện tử như set-top box. Ban đầu, nó được gọi là Greentalk bởi James Gosling và với đuôi là .gt.

Sau đó, nó được gọi là Oak và được phát triển như là một phần của Green Project. Oak là loại cây sồi khỏe mạnh, sống nhiều ở các quốc gia như Mỹ, Pháp, Đức, … Bên ngoài khu làm việc của Green Team là các cây sồi tươi tốt quanh năm. Năm 1995, Oak được đổi tên thành Java.

Đổi tên thành Java. Team muốn thu thập để chọn lựa ra một tên mới. Các từ bao gồm dynamic, revolutionary, Silk, jolt, DNA, … Họ muốn cái gì đó mà phản ánh đúng bản chất của công nghệ, đó là: một cuộc cách mạng, có tính động cao, duy nhất, đánh vần dễ dàng, …Theo James Gosling thì Java là một trong các lựa chọn hàng đầu cùng với Silk. Tuy nhiên, vì Java có tính duy nhất hơn, nên hầu như tất cả thành viên team đều lựa chọn Java.

Java là một hòn đảo ở Indonesia, ở nơi này sản phẩm coffee đầu tiên được sản xuất (gọi là java coffee).

Java được phát triển đầu tiên bởi Jamas Gosling tại Sun Microsystems (bât giờ là công ty con của Oracle Coporation) và được công bố năm 1995. Năm 1995, tạp chí Time bình chọn Java là một trong 10 sản phẩm tốt nhất năm 1995.

JDK 1.0 được công bố vào 23/1/1996.

2.4.2. Đặc điểm nổi bật:

- ☐ Máy ảo Java (JVM – Java Virtual Machine).
- ☐ Thông dịch.
- ☐ Độc lập nền.
- ☐ Hướng đối tượng.
- ☐ Đa nhiệm – đa luồng (MultiTasking – Multithreading).
- ☐ Khả chuyển (portable).
- ☐ Hỗ trợ mạnh cho việc phát triển ứng dụng.

2.4.3. Ứng dụng thực tế:

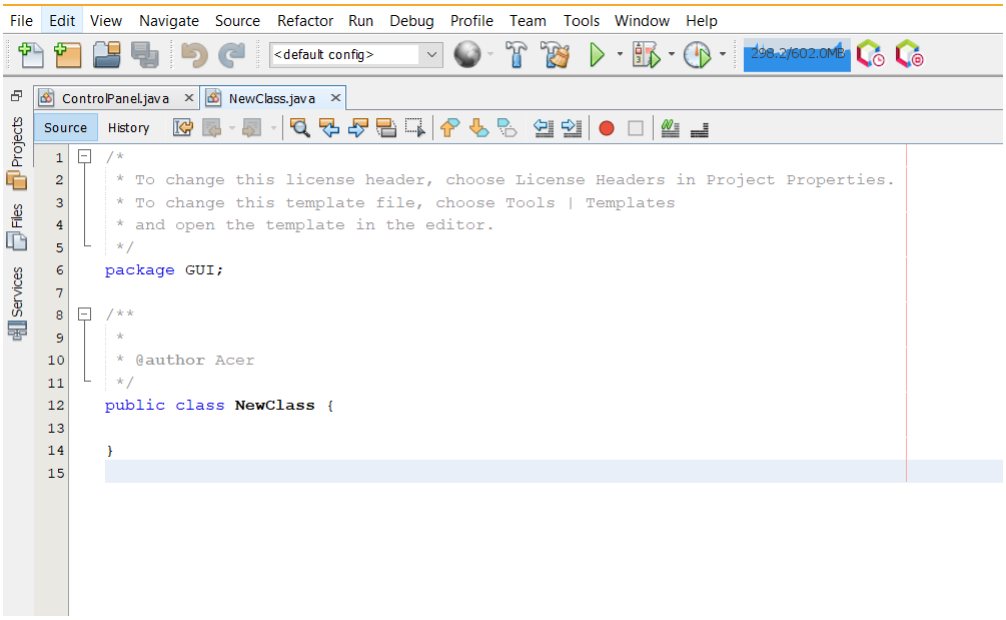
- ☐ Java làm ứng dụng Anroid.
- ☐ Java làm hệ thống giao dịch trong ngành Dịch vụ tài chính.
- ☐ Java làm ứng dụng Web.
- ☐ Java làm phần mềm phát triển.
- ☐ Java làm ứng dụng giao dịch.
- ☐ Java làm ứng dụng J2ME.
- ☐ Java làm lập trình nhúng.
- ☐ Java trong công nghệ Big Data.
- ☐ Java làm hệ thống hiệu suất cao.

☐ Java làm ứng dụng khoa học.

2.5. Tổng quan công cụ sử dụng:

2.5.1. Công cụ Netbeans

NetBeans là một môi trường phát triển tích hợp (IDE) cho Java. NetBeans cho phép các ứng dụng được phát triển từ một tập hợp các thành phần phần mềm được gọi là modules. NetBeans chạy trên Windows, macOS, Linux và Solaris. Ngoài việc phát triển Java, nó còn có các phần mở rộng cho các ngôn ngữ khác như PHP, C, C++, HTML5 và JavaScript. Các ứng dụng dựa trên NetBeans, bao gồm NetBeans IDE, có thể được mở rộng bởi các nhà phát triển bên thứ ba.



Hình 2. 14 Giao diện NetBean 12.3

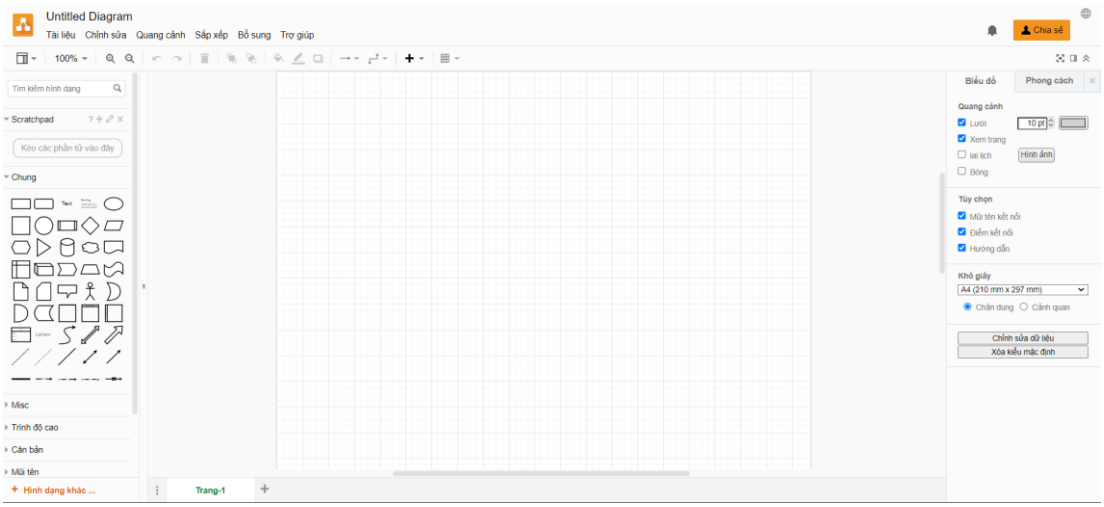
NetBeans IDE hỗ trợ phát triển tất cả các loại ứng dụng Java (Java SE (bao gồm JavaFX), Java ME, web, EJB và ứng dụng mobile). Trong số các tính năng khác là hệ thống dự án dựa trên Ant, hỗ trợ Maven, cải tiến mã nguồn, quản lý phiên bản (hỗ trợ CVS, Subversion, Git, Mercurial và Clearcase).

Qua nhiều phiên bản cũ, nay công cụ lập trình NetBeans IDE đã cập nhật lên phiên bản 8.2 mới nhất có rất nhiều thay đổi và nâng cấp so với các phiên bản cũ hơn có thể kể đến như:

- ☐ Nâng cao ngôn ngữ lập trình C và C++.
- ☐ Nâng cấp, cải tiến các công cụ soạn thảo Profiler và Java.
- ☐ Hỗ trợ Docker, PHP 7.
- ☐ ECMAScript 6 và hỗ trợ thử nghiệm ECMAScript 7.
- ☐ Nâng cấp, cải tiến HTML 5 và Javascript.

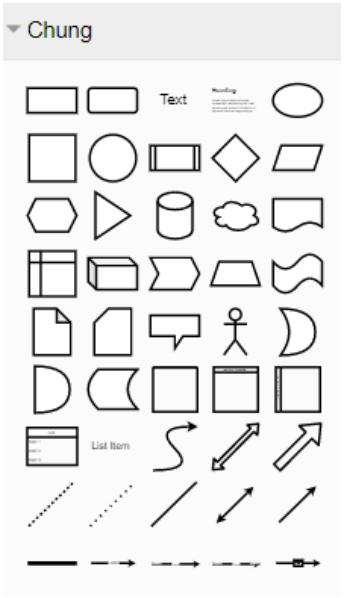
2.5.2. Công cụ vẽ Draw.io

Draw.io là một công cụ vẽ sơ đồ rất mạnh mẽ, hỗ trợ nhiều hình khối, chạy online không cần cài đặt mà lại miễn phí và không bị giới hạn số biểu đồ như nhiều tool vẽ nền web khác. Người dùng có thể vẽ sơ đồ về mạng, điện, phác thảo vị trí các căn phòng trong nhà, hay vẽ các quy trình kinh doanh, vận hành, sản xuất. Cho phép người dùng vẽ hàng tá sơ đồ thiết kế phần mềm, phần cứng và hệ thống.



Hình 2. 15 Giao diện Draw.io

Draw.io có thư viện template rất phong phú để người dùng có thể bắt đầu nhanh hơn, không phải tự mình vẽ lại hết tự đầu.

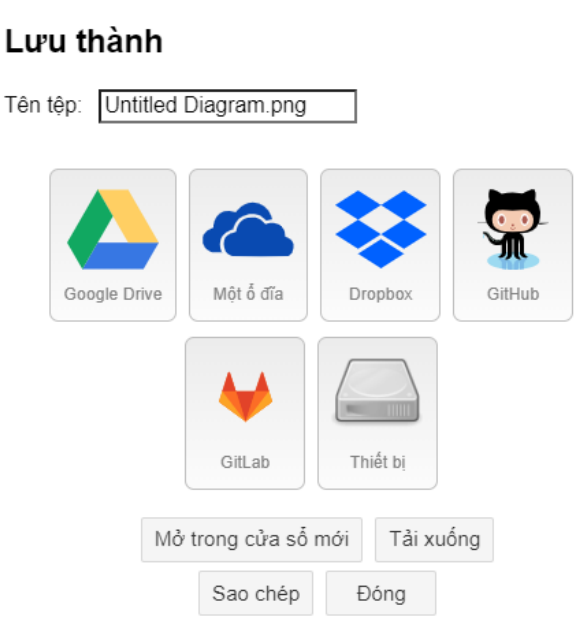


Hình 2. 16 Thư viện template phong phú, đa dạng của Draw.io

Sau khi vẽ xong, người dùng có thể:

- Lưu file draw.io vào Google. Vì file ở trên Drive nên khi nào mở ra cũng có, vô cùng an toàn.

- Download file về máy tính. Hỗ trợ các định dạng hình ảnh, PDF và ảnh vector SVG để người dùng nhúng vào các ứng dụng, tài liệu khác.

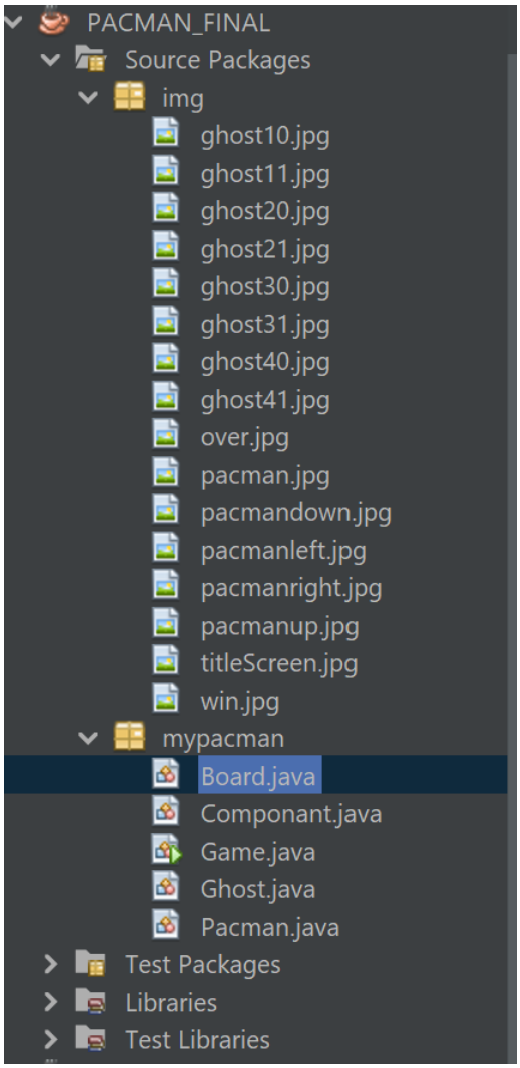


Hình 2. 17 Lưu các file thành nhiều dạng khác nhau tại trang web Draw.io

CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH

3.1. Phân tích cấu trúc dự án:

Dự án sẽ có 2 packages và gồm 5 class có chức năng riêng. Trong đó package mypacman chứa các lớp: Board, Componant, Game, Ghost, Pacman. Package img chứa các file hình ảnh được sử dụng trong game.



Hình 3. 1 Cấu trúc chương trình

Dự án được xây dựng theo mô hình Model (dữ liệu) – Controller (điều khiển) – View (hiển thị). Mô hình này là tiêu chuẩn để tổ chức code một cách thống nhất và mạch lạc. Model là nơi chứa dữ liệu và xử lý các nghiệp vụ thao tác trên dữ liệu đó. View là nơi chỉ thực hiện chức năng hiển thị và gửi yêu cầu

(sự kiện) cho Controller. Controller đóng vai trò trung chuyển, chuyển hoá yêu cầu từ View thành các lệnh yêu cầu Model xử lý dữ liệu, đồng thời chuyển kết quả xử lý từ Model lên cho View hiển thị.

Trong đoạn code này, có thể thấy rằng nó tuân theo mô hình Model-View-Controller (MVC) trong đó:

- Model: Được biểu diễn bởi lớp Board, Pacman và Ghost, đại diện cho trạng thái của trò chơi và tất cả các thành phần của nó như pacman, các điểm trên bảng đồ, các con ma và điểm số.
- View: Được biểu diễn bởi JFrame và Board, đại diện cho giao diện người dùng của trò chơi. JFrame là khung chứa cho Board và được sử dụng để hiển thị các thành phần giao diện người dùng như các label, các hộp thoại. Board là nơi hiển thị tất cả các thành phần của trò chơi như pacman, các điểm, các con ma và điểm số.
- Controller: Được biểu diễn bởi lớp Game và phương thức keyListener. Game là lớp chứa tất cả các logic của trò chơi như di chuyển của pacman và các con ma, phát hiện va chạm và cập nhật trạng thái của các thành phần của trò chơi. keyListener được sử dụng để lắng nghe sự kiện từ bàn phím để thay đổi hướng đi của pacman.

"Board" được khởi tạo ở "Game", và chỉ duy nhất một đối tượng "Board" này tồn tại. Các class "Pacman", "Ghost" chỉ thực hiện chức năng logic như di chuyển, va chạm, không trực tiếp hiển thị lên Frame vì mỗi thực thể sẽ chứa dữ liệu riêng, đồng bộ dữ liệu giữa 3 thực thể riêng biệt cực kỳ phức tạp và dễ dẫn đến sai sót. Vì thế cách tốt nhất là chỉ khởi tạo một đối tượng "Board" duy nhất trong "Game", ở các class khác có yêu cầu gì thì gửi yêu cầu đó cho "Game", "Game" sẽ truy cập trực tiếp với "Board" để thao tác với dữ liệu. Do đó cần sử dụng interface làm lớp giao tiếp chung. Các interface này dùng để tạo ra các bộ lắng nghe sự kiện (Listener) từ các class. Các bộ lắng nghe sự kiện sẽ truyền các giá trị sang các class phát sinh sự kiện. Trong đó class Game chứa

Interface KeyLisener lắng nghe sự kiện từ bàn phím và thực hiện các chức năng trong game.

3.2. Phân tích và xây dựng chương trình

3.2.1. Tạo giao diện:

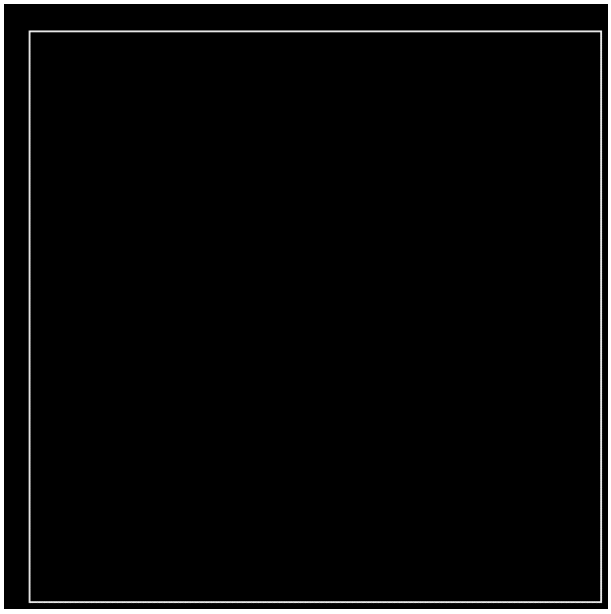
Bước đầu tiên khi thiết kế game Pacman là thiết kế khung nhìn cho game. Việc này sẽ được Class Game và class Board phối hợp đảm nhiệm,... Nó khởi tạo một game Pacman, một đối tượng Board và một JFrame để hiển thị trò chơi. Sau đó, nó thiết lập các thuộc tính của JFrame, bao gồm tiêu đề và thuộc tính đóng JFrame khi người dùng nhấn nút thoát. Nó thêm đối tượng Board vào JFrame và hiển thị JFrame. Cuối cùng, nó yêu cầu đối tượng Board thực hiện các chức năng để thiết lập hoàn chỉnh màn hình game và bắt đầu game.

Ở phía class Board các bước khởi tạo có thể thấy trong phương thức paint như sau:

```
@Override
public void paint (Graphics g){
    g.setColor( c:Color.black);
    g.fillRect( x:0,  y:0,  width:420,  height:500);


    drawBoard(g);
    drawBalls(g);
    drawLives(g);

    Font f = new Font( name:"Arial", style:Font.BOLD, size:20);
    g.setFont( font:f);
    g.drawString("Score: "+score, Componant.max/2+50, Componant.max+30);

    g.drawImage(G_red[ghost1.index],  x:ghost1.x,  y:ghost1.y, observer:null);
    g.drawImage(G_yellow[ghost2.index],  x:ghost2.x,  y:ghost2.y,  observer:null);
    g.drawImage(G_pink[ghost3.index],  x:ghost3.x,  y:ghost3.y,  observer:null);
    g.drawImage(G_blue[ghost4.index],  x:ghost4.x,  y:ghost4.y,  observer:null);

    g.drawImage(Pacman_images[pacman.index], x:pacman.x, y:pacman.y  , observer:null);

    if (title)
        g.drawImage( img:titleScreen,  x:0,  y:0,  observer:null);
    if (lives==0){
        g.drawImage( img:over,  x:0,  y:0,  observer:null);
    }
    if (check())
        g.drawImage( img:win,  x:0,  y:0,  observer:null);
}
```

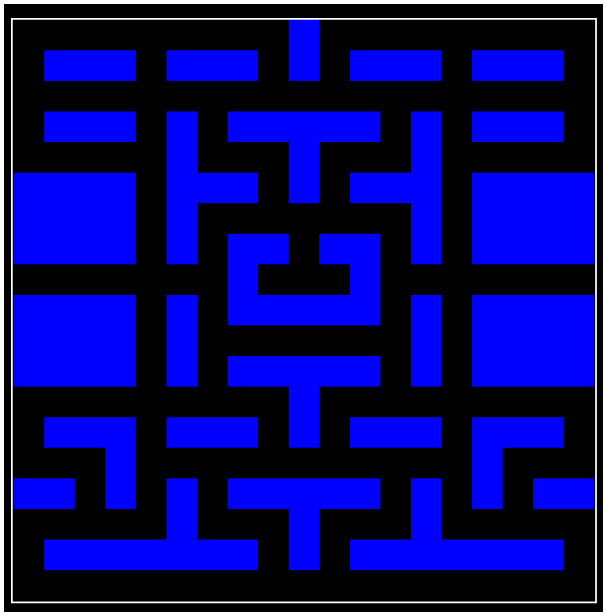
Hình 3. 2 Phương thức paint trong class Board

- Đầu tiên class Game sẽ tô đen toàn bộ nền của Jframe.
- Tiếp theo phương thức paint sẽ gọi phương thức drawBoard để vẽ một viền màu trắng nhằm để người chơi nhận biết khung chơi và để người lập trình xác định được kích thước khung chơi.



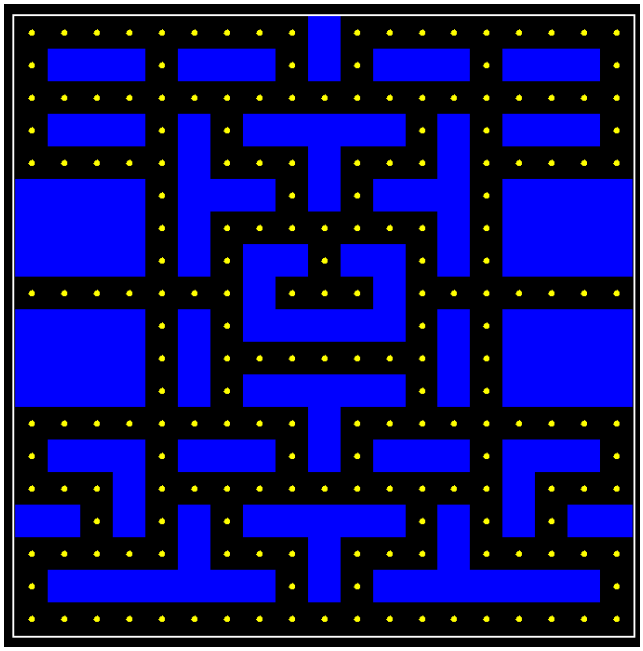
Hình 3. 3 Bước 1 tạo mê cung

- Phương thức drawBoard sẽ tiếp tục vẽ các khối màu xanh với kích thước và vị trí cụ thể đã được xác định trong phương thức drawBoard.



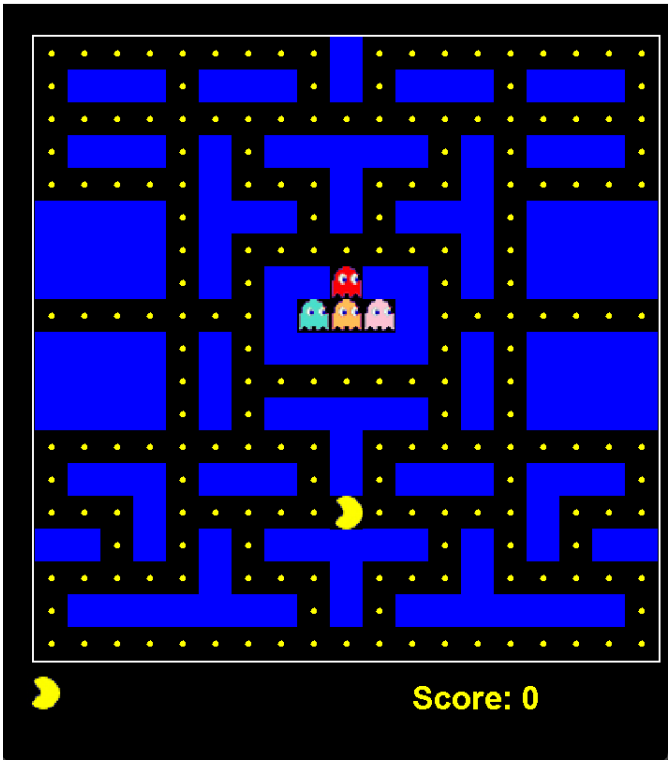
Hình 3. 4 Bước 2 tạo mê cung

- Sau đó phương thức drawBalls sẽ được gọi, phương thức này có nhiệm vụ vẽ các điểm (là thứ Pacman sẽ ăn và tăng điểm số) bằng cách sử dụng 2 vòng lặp xác định tọa độ sau đó tại mỗi tọa độ vẽ một hình oval với kích thước 4x4 pixel.



Hình 3. 5 Bước 3 tạo mê cung

- Cuối cùng sẽ gọi phương thức drawString và drawImage để vẽ điểm số, mạng và các nhân vật tại các vị trí đã được xác lập trên màn hình.



Hình 3. 6 Bước 4 tạo mê cung

3.2.2. Tạo vòng lặp và vẽ các đối tượng lên màn hình:

Sau khi tạo được khung nhìn, mê cung và vẽ được các đối tượng lên màn hình chơi. Bước tiếp theo ta cần tạo một vòng lặp để liên tục cập nhật các hoạt động của các đối tượng lên màn hình chơi.

Đây là một trong những yếu tố quan trọng trong việc lập trình game vì game là được xây dựng giống như việc làm film, là một chuỗi các hoạt động được cập nhật và chiếu lên liên tục từ đó tạo nên hoạt ảnh giúp người chơi thấy nhân vật đang thực sự chuyển động.

```
        timer = new Timer( delay: 60, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (!board.title && board.lives>0) {

                    if (flag) {
                        try {
                            Thread.sleep( millis: 2000);
                            flag = false;
                        } catch (InterruptedException ex) {
                            java.util.logging.Logger.getLogger( name: Game.class.getName()).log( level: Level.SEVERE,
                        }

                    }
                    board.ghost1.move();
                    if(board.ghost1.getShape().intersects( r: board.pacman.getShape())){
                        board.reset();
                    }

                    board.ghost2.move();
                     if(board.ghost2.getShape().intersects( r: board.pacman.getShape())){
                        board.reset();
                    }

                    board.ghost3.move();
                     if(board.ghost3.getShape().intersects( r: board.pacman.getShape())){
                        board.reset();
                    }
```

```
                    board.ghost4.move();
                     if(board.ghost4.getShape().intersects( r: board.pacman.getShape())){
                        board.reset();
                    }

                    board.ghost1.updateState( UpdateStates: board.states);
                    board.ghost2.updateState( UpdateStates: board.states);
                    board.ghost3.updateState( UpdateStates: board.states);
                    board.ghost4.updateState( UpdateStates: board.states);

                    board.pacman.move(direction);
                        if (board.balls[board.pacman.x/20][board.pacman.y/20]) {
                            board.balls[board.pacman.x/20][board.pacman.y/20]= false;
                            board.score++;
                        }

                    board.pacman.updateState( UpdateStates: board.states);
                    }else if (board.lives==0 || board.check()) {
                        try {
                            board.showEnd();
                        } catch (InterruptedException ex) {
                            Logger.getLogger( name: Game.class.getName()).log( level: Level.SEVERE,  msg: null,  thrown: ex);
                        }
                    }
                }
            });
        timer.start();


    }
```

Hình 3. 7 Tạo vòng lặp để vận hành Game bằng Timer

Công việc này được class Game đảm nhiệm thông qua một đối tượng Timer, sau khi ta khởi tạo một JFrame và thêm vào đó một đối tượng của lớp Board (bảng chứa các thành phần của trò chơi). Sau đó, ta tạo một Timer với

tần số là 60 fps (frames per second) và khai báo một ActionListener để xử lý sự kiện của Timer.

Trong phương thức actionPerformed của ActionListener, ta kiểm tra nếu trò chơi chưa kết thúc (board.title == false) và số mạng của Pacman vẫn còn (board.lives > 0), thì ta tiến hành di chuyển các con ma và Pacman, cập nhật trạng thái của chúng và kiểm tra va chạm giữa chúng. Nếu Pacman ăn được điểm, ta cập nhật điểm số.

Nếu trò chơi kết thúc (board.lives == 0 hoặc đã ăn hết các hạt), ta gọi phương thức showEnd của đối tượng Board để hiển thị thông báo kết thúc trò chơi. Trong trường hợp này, Timer sẽ không tiếp tục chạy nữa.

Công việc này được thực hiện liên tục thông qua phương thức repaint () trong mỗi 1/60 giây.

3.2.3. Xây dựng class Componant – nền tảng cho các đối tượng trong game:

Ứng dụng OOP vào việc thiết kế game là một điều tất yếu khi sử dụng ngôn ngữ Java vì vậy để các đối tượng có thể sử dụng các phương thức chung chúng ta cần xây dựng một lớp cha chứa các phương thức để các lớp con kế thừa và sử dụng.

Class Componant này là một phần trong trò chơi Pac-Man, là một lớp abstract nó chứa các thông tin về Pac-Man và mê cung, cung cấp các phương thức để xử lý và kiểm tra việc di chuyển của Pac-Man trên mê cung.

Thuộc tính:

- index: là một số nguyên, được khởi tạo mặc định là 0. Dùng để các class con lưu trữ chỉ số hình ảnh trong sprite sheet từ đó lấy đc hình ảnh phù hợp với hướng di chuyển.
- states: là một mảng 2 chiều boolean kích thước 20x20, mỗi phần tử của mảng này thể hiện trạng thái của một ô trong mê cung Pac-Man. Nếu giá

trị của phần tử là true, thì ô đó có thể đi qua, ngược lại nếu giá trị của phần tử là false thì ô đó không thể đi qua. Đây là thông tin quan trọng để Pac-Man có thể di chuyển đúng trên mê cung.

- cellSize: là một số nguyên tĩnh bằng 20, thể hiện kích thước của một ô trong mê cung.
- max: là một số nguyên tĩnh bằng 400, thể hiện kích thước của mê cung (20 ô x 20 ô x kích thước ô). Đây là giá trị tối đa của tọa độ x hoặc y mà Pac-Man có thể đạt được trên mê cung.
- speed: là một số nguyên tĩnh bằng 4, thể hiện tốc độ di chuyển của Pac-Man trên mê cung.
- direction: là một ký tự, thể hiện hướng di chuyển của Pac-Man. Ký tự này có thể là 'u' (lên), 'd' (xuống), 'l' (qua trái), 'r' (qua phải).
- x, y: hai số nguyên, thể hiện tọa độ của Pac-Man trên mê cung. Tọa độ này được tính bằng số ô (cell) trên trục x hoặc y, ví dụ: tọa độ (x=2, y=3) thể hiện Pac-Man đang ở ô thứ hai trên trục x và ô thứ ba trên trục y.

Phương thức:

- Componant(): là hàm tạo của lớp, khởi tạo giá trị mặc định cho mảng states. Mỗi phần tử trong mảng được khởi tạo là false, tức là không có ô nào trong mê cung có thể đi qua ban đầu.
- updateState(boolean [][] UpdateStates): là phương thức để cập nhật trạng thái của m ê cung Pac-Man. Phương thức này nhận vào một mảng 2 chiều boolean UpdateStates, thể hiện trạng thái mới của các ô trong mê cung. Phương thức sẽ cập nhật mảng states của Pac-Man bằng cách gán giá trị của từng phần tử trong UpdateStates cho phần tử tương ứng trong states.
- isValid(int x, int y): là phương thức để kiểm tra xem vị trí mới của Pac-Man có hợp lệ không. Phương thức này nhận vào 2 tham số là x và y, thể hiện tọa độ mới của Pac-Man. Nếu tọa độ mới nằm trong mê cung và ô

tại tọa độ đó có thể đi qua thì phương thức trả về giá trị true, ngược lại trả về giá trị false.

- getShape(): là phương thức để lấy hình dạng của Pac-Man, trả về một đối tượng Rectangle có kích thước 20x20 và tọa độ x, y của Pac-Man. Đối tượng Rectangle này được sử dụng để vẽ Pac-Man lên màn hình.

Các đối tượng Pacman và Ghost được kế thừa từ class Componant để sử dụng các thuộc tính và phương thức này. Điều này giúp giảm thiểu việc lặp lại mã và tăng tính tái sử dụng của code. Dễ bảo trì và nâng cấp chương trình sau này.

Các Class extend class này là Pacman và Ghost.

3.2.4. Nhân vật PacMan:

Class Pacman là một lớp đại diện cho nhân vật Pacman trong trò chơi Pacman. Nó kế thừa từ lớp Component, do đó nó có thể truy cập và sử dụng các thuộc tính và phương thức được định nghĩa trong lớp Component.

Các thuộc tính của Pacman bao gồm:

- int x: tọa độ x của Pacman trên màn hình.
- int y: tọa độ y của Pacman trên màn hình.
- Phương thức khởi tạo Pacman được định nghĩa để khởi tạo giá trị ban đầu của Pacman với các giá trị x và y được truyền vào.
- Phương thức move(char direction) được định nghĩa để di chuyển Pacman dựa trên hướng di chuyển được truyền vào. Hướng di chuyển được truyền vào dưới dạng một ký tự, có thể là 'L' (trái), 'R' (phải), 'U' (lên) hoặc 'D' (xuống).

Trong phương thức move, switch-case được sử dụng để xác định hướng di chuyển và thực hiện việc di chuyển tương ứng. Nếu hướng di chuyển là 'L' hoặc 'R', Pacman sẽ di chuyển sang trái hoặc phải, tương ứng với giá trị của biến

x. Nếu hướng di chuyển là 'U' hoặc 'D', Pacman sẽ di chuyển lên hoặc xuống, tương ứng với giá trị của biến y.

```
public void move(char direction){

    switch (direction) {
        case 'L':
            if (isValid(x-speed, y) && isValid(x-speed, y+cellSize-5)) {
                x-=speed;
            }else if (y > 178 && y<182 && x== 20) {
                x= 380;
            }
            index= 0;
            break;
        case 'R':
            if (isValid(x+cellSize, y) && isValid(x+cellSize, y+cellSize-5)) {
                x+=speed;
            }else if (y > 178 && y<182 && x== 20) {
                x= 20;
            }
            index = 1;
            break;
        case 'U':
            if (isValid(x, y-speed) && isValid(x+cellSize-5, y-speed)) {
                y-=speed;
            }
            index=2;
            break;
        case 'D':
            if (isValid(x, y+cellSize) && isValid(x+cellSize-5, y+cellSize)) {
                y+=speed;
            }
            index = 3;
            break;
    }
}
```

Hình 3. 8 Phương thức move.

Để kiểm tra xem Pacman có thể di chuyển theo hướng đã chọn hay không, phương thức isValid() được sử dụng. Nếu Pacman đang ở giữa một ô, có nghĩa là x và y là bội số của cellSize (kích thước của một ô), Pacman có thể di chuyển đến ô tiếp theo. Nếu Pacman gặp phải tường, nó sẽ không thể di chuyển.

Ngoài ra, nếu Pacman chạm vào một trong các vật phẩm trên màn hình, nó sẽ thay đổi hướng di chuyển. Để làm điều này, giá trị index được sử dụng để xác định hướng di chuyển mới. Nếu index là 0, Pacman sẽ đi sang trái; nếu

index là 1, Pacman sẽ đi sang phải; nếu index là 2, Pacman sẽ đi lên; nếu index là 3, Pacman sẽ đi xuống.

Trong phương thức move của lớp Pacman, nếu Pacman đi đến đầu đường đi (góc trái trên cùng của màn hình), thì sẽ có một điều kiện kiểm tra để kiểm tra xem Pacman có nằm trên đúng vị trí góc trên cùng của màn hình hay không:

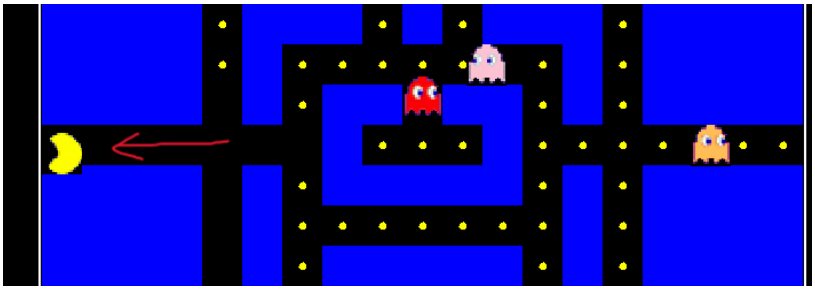
```
}else if (y > 178 && y<182 && x== 20) {
    x= 380;
}
```

Hình 3. 9 Code pacman đi xuyên bản đồ từ bên trái

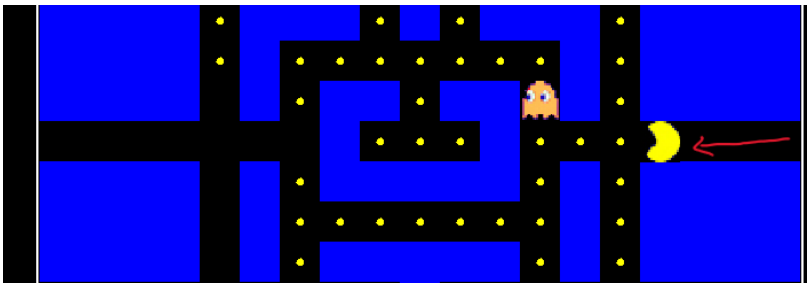
```
}else if (y > 178 && y<182 && x== 380) {
    x= 20;
}
```

Hình 3. 10 Code pacman đi xuyên bản đồ từ bên phải

Nếu Pacman đúng vị trí góc giữa màn hình (y > 178 và y < 182) và x bằng 20 (đây là tọa độ x của đầu đường đi), thì vị trí của Pacman sẽ được thiết lập lại ở phía bên phải của màn hình (x = 380). Nếu Pacman tiếp tục đi qua đầu đường đi từ bên phải sang bên trái, thì tọa độ x của nó sẽ trở lại 20. Việc này tạo ra cảm giác như Pacman đã vượt qua đầu đường đi và tiếp tục di chuyển trong mê cung.



Hình 3. 11 Trong game Pacman đi xuyên tường từ bên trái



Hình 3. 12 Sau đó xuất hiện từ cổng bên phải

3.2.5. Kẻ thù Ghost:

Lớp Ghost là một đối tượng trong trò chơi Pacman, đại diện cho các ma trong trò chơi. Lớp này có các phương thức và thuộc tính để điều khiển hành động của ma trong trò chơi.

Phương thức khởi tạo Ghost(int x, int y) được sử dụng để tạo một đối tượng Ghost mới tại vị trí xác định bởi các tham số x và y. Thuộc tính x và y của đối tượng Ghost sẽ được khởi tạo với giá trị tương ứng với x và y, và hướng của ma sẽ được khởi tạo với giá trị 'L' (trái).

Phương thức choice() sẽ kiểm tra xem ma có đang ở tại một ô chính xác hay không. Nếu ma đang ở tại một ô chính xác, phương thức sẽ trả về true, ngược lại trả về false.

Phương thức selectDirection() được sử dụng để chọn một hướng di chuyển mới cho ma. Đầu tiên, phương thức sẽ tính toán hướng đối diện với hướng hiện tại của ma (biến backwards). Sau đó, phương thức sẽ lặp lại việc chọn hướng mới cho đến khi tìm thấy một hướng hợp lệ khác hướng đối diện và có thể được di chuyển đến. Trong quá trình này, ma sẽ không di chuyển và chỉ chọn hướng mới. Nếu ma đã chọn ba hướng không hợp lệ liên tiếp, phương

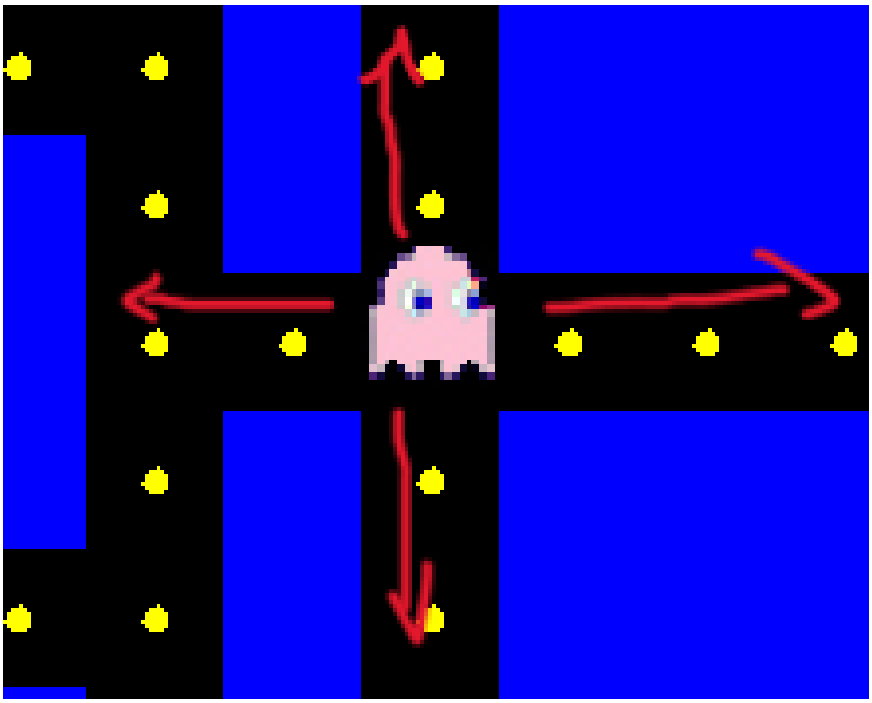
thức sẽ xóa tập hợp chứa các hướng đã chọn và chọn lại hướng đối diện để thử lại.

```
public char selectDirection(){

    int random;
    int newx=x,newy=y;
    Set<Character> mySet = new HashSet<Character>();

    char backwards = 'R';

    switch (direction) {
        case 'L':
            backwards='R';
            break;
        case 'R':
            backwards='L';
            break;
        case 'U':
            backwards='D';
            break;
        case 'D':
            backwards='U';
            break;
    }
    char newDirection = backwards;
    while (newDirection==backwards || !isValid( x:newx,  y:newy)) {

        if (mySet.size() == 3) {
            mySet.clear();
            newDirection=backwards;
            break;
        }
        random = (int)(Math.random() *4) + 1;
```

```
        if (random == 1) {
            newDirection = 'L';
            newx-=speed;
        }else if (random == 2) {
            newDirection='R';
            newx+=cellSize;
        }else if (random == 3) {
            newDirection='U';
            newy-=speed;
        }else if (random == 4) {
            newDirection='D';
            newy+=cellSize;
        }

        if (newDirection != backwards) {
            mySet.add( e:newDirection);
        }

        index = random%2;
    }
    return newDirection;
```

Hình 3. 13 Phương thức SelectDirection

Phương thức move() được sử dụng để di chuyển ma theo hướng đã chọn. Trước tiên, phương thức sẽ kiểm tra xem ma có đang ở tại một ô chính xác hay không bằng cách gọi phương thức choice(). Nếu ma đang ở tại một ô chính xác, phương thức sẽ chọn một hướng di chuyển mới cho ma bằng cách gọi phương thức selectDirection(). Sau đó, phương thức sẽ di chuyển ma theo hướng đã chọn bằng cách thay đổi giá trị của thuộc tính x hoặc y tùy theo hướng di chuyển.



Hình 3. 14 Ghost có thể di chuyển theo hướng mũi tên

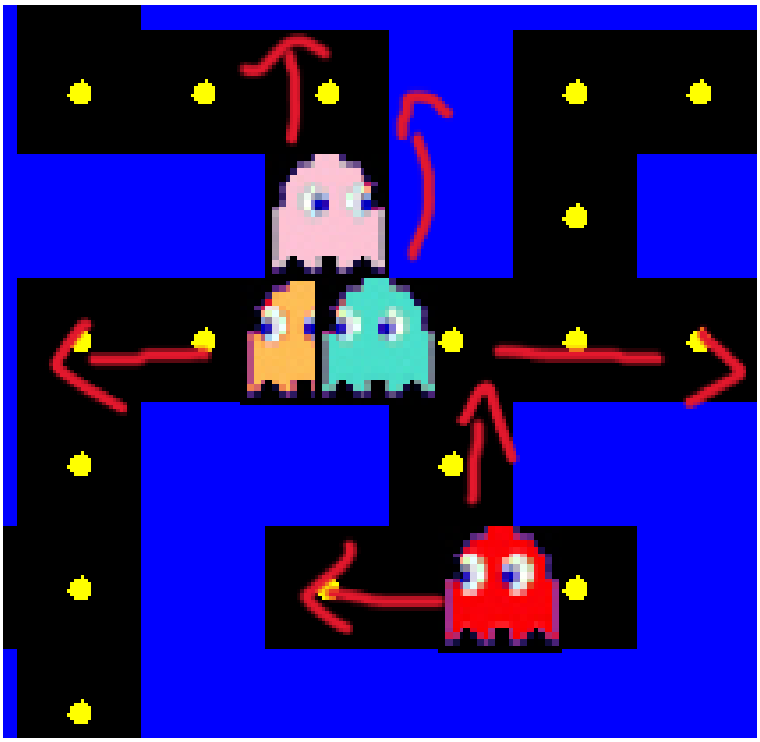
Phương thức isValid(int x, int y) được sử dụng để kiểm tra xem vị trí (x, y) có hợp lệ để di chuyển đến hay không. Nếu vị trí (x, y) thuộc một trong các ô không thể di chuyển trong trò chơi (như tường, gạch chắn) hoặc nếu vị trí đó trùng với vị trí của một trong số các ma khác, phương thức sẽ trả về false, ngược lại nếu vị trí (x, y) hợp lệ để di chuyển, phương thức sẽ trả về true.

Phương thức selectDirection() là phương thức quan trọng trong lớp Ghost, nó sẽ chọn hướng di chuyển mới cho ma dựa trên hướng di chuyển hiện tại và vị trí của Pacman. Đầu tiên, phương thức sẽ xác định hướng đi ngược lại so với hướng hiện tại và lưu vào biến backwards. Sau đó, phương thức sử dụng một vòng lặp while để tìm kiếm một hướng đi mới hợp lệ cho ma.

Trong vòng lặp while, phương thức sẽ chọn ngẫu nhiên một hướng đi mới. Nếu hướng đi này không phải là hướng đi ngược lại, nó được thêm vào một Set để tránh ma di chuyển quá nhiều lần theo cùng một hướng. Nếu Set chứa 3 hướng đi, nó sẽ được xóa và hướng đi ngược lại được chọn.

Phương thức choice() sẽ trả về true nếu ma đang ở đầu đường đi, nghĩa là nó có thể chọn một hướng đi mới. Nếu không, phương thức sẽ trả về false và ma sẽ tiếp tục di chuyển theo hướng hiện tại.

Phương thức move() được sử dụng để di chuyển ma đến vị trí mới. Nếu ma ở đầu đường đi, phương thức sẽ chọn hướng đi mới bằng cách gọi phương thức selectDirection(). Sau đó, phương thức sử dụng một câu lệnh switch-case để di chuyển ma theo hướng mới. Nếu vị trí mới hợp lệ, ma sẽ di chuyển đến vị trí đó bằng cách cập nhật các giá trị x và y của nó. Nếu không, ma sẽ tiếp tục di chuyển theo hướng hiện tại.

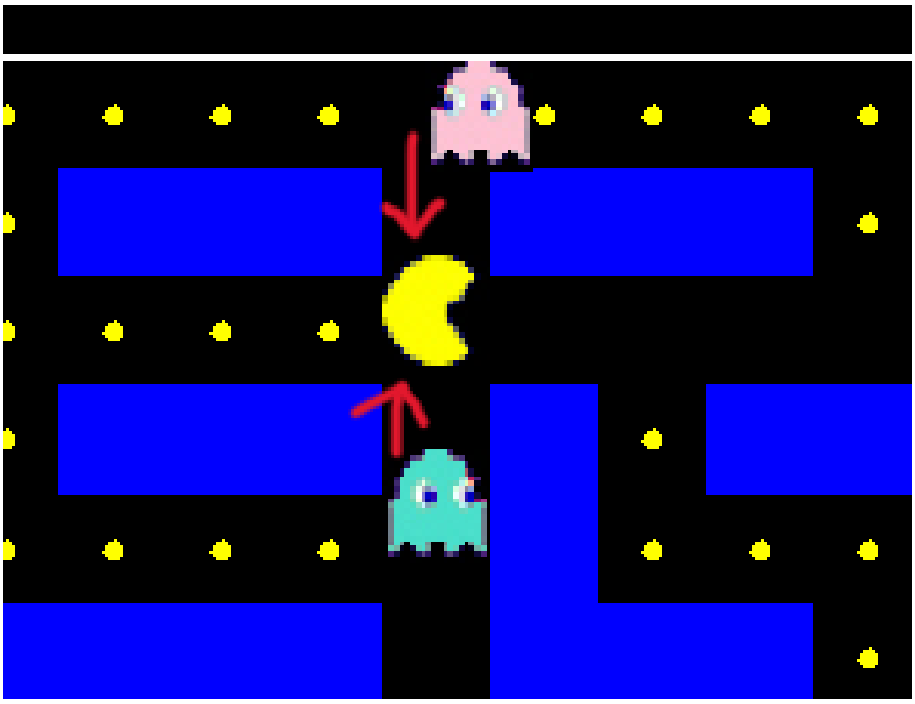


Hình 3. 15 Ghost cố gắng tìm đường mới để di chuyển hạn chế đường cũ

Trên cơ sở đó, lớp Ghost đã có thể thực hiện được chức năng di chuyển và lựa chọn hướng đi của ma trong trò chơi Pacman.

3.2.6. Va chạm giữa Pacman và Ma:

Việc xử lý va chạm giữa Pacman và ma (Ghost) là một trong những chức năng quan trọng trong game Pacman. Khi Pacman và Ghost gặp nhau trên cùng một ô vuông (cell) trên mê cung, ta cần phải thực hiện các xử lý để điều khiển game phù hợp.



Hình 3. 16 Pacman bị mất

Trong class Ghost, ta có phương thức move() để di chuyển ma trên mê cung, phương thức selectDirection() để chọn hướng di chuyển tiếp theo dựa trên các điều kiện được thiết lập . Ta sử dụng phương thức isValid() để kiểm tra xem ô vuông tiếp theo có hợp lệ để di chuyển hay không.

Trong class Pacman, ta cũng có phương thức move() để di chuyển Pacman trên mê cung, với đầu vào là hướng di chuyển (direction) được xác định bởi người chơi. Ta cũng sử dụng phương thức isValid() để kiểm tra tính hợp lệ của ô vuông tiếp theo.

Sự kiện va chạm giữa Pacman và Ghost do class Game đảm nhiệm.

Khi Pacman và Ghost cùng đến một ô vuông, ta sử dụng phương thức intersect() để kiểm tra xem chúng có va chạm hay không. Nếu có va chạm, ta thực hiện một số xử lý như hiển thị thông báo thua cuộc, dừng game hoặc cộng điểm tùy thuộc vào điều kiện được thiết lập.

```
board.ghost1.move();
if(board.ghost1.getShape().intersects( r:board.pacman.getShape())){
    board.reset();
}

board.ghost2.move();
 if(board.ghost2.getShape().intersects( r:board.pacman.getShape())){
    board.reset();
}

board.ghost3.move();
 if(board.ghost3.getShape().intersects( r:board.pacman.getShape())){
    board.reset();
}

board.ghost4.move();
 if(board.ghost4.getShape().intersects( r:board.pacman.getShape())){
    board.reset();
}

board.ghost1.updateState( UpdateStates:board.states);
board.ghost2.updateState( UpdateStates:board.states);
board.ghost3.updateState( UpdateStates:board.states);
board.ghost4.updateState( UpdateStates:board.states);
```

Hình 3. 17 Code xử lí va chạm Pacman và Ghost

Trong đoạn mã trên, các đối tượng Ghost và Pacman được di chuyển bằng cách gọi phương thức move của chúng, sau đó kiểm tra nếu hình dạng (shape) của một trong số chúng intersect với hình dạng của đối tượng Pacman, thì phương thức reset sẽ được gọi để khởi động lại trò chơi.

Cụ thể, trong mỗi vòng lặp của game loop, Ghost1, Ghost2, Ghost3, Ghost4 đều được di chuyển bằng cách gọi phương thức move () của chúng, sau đó được kiểm tra để xem liệu chúng có va chạm với Pacman hay không. Nếu hình dạng (shape) của bất kỳ Ghost nào intersect với hình dạng của Pacman, phương thức reset sẽ được gọi để khởi động lại trò chơi.

Sau khi kiểm tra va chạm, tất cả các đối tượng Ghost đều được cập nhật lại trạng thái của chúng bằng cách gọi phương thức updateState ().

Số mạng (lives) của người chơi sẽ giảm 1 đến khi về bằng 0 phương thức xử lí checkEnd sẽ vẽ lên màn hình thông báo thắng thua cùng hộp thoại thông báo điểm và các tùy chọn chơi tiếp hay thoát game.

Tóm lại, việc xử lý va chạm giữa Pacman và Ghost là một trong những bước quan trọng để thực hiện trong game Pacman. Bằng cách sử dụng các phương thức và điều kiện được thiết lập, ta có thể đảm bảo tính chính xác và logic của game, đồng thời tạo ra trải nghiệm chơi game thú vị và thử thách cho người chơi.

3.2.7. Tăng điểm:

Trong game Pacman, mục tiêu của người chơi là điều khiển nhân vật Pacman ăn hết tất cả các viên bánh trên mê cung để tăng điểm số. Trong code của bạn, việc tăng điểm số khi Pacman ăn được các chấm trên mê cung được thực hiện như sau:

```
board.pacman.move(direction);
    if (board.balls[board.pacman.x/20][board.pacman.y/20]) {
        board.balls[board.pacman.x/20][board.pacman.y/20]= false;
        board.score++;
    }
```

Hình 3. 18 Code cơ chế tăng điểm game Pacman

Trong đó, board. balls là một mảng 2 chiều lưu trữ trạng thái của các viên bánh trên mê cung. Mỗi phần tử của mảng có giá trị là true nếu ở vị trí tương ứng có một viên bánh chưa được ăn, và false nếu đã bị ăn hoặc không có viên bánh nào ở vị trí đó.

Khi Pacman di chuyển đến một ô trên mê cung, ta kiểm tra xem ở vị trí đó có chứa viên bánh hay không bằng cách truy cập vào phần tử tương ứng của mảng board. balls. Nếu tìm thấy một viên bánh chưa được ăn ở vị trí đó, ta đặt giá trị của phần tử tương ứng trong mảng board. balls thành false (đánh dấu viên bánh đã bị ăn), và tăng điểm số board. score lên 1 đơn vị.

Cơ chế này cho phép người chơi có thể kiểm soát được việc ăn các viên bánh trên mê cung và tính toán điểm số của mình. Việc tăng điểm số cũng giúp tạo động lực cho người chơi hoàn thành một màn chơi và cố gắng đạt được điểm số cao hơn ở những màn chơi tiếp theo.

3.2.8. Xử lí End Game:

Trong code của bạn, khi Pacman ăn hết tất cả các điểm trên mê cung, game sẽ kết thúc và người chơi sẽ thắng. Tương tự, nếu Pacman hết mạng, game sẽ kết thúc và người chơi sẽ thua.

Đoạn code để xử lý việc kết thúc game nằm trong vòng lặp while trong phương thức run của class Game. Nếu Pacman ăn hết tất cả các chấm trên mê cung hoặc hết mạng, điều kiện if (board.lives==0 || board.check()) sẽ trả về true và game sẽ kết thúc.

Trong trường hợp Pacman ăn hết tất cả các chấm trên mê cung, điều kiện board.check() sẽ trả về true và người chơi sẽ thắng. Khi điều kiện này được thỏa mãn, phương thức showEnd sẽ được gọi để hiển thị thông báo người chơi thắng. Nếu Pacman hết mạng, điều kiện board.lives==0 sẽ trả về true và phương thức showEnd cũng sẽ được gọi để hiển thị thông báo người chơi thua.

Để xử lý việc kết thúc game và hiển thị thông báo, trong phương thức showEnd, trạng thái game sẽ được đặt lại và thông báo người chơi thắng hoặc thua sẽ được hiển thị bằng cách sử dụng hộp thoại đơn giản JOptionPane.

Dưới đây là đoạn code để xử lý việc kết thúc game:

```
}else if (board.lives==0 || board.check()) {
    try {
        board.showEnd();
    } catch (InterruptedException ex) {
        Logger.getLogger( name:Game.class.getName()).log( level:Level.SEVERE,  msg:null,  thrown:ex);
    }
}
```

Hình 3. 19 Code xử lí khi End Game

Trong đó:

- board. lives == 0 kiểm tra xem Pacman còn mạng không. Nếu Pacman không còn mạng, điều kiện này sẽ trả về true.
- board. Check () kiểm tra xem các điểm trên mê cung đã được ăn hết chưa. Nếu các điểm đã được ăn hết, điều kiện này sẽ trả về true.
- board. showEnd () hiển thị thông báo người chơi thắng hoặc thua bằng cách gọi một hộp thoại bằng phương thức Joption.showOptionDialog thông báo số điểm bằng việc nhận tham số score. Đưa ra hai lựa chọn là "Ván nữa" – khi đó các thông số như mạng, số điểm trên bản đồ, score, ... đều sẽ được reset để bắt đầu game mới. Hoặc chọn "Nghỉ game" để thoát qua phương thức System.exit(0);

```
public void showEnd() throws InterruptedException{
    Thread.sleep( millis: 1000);
                int option = JOptionPane.showOptionDialog( parentComponent: null, "Điểm của bạn là :"+score,  title: "Chà
                 optionType: JOptionPane.DEFAULT_OPTION,  messageType: JOptionPane.PLAIN_MESSAGE,  icon: null,
                new Object[] { "Ván nữa", "Nghỉ Game" },  initialValue: "Ván nữa");
                if (option == 0) {
                    lives = 3;
                    score = 0;
                    init();
                } else {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Hẹn gặp lại người anh em !!!");
                    System.exit( status: 0);
                }
}
```

Hình 3. 20 Phương thức showEnd thực hiện các hành động khi end game

CHƯƠNG 4: HIỆN THỰC CHƯƠNG TRÌNH

4.1. Màn Hình Chờ:

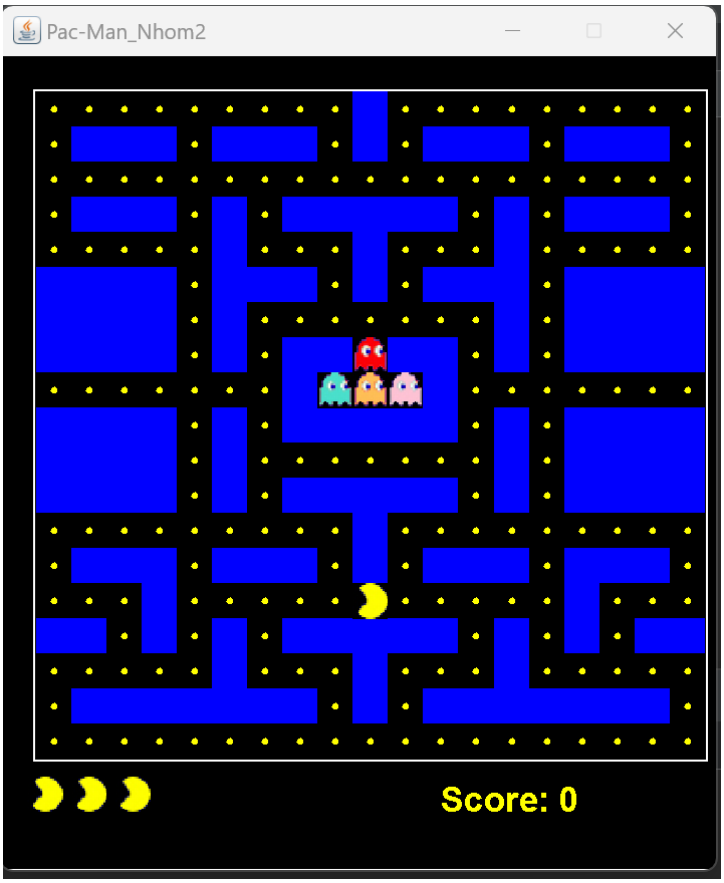
Hiển thị tên game, các thông tin liên quan, hướng dẫn chơi game.

Người chơi nhấn ENTER để bắt đầu game.

Hình 4. 1 Màn hình chờ

4.2. Bắt đầu trò chơi:

Khi người dùng nhấn chạy chương trình game sẽ được khởi tạo, bạn hãy đọc hướng dẫn được hiện lên trên màn hình để hiểu về cách chơi game sau đó nhấn ENTER để bắt đầu trò chơi. Mặc định bạn sẽ có 3 mạng và 0 điểm khi bắt đầu game.



Hình 4. 2 Màn hình bắt đầu game

Sau khi ấn ENTER game sẽ bắt đầu, nhiệm vụ của bạn là cố gắng sống sót và ăn nhiều điểm nhất có thể mỗi khi ăn được một chấm vàng trên màn hình số điểm của bạn sẽ được cộng 1.

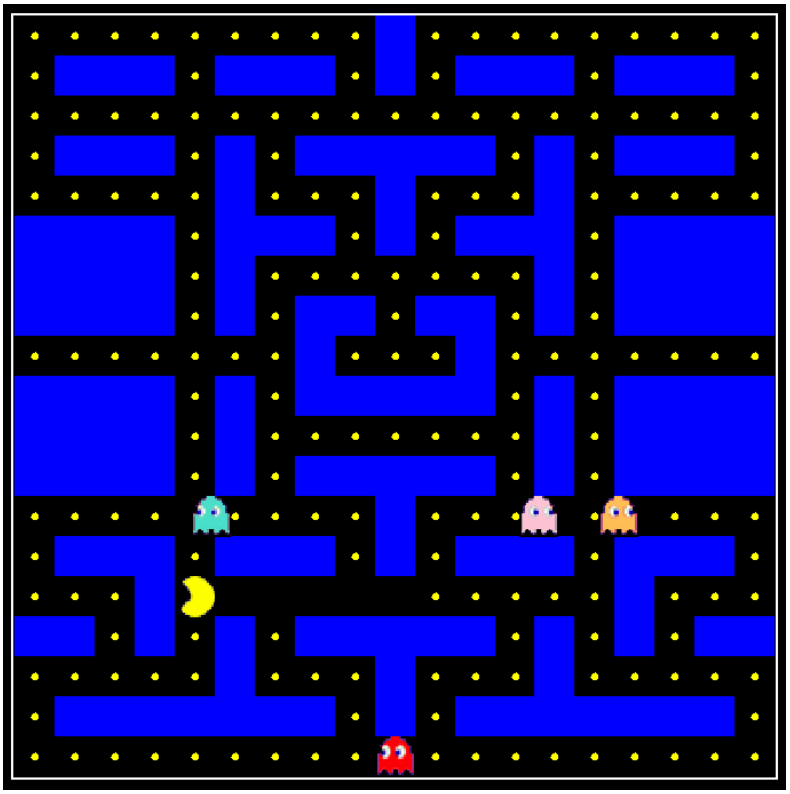
Cố gắng né các con ma đang đi tuần tra trên bản đồ đang tìm kiếm bạn, nếu bạn va chạm với 1 trong số những con ma trên bản đồ bạn sẽ mất 1 mạng và sẽ được đưa về vị trí bắt đầu.

4.3. Chơi trò chơi:

Một số lưu ý và hướng dẫn chơi game nhanh:

- Sử dụng các phím điều hướng để di chuyển Pacman ăn các chấm trên bản đồ theo các lối màu đen.
- Các ô màu xanh dương là tường và bạn không thể vượt qua chúng.

- Chạm vào những con ma là điều cấm kị vì làm thế bạn sẽ hy sinh.
- Đường đi nằm ngang giữa màn hình là một con đường thông nhau và bạn có thể đi từ đầu bên này sang đầu bên kia.



Hình 4. 3 Màn hình chơi game.

4.4. Thắng trò chơi:

Khi người chơi ăn hết các chấm trên mê cung mà vẫn còn mạng người chơi sẽ chiến thắng game này.

Hình 4. 4 Màn hình chiến thắng

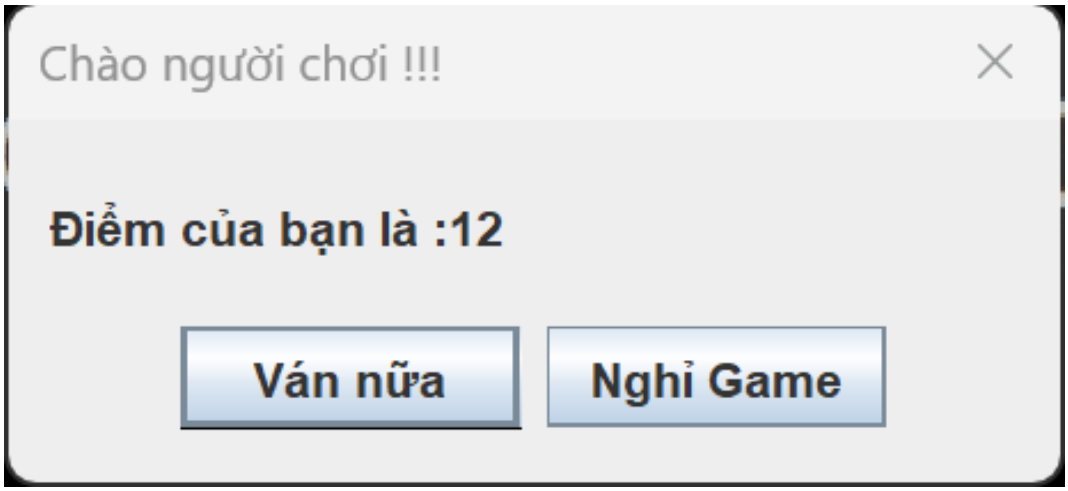
4.5. Thua trò chơi:

Khi người chơi bị kẻ thù ăn hết mạng (lives = 0), màn hình sẽ hiện lên thông báo bạn đã thua game này.

Hình 4. 5 Game Over

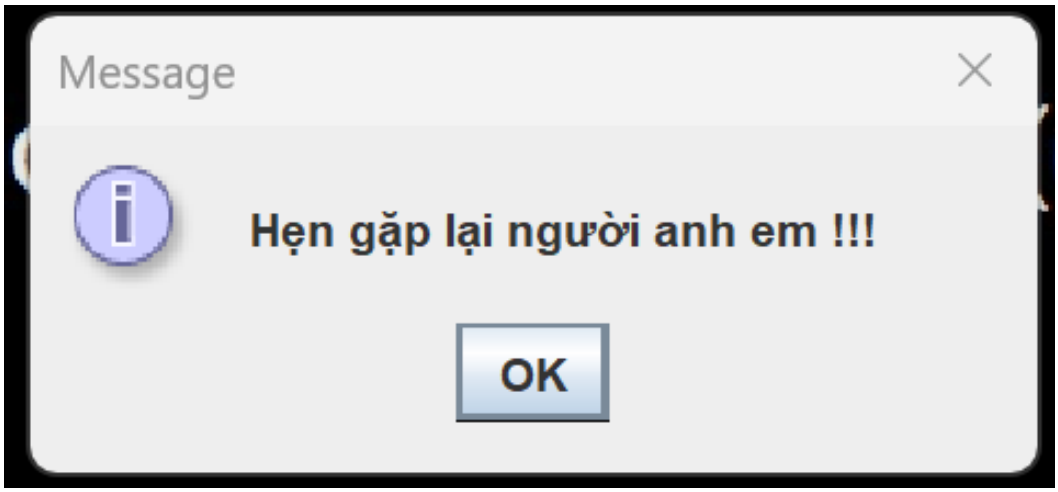
4.6. Hộp thoại thông báo điểm:

Khi kết thúc trò chơi dù người chơi thắng hay thua đều sẽ xuất hiện hộp thoại "Chào người chơi!!!" thông báo số điểm bạn đạt được, đồng thời đưa ra 2 lựa chọn "Ván nữa" để tạo game mới hoặc "Nghỉ Game" để kết thúc trò chơi.



Hình 4. 6 Hộp thoại hiển thị điểm và các lựa chọn

Khi xác nhận nghỉ game sẽ có một Messeger tạm biệt xuất hiện.



Hình 4. 7 Hộp thoại tạm biệt

CHƯƠNG 5: KẾT LUẬN

5.1. Đáng giá chung:

Sau một thời gian thực hiện, cơ bản đề tài đã được hoàn thành đúng thời hạn và có kết quả thu được gần đúng với yêu cầu đề ra. Thông qua tìm hiểu, xây dựng và phân tích trò chơi PacMan, chúng em đã có thêm kinh nghiệm trong việc lập trình bằng ngôn ngữ Java: thiết kế code, xử lý các sự kiện, vấn đề debug trong ngôn ngữ này.

Việc lập trình mới chỉ đạt yêu cầu về chức năng của game, vẫn còn nhiều thiếu sót do hạn chế về thời gian cũng như kinh nghiệm lập trình, thiết kế code.

Trong quá trình giảng dạy và học tập, nhờ có sự hướng dẫn tận tình của ThS. Nguyễn Thanh Trường cùng sự giúp đỡ của các bạn trong lớp, chúng em đã hoàn thành được đồ án với đề tài **"Phân tích và xây dựng chương trình game PacMan".** Từ đó, chúng em đã biết và tìm hiểu được các trình ứng dụng được tạo ra như thế nào, làm sao để vận hành, đồng thời giúp chúng em nắm vững được các thuật toán cốt lõi.

5.2. Chức năng đã thực hiện được:

- Xây dựng trò chơi với 4 màn chơi khác nhau.
- Xây dựng được mê cung trò chơi với các chấm Pác.
- Hình thành nhân vật PacMan với 3 mạng sống.
- Hiển thị thông báo qua màn, thua trò chơi.
- Tạo âm thanh cho trò chơi sinh động hơn.
- Xây dựng được các con ma là kẻ thù của Pacman.

5.3. Chức năng chưa thực hiện được:

- Tạo các viên năng lượng để hạn chế con ma.
- Xếp hạng người chơi.

- Tạo AI để các con ma thông minh hơn, biết đuổi theo PacMan hoặc trốn tránh PacMan khi Pacman ăn viên năng lượng.

5.4. Hạn chế:

Với một thời gian xây dựng khá ngắn cũng như lượng thành viên trong nhóm hạn chế, tuy chương tình đã giải quyết được những khó khăn khi xây dựng chương trình nhưng thực tế chương trình cũng khó tránh khỏi các sai sót.

Các sai sót thực tế có thể xảy ra và các lỗi vẫn chưa hoàn toàn được bắt có thể gây ra các thông báo không chính xác hay hiển thị không hoàn toàn theo ý muốn.

Về mặt giao diện còn đơn giản do kỹ năng lập trình còn chưa cao, kinh nghiệm viết code còn hạn chế, dẫn đến các kỹ thuật chưa được nắm bắt hết.

5.5. Hướng phát triển của đề tài:

Hướng phát triển của chúng em sẽ là nâng cao khả năng đồ họa của phần mềm trò chơi này, với nhiều hiệu ứng và màu sắc bắt mắt hơn, đồng thời sẽ xây dựng thêm một số chức năng như: tính điểm, cho phép nhiều người cùng chơi … Nhưng hướng chủ yếu là tăng hiệu ứng đồ họa để người chơi có thể thấy một cách trực quan hơn với trò chơi, tránh cảm giác nhàm chán khi chơi.

TÀI LIỆU THAM KHẢO

1. Tài liệu:

[1] Slide bài giảng Java, Th.S Nguyễn Thanh Trường, Trường Đại học Tài chính – Marketing.

2. Website:

[1] http://youtube.com

[2] https://viettuts.vn/java

[3] https://vncoder.vn/bai-hoc

[4] https://github.com