HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
----------o0o----------

TRẦN DUY HIẾU

NGHIÊN CỨU SỬ DỤNG KỸ THUẬT PROCEDURAL CONTENT GENERATION (PCG) ĐỂ PHÁT TRIỂN TỰ ĐỘNG CÁC MÀN CHƠI TRONG LẬP TRÌNH GAME

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT
(Theo định hướng ứng dụng)

HÀ NỘI - NĂM 2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

----------o0o----------

Trần Duy Hiếu

NGHIÊN CỨU SỬ DỤNG KỸ THUẬT PROCEDURAL CONTENT GENERATION (PCG) ĐỂ PHÁT TRIỂN TỰ ĐỘNG CÁC MÀN CHƠI TRONG LẬP TRÌNH GAME

CHUYÊN NGÀNH: KHOA HỌC

MÁY TÍNH MÃ SỐ: 8.48.01.04

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS. TS. NGÔ QUỐC DŨNG

HÀ NỘI – NĂM 2025

LỜI CAM ĐOAN

Tôi xin cam đoan đề án tốt nghiệp của tôi với tên đề án: " **Nghiên cứu sử dụng kỹ thuật Procedural Content Generation (PCG) để phát triển tự động các màn chơi trong lập trình game**" là công trình nghiên cứu của riêng tôi. Tôi đã sử dụng các nguồn tài liệu tham khảo chính xác và đầy đủ. Tôi không sao chép hay sử dụng bất kỳ ý tưởng hay kết quả nghiên cứu của người khác mà không ghi rõ nguồn gốc.

Tôi xin chịu trách nhiệm trước Hội đồng nếu có sai sót trong đề án tốt nghiệp của tôi.

Hà Nội, ngày 20 tháng 06 năm 2025

Học viên thực hiện đề án

Trần Duy Hiếu

LỜI CẢM ƠN

Trước hết, tôi xin bày tỏ lòng biết ơn sâu sắc đến **PGS.TS Ngô Quốc Dũng** người đã tận tình hướng dẫn, giúp đỡ tôi hoàn thành đề án tốt nghiệp này. Xin chân thành cảm ơn những lời khuyên, chỉ bảo quý báu của Thầy đã giúp tôi có thêm kiến thức và kinh nghiệm trong quá trình nghiên cứu.

Tôi xin gửi lời cảm ơn chân thành đến **Khoa Đào tạo sau đại học – Học viện Công nghệ Bưu chính Viễn thông** đã tạo mọi điều kiện thuận lợi cho tôi trong suốt quá trình học tập và hoàn thành đề án tốt nghiệp.

Cuối cùng, tôi xin gửi lời cảm ơn đến gia đình, bạn bè đã luôn động viên, ủng hộ tôi trong suốt quá trình học tập và nghiên cứu.

Tôi xin chân thành cảm ơn!

Hà Nội, ngày 20 tháng 06 năm 2025

Học viên thực hiện đề án

Trần Duy Hiếu

MỤC LỤC

DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
PCG	Procedural Content Generation	Kỹ thuật tạo nội dung tự động
	Offline PCG	PCG ngoại tuyến
	Online PCG	PCG trực tuyến
CFGs	Context-free grammars	Cấu trúc ngôn ngữ
L-systems	Lindenmayer systems	Hệ thống Lindenmayer
MCTS	Monte Carlo Tree Search	Thuật toán tìm kiếm cây dữ liệu Monte Carlo
PSO	Particle Swarm Optimization	Thuật toán Tối ưu hóa bầy hạt
PCGRL	PCG Reinforcement Learning	Thuật toán tạo nội dung tự động ứng dụng học tăng cường
GAN	Generative Adversarial Networks	Mạng đối nghịch tạo sinh
RNN	Recurrent Neural Networks	Mạng Neural
ML	Machine Learning	Học máy
NPC	Non-Player Characte	Nhân vật không phải người chơi
AI	Artificial Intelligence	Trí tuệ nhân tạo
LLN	Law of Large Numbers	Luật số lớn
CLT	Central Limit Theorem	Định lý giới hạn trung tâm
FPS	First-person shooter	Trò chơi bắn súng góc nhìn thứ nhất
BSP	Binary Space Partitioning	Thuật toán Phân vùng không gian nhị phân
DOM	Document Object Model	Mô hình Đối tượng Tài liệu
JSON	JavaScript Object Notation	Định dạng dữ liệu văn bản Javascript

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG

MỞ ĐẦU

1. Lý do chọn đề án

Ngày này, ngành công nghiệp game đã trở thành một phần không thể thiếu của giải trí toàn cầu. Người chơi ngày càng mong đợi các trò chơi có thế giới rộng lớn, cốt truyện phong phú và trải nghiệm chơi mới mẻ mỗi khi quay lại. Điều này đòi hỏi các nhà phát triển game phải tìm cách tạo ra lượng lớn nội dung mà vẫn đảm bảo chất lượng và tính sáng tạo.

Procedural Content Generation (PCG) đáp ứng được nhu cầu này bằng cách cho phép tự động tạo ra nội dung game, từ cảnh quan, địa hình đến các nhiệm vụ và thử thách trong game. Việc sử dụng PCG không chỉ giúp tiết kiệm thời gian và công sức mà còn mở ra khả năng tạo ra những trải nghiệm chơi độc đáo, mang tính ngẫu nhiên và bất ngờ, giúp trò chơi có khả năng lôi kéo người chơi quay lại.

Phát triển game là một quá trình phức tạp và tốn kém, đòi hỏi sự phối hợp chặt chẽ giữa các bộ phận từ thiết kế, lập trình đến kiểm thử. Trong đó, việc thiết kế và tạo ra các màn chơi đòi hỏi sự sáng tạo cao và thường chiếm nhiều thời gian của đội ngũ phát triển. Với PCG, các màn chơi có thể được tạo ra tự động dựa trên các thuật toán và quy tắc nhất định, giúp giảm thiểu công việc lặp đi lặp lại và cho phép đội ngũ phát triển tập trung vào những khía cạnh sáng tạo hơn của trò chơi.

Ngoài ra, việc sử dụng PCG còn giúp giảm thiểu lỗi trong quá trình thiết kế màn chơi. Các thuật toán PCG có thể được lập trình để tuân thủ các quy tắc nhất định, đảm bảo rằng các màn chơi được tạo ra đều có cấu trúc hợp lý, độ khó phù hợp và không gây ra những lỗi nghiêm trọng ảnh hưởng đến trải nghiệm của người chơi.

Vì vậy, việc lựa chọn đề án "Nghiên cứu sử dụng kỹ thuật Procedural Content Generation (PCG) để phát triển tự động các màn chơi trong lập trình game", không chỉ mang lại giá trị về mặt học thuật mà còn có ý nghĩa thực tiễn cao, đề án có thể đáp ứng nhu cầu thực tiễn của ngành công nghiệp game hiện đại, đồng thời mở ra nhiều tiềm năng phát triển và ứng dụng trong tương lai. Đề án này không chỉ giúp tối ưu hóa quy trình phát triển game mà còn mang lại những giá trị nghiên cứu quan trọng, góp phần vào sự phát triển của công nghệ và sáng tạo trong ngành game.

Từ đó học viên chọn đề án " ***Nghiên cứu sử dụng kỹ thuật Procedural Content Generation (PCG) để phát triển tự động các màn chơi trong lập trình game***". Bố cục chia làm 3 phần:

- Chương 1: Cơ sở lý thuyết
- Chương 2: Nghiên cứu lựa chọn mô hình ứng dụng kỹ thuật PCG tích hợp vào phát triển tự động tạo ra các màn chơi cho game
- Chương 3: Triển khai và xây dựng phần mềm

2. Tổng quan các đối tượng nghiên cứu

Đầu những năm 1990 game đã đến Việt Nam. Từ một ngành công nghiệp sơ khai và bị xem nhẹ, game ở Việt Nam đã phát triển thành một lĩnh vực kinh doanh lớn với nhiều cơ hội và thách thức. Với sự phát triển của công nghệ và sự quan tâm của cộng đồng, ngành công nghiệp game tại Việt Nam hứa hẹn sẽ tiếp tục có những bước tiến đáng kể trong tương lai.

Với sự phát triển nhanh chóng của ngành công nghiệp game và các ứng dụng giải trí không chỉ ở Việt Nam mà trên toàn thế giới, người chơi ngày càng mong đợi các trò chơi có thế giới rộng lớn, cốt truyện phong phú và trải nghiệm chơi mới mẻ mỗi khi quay lại. Điều này đòi hỏi các nhà phát triển game phải tìm cách tạo ra lượng lớn nội dung mà vẫn đảm bảo chất lượng và tính sáng tạo.

Procedural Content Generation (PCG) là một phương pháp sử dụng thuật toán để tự động tạo ra các nội dung số mà không cần sự can thiệp thủ công. Trong ngành công nghiệp game, PCG đóng vai trò quan trọng trong việc tạo ra các yếu tố như địa hình, màn chơi, cốt truyện, nhân vật, và thậm chí là âm thanh[2]. Khác với phương pháp truyền thống, PCG giúp tạo ra nội dung phong phú, đa dạng và có tính ngẫu nhiên cao, từ đó nâng cao trải nghiệm của người chơi.[3]

Vấn đề "***Nghiên cứu kỹ thuật Procedural Content Generation (PCG) để phát triển tự động các màn chơi trong lập trình game***" là một lĩnh vực đầy thách thức nhưng cũng rất tiềm năng. Việc giải quyết các vấn đề hiện tại và khai thác các cơ hội mới sẽ không chỉ giúp tối ưu hóa quá trình phát triển game mà còn góp phần quan trọng vào việc nâng cao chất lượng và trải nghiệm của người chơi. Điều này không chỉ có ý nghĩa quan trọng trong ngành công nghiệp game mà còn mở ra nhiều cơ hội nghiên cứu và ứng dụng PCG trong các lĩnh vực khác trong tương lai.

3. Mục đích nghiên cứu

Mục tiêu của đề án này là nghiên cứu các thuật toán PCG và phát triển hệ thống ứng dụng kỹ thuật PCG để sinh ra các màn chơi tự động, đảm bảo sự đa dạng, thú vị và

thách thức cho người chơi mà không cần sự can thiệp thủ công nhiều từ phía nhà phát triển, mong muốn của đề án là đạt được mục tiêu:

- **Nghiên cứu các thuật toán PCG phù hợp**: Tìm hiểu và phân tích các thuật toán ứng dụng kĩ thuật PCG và nghiên cứu khả năng áp dụng của các thuật toán trong việc tạo màn chơi.
- **Xây dựng phần mềm PCG tự động**: Xây dựng một phần mềm PCG có khả năng tạo ra các màn chơi với mức độ phức tạp, khó khăn và nội dung thay đổi dựa trên các thông số đầu vào cụ thể thử nghiệm trên một game để đánh giá hiệu quả.
- **Đảm bảo tính cân bằng và hợp lý**: Đảm bảo rằng các màn chơi tự động được tạo ra phù hợp về độ khó, không có lỗi nghiêm trọng và vẫn giữ được tính hấp dẫn

4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Đối tượng nghiên cứu của đề án là các nghiên cứu và công trình liên quan đến PCG, đặc biệt là trong lĩnh vực tạo màn chơi tự động.

Phạm vi nghiên cứu: Phạm vi nghiên cứu là các mô hình thuật toán PCG và các mô hình game có các màn chơi

5. Phương pháp nghiên cứu

Đề án được thực hiện dựa trên các phương pháp khảo sát, phân tích, thu thập dữ liệu từ các đối tượng nghiên cứu và ứng dụng tích hợp công cụ là phần mềm PCG vào một trò chơi cụ thể và đánh giá tác động của nó lên trải nghiệm người chơi.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1.Xu thế phát triển của ngành công nghiệp trò chơi điện tử và nhu cầu tự động hóa thiết kế màn chơi

1.1.1. Xu thế phát triển của ngành công nghiệp game hiện đại

Ngành công nghiệp trò chơi điện tử trong thế kỷ 21 đang chứng kiến sự tăng trưởng mạnh mẽ cả về quy mô thị trường, số lượng người dùng và công nghệ phát triển. Với sự phát triển nhanh chóng của công nghệ phần cứng và phần mềm, các nền tảng như máy console, PC, thiết bị di động và cả các hệ thống thực tế ảo (VR/AR) đều được khai thác tối đa nhằm đem lại trải nghiệm chơi game đa dạng và phong phú hơn. Đồng thời, công nghệ phần mềm như công cụ phát triển game (Unity, Unreal Engine), trí tuệ nhân tạo (AI), học máy (machine learning), thực tế ảo (VR) và thực tế tăng cường (AR) cũng góp phần thúc đẩy sự sáng tạo và đổi mới trong quá trình phát triển game.Theo báo cáo của Newzoo (2023), thị trường trò chơi điện tử toàn cầu dự kiến đạt giá trị hơn 200 tỷ USD trong năm 2025, với hơn 3,5 tỷ người chơi trên toàn thế giới[1]. Trong bối cảnh đó, các nhà phát triển game buộc phải thay đổi chiến lược, từ việc tập trung vào đồ họa và cốt truyện, chuyển sang chú trọng đến trải nghiệm người dùng, sự cá nhân hóa và khả năng cập nhật nội dung nhanh chóng

Một trong những xu thế nổi bật hiện nay là sự phát triển của các trò chơi có tính cá nhân hóa cao, khả năng tương tác sâu và môi trường chơi được cập nhật liên tục. Thay vì chỉ tập trung vào đồ họa hoặc cốt truyện, các nhà phát triển hiện nay chú trọng vào thiết kế trải nghiệm người dùng thông qua hệ thống gameplay phong phú, cơ chế chơi linh hoạt và đặc biệt là sự đa dạng trong hệ thống màn chơi (level design). Người chơi ngày nay không chỉ mong đợi một trò chơi có hình ảnh đẹp mà còn kỳ vọng vào khả năng khám phá không giới hạn, tính bất ngờ trong từng lần chơi và sự phản hồi thông minh từ hệ thống trò chơi.

Trong bối cảnh đó, các thể loại game như puzzle (giải đố), platformer, roguelike, sandbox hay các trò chơi phiêu lưu thế giới mở đã trở nên đặc biệt thịnh hành. Những thể loại này đòi hỏi hệ thống màn chơi có cấu trúc phức tạp, biến hóa linh hoạt và có khả năng mở rộng không giới hạn. Để đáp ứng được nhu cầu đó, việc phát triển và duy trì kho nội dung màn chơi chất lượng cao đang trở thành một thách thức không nhỏ đối với các studio game, đặc biệt là khi mô hình "Game-as-a-Service" (GaaS) đang ngày

càng phổ biến. Việc này đặt ra những yêu cầu cao hơn đối với khâu thiết kế và phát triển nội dung game.

1.1.2. Các thách thức trong việc phát triển màn chơi trong trò chơi điện tử hiện nay

Trong bối cảnh ngành công nghiệp trò chơi điện tử ngày càng phát triển mạnh mẽ, thiết kế màn chơi (level design) đang đóng vai trò then chốt trong việc nâng cao chất lượng trải nghiệm của người dùng. Sự cạnh tranh gay gắt trên thị trường cũng buộc các nhà phát hành phải liên tục tạo ra sự khác biệt về trải nghiệm người dùng. Trong đó, thiết kế màn chơi – nơi trực tiếp ảnh hưởng đến cảm nhận, thử thách và cảm xúc của người chơi – đang ngày càng trở thành yếu tố then chốt trong việc giữ chân người dùng. Những trò chơi có khả năng cung cấp trải nghiệm cá nhân hóa thông qua hệ thống màn chơi linh hoạt, phản hồi thông minh và mang tính ngẫu nhiên cao thường có lợi thế cạnh tranh lớn hơn.

Tuy nhiên, quá trình phát triển màn chơi truyền thống hiện đang đối mặt với nhiều thách thức cả về kỹ thuật lẫn chi phí, đặc biệt trong các dự án game quy mô lớn hoặc có tính liên tục cao. Những khó khăn này đặt ra nhu cầu cấp thiết về các giải pháp mới nhằm tối ưu hóa quy trình thiết kế nội dung, trong đó kỹ thuật Procedural Content Generation (PCG) nổi lên như một hướng đi đầy triển vọng [37].

Thứ nhất, chi phí và thời gian phát triển là một rào cản lớn trong thiết kế màn chơi. Đối với các trò chơi thương mại (AAA), việc xây dựng một màn chơi có thể tiêu tốn hàng tuần đến hàng tháng, yêu cầu sự phối hợp chặt chẽ giữa các bộ phận thiết kế, lập trình và đồ họa để đảm bảo đồng bộ về mặt logic và thẩm mỹ [38]. Quy trình này đòi hỏi thao tác thủ công trên công cụ thiết kế, dẫn đến chi phí phát triển cao và giới hạn về quy mô nội dung có thể triển khai trong một chu kỳ sản phẩm.

Thứ hai, vấn đề về sự đa dạng và chất lượng nội dung cũng là một yếu tố đáng lưu ý. Trong các trò chơi có số lượng lớn màn chơi như game di động hoặc game platform truyền thống, việc thiết kế nội dung thủ công thường dẫn đến tình trạng lặp lại về bố cục, thử thách và trải nghiệm [39]. Người chơi có thể dễ dàng nhận ra các mô-típ quen thuộc, làm giảm đáng kể động lực khám phá và mức độ gắn bó với trò chơi.

Thứ ba, yêu cầu cập nhật nội dung liên tục trong mô hình kinh doanh "Game as a Service" (GaaS) cũng đặt ra áp lực lớn lên đội ngũ phát triển. Các trò chơi hiện đại thường yêu cầu phải phát hành nội dung mới mỗi tháng hoặc theo các sự kiện đặc biệt. Việc đảm bảo tiến độ, chất lượng và tính nhất quán về mặt logic của các màn chơi mới

trong khi vẫn duy trì tốc độ phát triển cao là một thách thức lớn đối với mô hình thiết kế thủ công truyền thống [40].

Trước những thách thức nêu trên, kỹ thuật Kỹ thuật tạo nội dung tự động – Procedural Content Generation (PCG) nổi lên như một giải pháp khả thi nhằm:

- Tự động hóa quá trình sản xuất nội dung với tốc độ cao, tiết kiệm thời gian và chi phí [41];
- Tạo nên sự đa dạng trong thiết kế trong khi vẫn đảm bảo tuân thủ các nguyên tắc về gameplay và cấu trúc [42];
- Cho phép cá nhân hóa màn chơi dựa trên hành vi hoặc mức độ kỹ năng của người chơi thông qua các kỹ thuật PCG thích nghi [43];
- Mở rộng quy mô trò chơi một cách linh hoạt và không giới hạn, đặc biệt trong các game dạng thế giới mở ,sandbox, hoặc game được phát triển theo mô hình thủ tục [44].

Với sự phát triển vượt bậc của các kỹ thuật trí tuệ nhân tạo và học máy trong thập kỷ gần đây, PCG không còn chỉ dừng lại ở việc tạo bản đồ hoặc môi trường ngẫu nhiên đơn giản. Các hệ thống PCG hiện đại đã có khả năng học từ dữ liệu thực, phân tích hành vi người chơi và sinh ra các kịch bản phù hợp với logic thiết kế, cấp độ thử thách cũng như động lực trải nghiệm của người dùng [45]. Nhờ đó, PCG không chỉ đóng vai trò là công cụ hỗ trợ, mà còn trở thành một phần quan trọng trong việc phát triển nội dung trò chơi một cách hiệu quả, bền vững và có khả năng mở rộng cao trong tương lai.

1.2. Giới thiệu về kỹ thuật Procedural Content Generation (PCG)

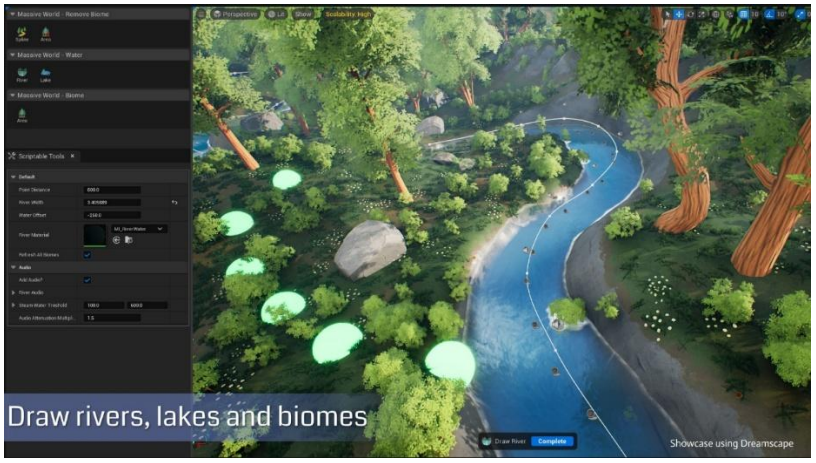
1.2.1. Tổng quan về Procedural Content Generation (PCG)

Kỹ thuật tạo nội dung tự động, còn gọi là Procedural Content Generation (PCG), được định nghĩa là quá trình tạo ra nội dung trò chơi một cách tự động hoặc bán tự động thông qua việc sử dụng các thuật toán, thay vì thiết kế thủ công từng thành phần nội dung bởi con người. Một định nghĩa phổ biến và chính xác hơn cho PCG được đưa ra là: “Việc tạo ra nội dung trò chơi thông qua các thuật toán mà không cần sự can thiệp trực tiếp của con người trong từng phần tử nội dung” [1]. Kỹ thuật này đóng góp phần trong việc giảm tải khối lượng công việc cho các nhà thiết kế trò chơi, đồng thời mang đến trải nghiệm phong phú, ngẫu nhiên, không lặp lại và có thể cá nhân hóa cho người chơi.

PCG lần đầu tiên được biết đến rộng rãi vào những năm 1980, đặc biệt thông qua trò chơi kinh điển Elite được phát hành năm 1984. Mặc dù đã xuất hiện từ sớm, tuy nhiên phải đến hai thập kỷ trở lại đây, cùng với sự phát triển mạnh mẽ của phần cứng máy tính, các thuật toán trí tuệ nhân tạo và công nghệ xử lý dữ liệu, kỹ thuật PCG mới thực sự trở thành một xu hướng quan trọng trong phát triển trò chơi điện tử.

Ngày nay, nhiều trò chơi hiện đại đã và đang tích cực áp dụng kỹ thuật PCG để tạo ra các thế giới phong phú, có chiều sâu và mang tính tái chơi cao. Tiêu biểu có thể kể đến các trò chơi như Minecraft, No Man's Sky, Spelunky và Hades, trong đó PCG được sử dụng để sinh môi trường chơi, bản đồ, nhiệm vụ và hệ thống vật phẩm đa dạng [2], [3].

Bên cạnh việc ứng dụng trong việc sinh bản đồ, thiết kế cấu trúc, tạo nhiệm vụ, sinh hình ảnh hay tạo nhân vật, kỹ thuật PCG còn mở rộng ứng dụng ra ngoài phạm vi trò chơi. Cụ thể, PCG đã được nghiên cứu và triển khai trong các lĩnh vực như mô phỏng kiến trúc đô thị, thiết kế nghệ thuật kỹ thuật số, sản xuất phim hoạt hình và đào tạo mô hình trí tuệ nhân tạo [4]. Có thể thấy rằng kỹ thuật Procedural Content Generation không chỉ là một công cụ hỗ trợ trong quá trình phát triển trò chơi hiện đại, mà còn là một giải pháp có tiềm năng ứng dụng rộng rãi trong các lĩnh vực công nghệ sáng tạo và trí tuệ nhân tạo.



Hình 1. 1 Ứng dụng PCG hỗ trợ tự tạo map trên Unity

Kỹ thuật PCG sở hữu một số đặc điểm quan trọng giúp nó trở nên phù hợp với các lĩnh vực yêu cầu nội dung đa dạng, phong phú:

- **Tính ngẫu nhiên có kiểm soát**: PCG thường sử dụng dữ liệu ban đầu ngẫu nhiên để sinh nội dung, nhưng vẫn dựa trên một tập luật hoặc cấu trúc xác định, giúp nội dung vừa đa dạng vừa hợp lý [5].

- **Khả năng mở rộng**: Một trong những điểm mạnh nổi bật của PCG là khả năng sinh ra một lượng lớn nội dung chỉ từ một mẫu nhỏ dữ liệu đầu vào. Điều này đặc biệt quan trọng trong môi trường hạn chế tài nguyên như các game indie hoặc dự án quy mô nhỏ [6].
- **Thích ứng theo người dùng**: Các phần mềm PCG hiện đại có khả năng thay đổi nội dung theo thời gian thực dựa trên hiệu suất, hành vi hoặc phản hồi của người chơi, từ đó tạo trải nghiệm được cá nhân hóa [7].
- **Tự động hóa quá trình sáng tạo**: Trong khi nhiều lĩnh vực đòi hỏi lao động thủ công cao (như thiết kế bản đồ, phần mềm nhiệm vụ trong game), PCG có thể thay thế hoặc hỗ trợ quá trình này bằng cách sinh nội dung lặp lại, tiết kiệm thời gian và nguồn lực.

1.2.2. Phân loại các kỹ thuật PCG

Tuy có các loại kỹ thuật PCG được chia theo thời điểm sử dụng hoặc mức độ can thiệp của con người, nhưng các kỹ thuật PCG được phân loại chính theo kỹ thuật sinh nội dung

- **Theo kỹ thuật sinh nội dung**:

Kỹ thuật dựa trên quy tắc (Rule-based Generation)

Kỹ thuật này sử dụng các tập luật cố định hoặc các quy tắc hình thức để điều khiển quá trình sinh nội dung[23]. Các luật có thể là:

- **L-systems** (Lindenmayer systems): Thường dùng để tạo thực vật, kiến trúc fractal.
- **Context-free grammars (CFGs)**: Dùng cho cấu trúc ngôn ngữ, level hoặc câu chuyện.
- **Grammar-based procedural modeling**: Ứng dụng mạnh trong kiến trúc đô thị, sinh ra toà nhà, phố xá.

Kỹ thuật dựa trên tìm kiếm và tối ưu (Search-based PCG)

Kỹ thuật này xem nội dung là một bài toán tối ưu, sử dụng các thuật toán tìm kiếm hoặc tiến hóa để tìm ra cấu hình nội dung tốt nhất thỏa mãn hàm mục tiêu[23].

Các thuật toán phổ biến:

- **Genetic Algorithm (GA)**: Tiến hóa một tập hợp các cá thể đại diện cho nội dung.
- **Simulated Annealing**, **A***, **Monte Carlo Tree Search (MCTS)**.

- **Evolution Strategies**, **Particle Swarm Optimization (PSO)**.

Kỹ thuật dựa trên học máy (Machine Learning-based PCG)

Các phần mềm học máy học từ tập dữ liệu mẫu để sinh nội dung tương tự. Chúng có thể là:

- **Supervised Learning**: Học từ cặp dữ liệu đầu vào – đầu ra.
- **Unsupervised Learning**: Như autoencoders để học biểu diễn trừu tượng.
- **Reinforcement Learning (PCGRL)**: Học chính sách sinh nội dung tốt thông qua phần thưởng.
- **Deep Generative Models**: Bao gồm GAN (Generative Adversarial Networks), RNN (Recurrent Neural Networks), Transformer.[25]

1.2.3. Khả năng ứng dụng của kỹ thuật PCG

Kỹ thuật Procedural Content Generation (PCG) là một hướng tiếp cận mạnh mẽ trong việc tạo ra nội dung tự động thông qua các thuật toán. Mặc dù có xuất phát điểm và được sử dụng phổ biến trong ngành công nghiệp trò chơi, nhưng hiện nay, PCG đã được mở rộng ứng dụng sang nhiều lĩnh vực khác nhau, từ mô phỏng, trí tuệ nhân tạo đến giáo dục và kiến trúc.

Kỹ thuật PCG được ứng dụng rộng rãi trong nhiều khía cạnh của phát triển trò chơi, bao gồm việc sinh bản đồ ngẫu nhiên, tạo nhiệm vụ, tạo đối thủ hoặc NPC, vật phẩm, cũng như đối thoại tương tác. Những ứng dụng này góp phần quan trọng trong việc tăng tính tái chơi mở rộng nội dung trò chơi mà không làm tăng đáng kể chi phí phát triển, đồng thời cải thiện đáng kể trải nghiệm người dùng.

Trong lĩnh vực mô phỏng và huấn luyện trí tuệ nhân tạo, PCG giúp tạo ra hàng loạt môi trường huấn luyện đa dạng, từ đó hỗ trợ các mô hình học sâu hoặc các hệ thống xe tự hành tiếp cận được với các kịch bản hiếm gặp, khó tái hiện trong thực tế [8], [9]. Ví dụ, hệ thống PCG có thể tạo ra hàng trăm tình huống giao thông giả lập, cho phép robot hoặc xe tự hành học cách phản ứng với các tình huống phức tạp một cách an toàn và hiệu quả.

Trong giáo dục và đào tạo, PCG đóng vai trò là công cụ tạo ra các nội dung cá nhân hóa, chẳng hạn như tạo bài tập hoặc câu hỏi kiểm tra phù hợp với năng lực từng học sinh. Ngoài ra, các trò chơi sử dụng PCG cũng được ứng dụng để mô phỏng các hiện tượng khoa học, phản ứng sinh học hoặc quá trình vật lý trong môi trường học tập.

Bên cạnh đó, trong lĩnh vực kiến trúc và quy hoạch đô thị, PCG được sử dụng để tạo ra các thiết kế tòa nhà, cảnh quan, hoặc các phương án bố trí mặt bằng dựa trên các thông số đầu vào nhất định. Việc tự động hóa quá trình này giúp tiết kiệm thời gian cho kiến trúc sư, đồng thời mở rộng khả năng khám phá các phương án thiết kế tối ưu hóa không gian [10].

Ưu điểm

- Tiết kiệm thời gian và chi phí phát triển: PCG giảm thiểu sự can thiệp thủ công của đội ngũ phát triển trong các tác vụ lặp đi lặp lại, đặc biệt hiệu quả đối với các dự án có quy mô nội dung lớn hoặc yêu cầu sinh hàng loạt bản đồ [6].
- Tăng tính ngẫu nhiên và khả năng tái chơi: Nhờ đặc tính sinh nội dung động, mỗi lần chơi sẽ mang đến trải nghiệm khác biệt, từ đó gia tăng giá trị sử dụng của trò chơi [3].
- Khả năng mở rộng linh hoạt: Chỉ với một tập luật hoặc thuật toán ban đầu, PCG có thể tạo ra hàng nghìn biến thể nội dung, cho phép hệ thống thích nghi với quy mô lớn.
- Khả năng cá nhân hóa nội dung: Khi kết hợp với dữ liệu hành vi người dùng, PCG có thể điều chỉnh nội dung phù hợp với từng cá nhân, nâng cao tính tương tác và khả năng học tập thích nghi [7].
- Ứng dụng đa lĩnh vực: Bên cạnh ngành công nghiệp game, PCG còn có khả năng áp dụng hiệu quả trong các lĩnh vực mô phỏng kỹ thuật, giáo dục, kiến trúc và nghệ thuật số.

Nhược điểm

- Thiếu kiểm soát chi tiết và khó phát hiện lỗi cục bộ: Do nội dung được sinh tự động, việc đảm bảo tính hợp lệ hoặc chất lượng chi tiết của từng phần tử có thể gặp khó khăn. Điều này có thể dẫn đến việc sinh ra nội dung không logic, hoặc thậm chí không thể hoàn thành được trong trò chơi [9].
- Khó duy trì chất lượng đồng đều: Nếu không có hệ thống đánh giá tự động hiệu quả, các nội dung sinh ra có thể có sự chênh lệch lớn về độ khó, tính thẩm mỹ hoặc khả năng chơi được.
- Thiếu sáng tạo nghệ thuật: Nội dung được sinh tự động thường không mang phong cách hoặc dấu ấn sáng tạo đặc thù như khi được thiết kế thủ công bởi các nhà thiết kế chuyên nghiệp. [14].

1.2.4. Vai trò của kỹ thuật PCG trong việc phát triển các màn chơi

Trong lĩnh vực phát triển trò chơi điện tử, một màn chơi (level) được hiểu là một đơn vị cấu trúc không gian hoặc logic mà người chơi phải tương tác và vượt qua để đạt được một mục tiêu cụ thể trong tiến trình chơi. Đối với các thể loại trò chơi như platformer, puzzle, roguelike, … thiết kế màn chơi không chỉ là một yếu tố quan trọng trong việc xây dựng trải nghiệm người dùng, mà còn chiếm một tỷ lệ đáng kể trong tổng chi phí phát triển trò chơi [1].

Quá trình thiết kế màn chơi đòi hỏi sự kết hợp hài hòa giữa yếu tố thách thức và khả năng giải trí, đồng thời phải đảm bảo tính nhất quán với phong cách nghệ thuật và quy luật hoạt động của trò chơi. Điều này thường yêu cầu sự tham gia trực tiếp của đội ngũ thiết kế thông qua các công cụ chỉnh sửa thủ công, dẫn đến chi phí sản xuất cao và hạn chế trong khả năng mở rộng quy mô nội dung.

Trước thực trạng này, kỹ thuật tạo nội dung tự động (Procedural Content Generation – PCG) đã và đang được ứng dụng ngày càng rộng rãi như một giải pháp tiềm năng nhằm tự động hóa quá trình thiết kế màn chơi. Thông qua việc sử dụng tập hợp các quy tắc, thuật toán, hoặc mô hình học máy, PCG có khả năng tạo ra hàng trăm đến hàng nghìn màn chơi khác nhau với chi phí thiết kế thủ công tối thiểu, đồng thời vẫn duy trì được tính nhất quán về logic và thẩm mỹ trong trò chơi tổng thể.

Việc tích hợp PCG trong thiết kế màn chơi mang lại nhiều lợi ích nổi bật như:

- Tăng khả năng chơi lại (replayability) nhờ nội dung ngẫu nhiên, không lặp lại;
- Cá nhân hóa trải nghiệm chơi dựa trên đặc điểm và năng lực của từng người dùng;
- Tạo môi trường học tập phong phú cho cả người chơi lẫn các hệ thống trí tuệ nhân tạo trong trò chơi học thích nghi (*adaptive learning games*).

Theo phân loại của Togelius và cộng sự [4], ứng dụng PCG trong thiết kế màn chơi có thể được chia thành ba cấp độ dựa trên phạm vi và chi tiết nội dung được tạo ra:

1. **PCG vi mô (Micro-level PCG)**: Tập trung vào việc sinh ra các chi tiết nhỏ trong màn chơi như vật phẩm, đối thủ, chướng ngại vật hoặc hiệu ứng hình ảnh. Đây là lớp thấp nhất trong phân cấp PCG, thường yêu cầu kiểm soát chính xác về vị trí và tính năng của từng phần tử[15].

2. **PCG trung bình (Meso-level PCG)**: Sinh ra các module cấu trúc trung gian như phòng, hành lang, hoặc khu vực chức năng. Ở cấp độ này, hệ thống cần đảm bảo sự

liên kết logic giữa các module để duy trì trải nghiệm người chơi mạch lạc và hợp lý[15].

3. **PCG vĩ mô (Macro-level PCG)**: Tạo ra toàn bộ bố cục của màn chơi hoặc thế giới trò chơi, bao gồm cấu trúc bản đồ tổng thể, tuyến hành trình và tiến trình của người chơi trong toàn bộ trò chơi. Đây là cấp độ có độ phức tạp cao nhất, yêu cầu sự phối hợp giữa nhiều yếu tố ngữ cảnh, mục tiêu, và hành vi người chơi.[15]

Mỗi cấp độ đều yêu cầu thuật toán và chiến lược kiểm soát riêng biệt để đảm bảo tính hợp lệ, tính khả thi và giá trị thẩm mỹ trong trải nghiệm người chơi.

1.3. Nghiên cứu các thuật toán PCG sử dụng trong phát triển tự động màn chơi cho game

Việc tạo màn chơi bằng PCG có thể sử dụng một hoặc nhiều kỹ thuật đã được giới thiệu trong phần 1.2. Các thuật toán PCG cho màn chơi có thể được phân thành cácnhóm chínhnhư sau:

- Thuật toán dựa trên quy tắc (Rule-based)
- Thuật toán tìm kiếm và tối ưu hóa (Search-based)
- Thuật toán dựa trên ràng buộc (Constraint-based)
- Thuật toán dựa trên học máy (Machine Learning-based)

1.3.1. Thuật toán dựa trên quy tắc (Rule-Based Generation)

a) Mô tả

Thuật toán này sử dụng một tập hợp các quy tắc xác định trước (predefined rules) để điều khiển quá trình tạo ra nội dung trò chơi, bao gồm bản đồ, vật thể, vị trí mục tiêu hoặc cấu trúc nhiệm vụ. Nội dung được sinh ra không phải ngẫu nhiên hoàn toàn, mà phải tuân theo các điều kiện logic và ràng buộc đã được lập trình sẵn, đảm bảo tính hợp lệ và khả năng chơi được của kết quả đầu ra.

Nguyên lý hoạt động cơ bản của Rule-Based Generation là định nghĩa một tập hợp các quy tắc mô tả cách các phần tử trong game nên được phân bố hoặc tổ chức. Các quy tắc này có thể được biểu diễn dưới dạng mệnh đề logic, biểu thức điều kiện, hoặc cây quyết định (decision trees), và được áp dụng tuần tự hoặc kết hợp để tạo ra bản đồ hợp lệ [24].

Trong phạm vi thiết kế màn chơi tự đông , một số loại quy tắc của thuật toán bao gồm:

- **Quy tắc cấu trúc (Structural Rules):** Định nghĩa mối quan hệ không gian giữa các thành phần, ví dụ như "không đặt tường chồng lên vật phẩm", "lối ra phải nối được với điểm bắt đầu".
- **Quy tắc vị trí (Placement Rules):** Chỉ định nơi hợp lệ để đặt các đối tượng như người chơi, kẻ thù, hộp, chìa khóa, mục tiêu.
- **Quy tắc tương tác (Interaction Rules):** Mô tả các mối liên hệ logic giữa các đối tượng, ví dụ "nếu có kẻ thù thì cần đặt vũ khí ở gần", hoặc "khoảng cách giữa hai vật phẩm cùng loại phải tối thiểu là 2 ô".

Quá trình thực thi thuật toán thường gồm ba giai đoạn chính:

1. **Khởi tạo không gian:** Tạo bản đồ nền có kích thước và cấu trúc cơ bản (như lưới hoặc biểu đồ).

2. **Áp dụng quy tắc sinh nội dung:** Quy tắc được gọi tuần tự (theo pipeline) hoặc theo trình tự ngẫu nhiên có kiểm soát.

3. **Xác nhận và tinh chỉnh:** Áp dụng các bước đánh giá hậu kiểm (post-validation) để đảm bảo tính hợp lệ và khả năng chơi được của nội dung đã sinh.

b) Thuật toán ví dụ

Dugeon Generator: Dungeon Generator là thuật toán thường sử dụng kỹ thuật space partitioning như Binary Space Partitioning (BSP)[16]. Thuật toán chia bản đồ thành các vùng con đệ quy, rồi tạo phòng ngẫu nhiên trong các vùng đó. Sau đó, các phòng được kết nối bằng hành lang đảm bảo tính liên thông toàn bản đồ.

Các bước chính:

- Chia toàn bộ bản đồ thành 2 phần bằng đường dọc hoặc ngang (đệ quy đến kích thước tối thiểu).
- Sinh ngẫu nhiên phòng bên trong mỗi vùng con.
- Tạo hành lang nối các phòng liền kề.
- Gắn thêm vật phẩm, cửa, theo quy tắc.

Cellular Automata: Thuật toán Cellular Automata áp dụng quy tắc cập nhật lân cận để "mô phỏng" địa hình tự nhiên, thường dùng để sinh các bản đồ hang động hoặc bản đồ có hình dạng phức tạp[16].

Các bước chính

1. Khởi tạo bản đồ 2D với các ô có xác suất là "đá" hoặc "trống".

2. Lặp nhiều vòng cập nhật trạng thái mỗi ô dựa vào số ô "đá" xung quanh nó.
3. Cắt bỏ các vùng nhỏ, cô lập.
4. Đảm bảo bản đồ còn liên thông (bằng BFS hoặc DFS kiểm tra connectivity).

c) Khả năng ứng dụng thuật toán trong các thể loại game

Thuật toán Rule-Based tỏ ra đặc biệt hiệu quả trong các tình huống có cấu trúc nội dung rõ ràng và các ràng buộc không gian cụ thể. Một số lĩnh vực ứng dụng tiêu biểu bao gồm:

- **Game giải đố (Puzzle Games):** Dễ kiểm soát logic, thiết lập độ khó từng bước bằng cách thay đổi quy tắc.
- **Game platformer 2D:** Dễ xây dựng bản đồ màn chơi theo lớp (layered layout), có thể áp dụng quy tắc vị trí và cấu trúc.
- **Trò chơi học tập (Educational Games):** Dễ cấu hình để sinh bài tập phù hợp với từng trình độ người chơi.

1.3.2. Thuật toán tìm kiếm và tối ưu hóa (Search-Based PCG)

a) Mô tả

Thuật toán này coi bài toán sinh nội dung – cụ thể là bản đồ hoặc cấu trúc của một màn chơi – là một quá trình tối ưu hóa. Nội dung cần tạo ra được xem như là một lời giải trong không gian tìm kiếm, và hệ thống sẽ sử dụng các thuật toán tìm kiếm để tìm ra lời giải tốt nhất theo một số tiêu chí xác định trước, thông qua một hàm đánh giá [25]

Quá trình hoạt động của SBPCG có thể được mô tả qua các bước chính sau:

1. **Định nghĩa không gian nội dung (Search Space):** Là tập hợp tất cả các màn chơi khả thi có thể được tạo ra, thường được mã hóa dưới dạng ma trận chuỗi ký hiệu, cây hoặc đồ thị, biểu diễn vị trí tường, vật cản, mục tiêu, nhân vật...

2. **Tạo cá thể khởi tạo (Initialization):** Một hoặc nhiều cá thể màn chơi được khởi tạo ngẫu nhiên để bắt đầu quá trình tìm kiếm.

3. **Đánh giá chất lượng (Fitness Evaluation):** Mỗi cá thể sẽ được đánh giá theo một hoặc nhiều tiêu chí như:

- Tính khả thi : màn chơi có thể hoàn thành được hay không.
- Độ khó: mức độ thử thách phù hợp với người chơi mục tiêu.
- Độ đa dạng và tính mới lạ
- Cân bằng game

4. **Tìm kiếm lời giải tối ưu:** Sử dụng các thuật toán tối ưu như **Genetic Algorithm**, **Simulated Annealing**, hoặc **Monte Carlo Tree Search (MCTS)** để tạo ra các biến thể mới từ những cá thể hiện có, nhằm cải thiện điểm đánh giá.

5. **Điều kiện dừng:** Quá trình sẽ lặp lại cho đến khi đạt

b) Thuật toán ví dụ

Genetic Algorithm (GA): mô phỏng tiến hóa để tìm ra màn chơi tốt nhất thông qua quần thể giải pháp.

Các bước thực hiện

- **Mã hóa cá thể**: Mỗi màn chơi được mã hóa thành chuỗi (ví dụ: ma trận tile).
- **Khởi tạo quần thể:** Tạo tập hợp màn chơi ngẫu nhiên ban đầu.
- **Đánh giá :** Sử dụng các hàm như: Độ khó phù hợp, Tỷ lệ tile tường/sàn hợp lý, Có thể hoàn thành không (playability test)
- **Lai ghép + đột biến:** Sinh cá thể mới bằng recombination + mutation.
- **Lặp:** Lặp đến khi đạt được cá thể tối ưu hoặc số thế hệ tối đa.

Monte Carlo Tree Search (MCTS): là thuật toán tìm kiếm trong không gian hành động bằng cách lấy mẫu ngẫu nhiên và xây dựng cây quyết định.

Các bước chính

- **Selection:** Từ gốc, chọn node theo UCB1 (Upper Confidence Bound).
- **Expansion:** Nếu node không phải lá, mở rộng cây bằng cách thêm hành động mới.
- **Simulation:** Thực hiện ngẫu nhiên đến trạng thái kết thúc (màn chơi đầy đủ).
- **Backpropagation:** Cập nhật giá trị node dựa trên kết quả mô phỏng.

c) Khả năng ứng dụng thuật toán trong các thể loại game

Thuật toán Search-Based PCG đặc biệt phù hợp trong các bối cảnh thiết kế trò chơi đòi hỏi tính đa dạng nội dung, sự tùy biến cao và khả năng tối ưu hóa dựa trên các mục tiêu định lượng cụ thể.

- **Game giải đố (Puzzle Games):** Có thể được dùng để sinh ra các thử thách logic có mức độ khó khác nhau, đồng thời đảm bảo tính duy nhất của lời giải.
- **Game chiến thuật thời gian thực (RTS) hoặc chiến thuật theo lượt (TBS):** Thuật toán có thể hỗ trợ sinh bản đồ thi đấu, bố trí tài nguyên, hoặc thiết lập điều kiện khởi đầu sao cho đảm bảo cân bằng giữa các bên
- **Game platformer:** Với đặc trưng về chuyển động theo chiều ngang, nhảy và

né tránh vật cản, các game platformer có thể tận dụng SBPCG để sinh ra địa hình, vật phẩm, và chướng ngại sao cho phù hợp với nhịp độ và kỹ năng của người chơi

1.3.3. Thuật toán dựa trên ràng buộc (Constraint-Based PCG)

a) Mô tả

Thuật toán này sử dụng các tập hợp ràng buộc– tức là các quy tắc, điều kiện hoặc giới hạn được xác định trước – để định hướng quá trình sinh nội dung một cách hợp lệ, có kiểm soát. Đây là cách tiếp cận đặc biệt phù hợp trong các trò chơi dạng puzzle như Sokoban, sudoku, hoặc các game xây dựng màn chơi có cấu trúc phức tạp, đòi hỏi tuân thủ các quy luật logic nhất định[24].

BPCG thường được cài đặt thông qua các kỹ thuật giải bài toán ràng buộc như:

- **Constraint Satisfaction Problem (CSP):** sử dụng trong việc xác định các giá trị biến hợp lệ với các ràng buộc rời rạc.
- **Backtracking search:** tìm kiếm lời giải trong không gian trạng thái, lùi lại khi gặp mâu thuẫn.
- **SAT/SMT solvers:** áp dụng cho các bài toán ràng buộc logic phức tạp.

b) Thuật toán ví dụ

Wave Function Collapse: sinh bản đồ bằng cách chọn trạng thái khả thi cho mỗi ô sao cho thỏa mãn các ràng buộc từ lân cận.

Các bước thực hiện

- Phân tích mẫu đầu vào: Trích xuất các tile mẫu và luật liên kết
- Khởi tạo mỗi ô bản đồ ở trạng thái đa khả năng
- Chọn ô có entropy thấp nhất (ít lựa chọn nhất).
- Collapse (gán giá trị) → Lan truyền ràng buộc → Lặp lại đến khi hoàn tất hoặc thất bại**.**

c) Khả năng ứng dụng thuật toán trong các thể loại game

Thuật toán dựa trên ràng buộc (Constraint-Based PCG) thể hiện hiệu quả cao trong việc tạo nội dung cho các trò chơi có cấu trúc logic chặt chẽ hoặc yêu cầu nghiêm ngặt về bố cục, sự hợp lệ và tính khả thi trong gameplay. Thuật toán phù hợp với nhiều thể loại game hiện đại có yêu cầu cao về độ chính xác và kiểm soát nội dung

- **Game giải đố (Puzzle Games):**. Các bài toán logic như Sudoku, Sokoban, hoặc các loại câu đố không gian đều yêu cầu hệ thống sinh nội dung đảm bảo tính duy nhất của lời giải và độ khó có thể kiểm soát được

- **Game xây dựng và mô phỏng (Simulation / City-building Games):** Constraint-Based PCG giúp tự động hóa việc sinh bản đồ, cấu trúc khu vực, hoặc thiết kế không gian sao cho phù hợp với mục tiêu phát triển của người chơi và các quy luật phát triển đô thị ảo.
- **Game chiến thuật và nhập vai (Strategy / RPG Games):** Các bản đồ chiến thuật, hệ thống nhiệm vụ hoặc mô hình cây kỹ năng có thể được sinh theo ràng buộc về tiến trình, cấp độ nhân vật, hoặc tương tác giữa các nhân tố gameplay

1.3.4. Thuật toán dựa trên học máy (Machine Learning-Based PCG)

a) Mô tả

Thuật toán sẽ sinh nội dung trò chơi bằng cách sử dụng các mô hình học máy được huấn luyện từ tập dữ liệu mẫu (dataset), thường bao gồm các màn chơi đã được thiết kế thủ công. Mô hình học từ các đặc điểm đặc trưng trong dữ liệu (ví dụ: cấu trúc không gian, mức độ thử thách, cách bố trí vật thể…) để từ đó tổng quát hóa và tạo ra các màn chơi mới[26].

Một số kỹ thuật thường được áp dụng trong ML-PCG bao gồm:

- **Supervised Learning (Học có giám sát)**: Áp dụng khi có dữ liệu gán nhãn rõ ràng, ví dụ: phân loại độ khó của màn chơi, dự đoán vị trí vật thể.
- **Unsupervised Learning (Học không giám sát)**: Dùng để khám phá cấu trúc tiềm ẩn trong dữ liệu, ví dụ: nhóm các loại màn chơi tương đồng.
- **Generative Adversarial Networks (GANs)**: Mô hình sinh đối kháng được ứng dụng để sinh ra bản đồ hoặc bố cục màn chơi tương tự như bản thiết kế thật.
- **Variational Autoencoders (VAEs)**: Mô hình mã hóa – giải mã giúp học biểu diễn ẩn của dữ liệu màn chơi và tạo ra biến thể mới có tính nhất quán.
- **Reinforcement Learning (Học tăng cường)**: Áp dụng trong môi trường động, nơi hệ thống sinh màn chơi học cách tối ưu hóa cấu trúc dựa trên phản hồi từ người chơi hoặc mô phỏng.

b) Thuật toán ví dụ

Generative Adversarial Networks : GAN gồm hai mạng học sâu – Generator (G) và Discriminator (D) – đấu với nhau. G cố gắng sinh màn chơi giống thật, D cố gắng phân biệt thật/giả[26].

Các bước thực hiện

- G sinh bản đồ từ vector nhiễu ngẫu nhiên.

- D đánh giá bản đồ là thật (từ dữ liệu huấn luyện) hay giả.
- G và D học liên tục: G cải thiện khả năng lừa D, D cải thiện khả năng phân biệt.

Long Short-Term Memory (LSTM): LSTM là mạng hồi tiếp cải tiến có khả năng ghi nhớ các quan hệ dài hạn trong chuỗi dữ liệu – thích hợp để sinh các chuỗi tile như màn chơi dạng tuyến tính (platform)[26].

Các bước chính

- **Selection:** Từ gốc, chọn node theo UCB1 (Upper Confidence Bound).
- **Expansion:** Nếu node không phải lá, mở rộng cây bằng cách thêm hành động mới.
- **Simulation:** Thực hiện ngẫu nhiên đến trạng thái kết thúc (màn chơi đầy đủ).
- **Backpropagation:** Cập nhật giá trị node dựa trên kết quả mô phỏng.

c) Khả năng ứng dụng thuật toán trong các thể loại game

Thuật toán dựa trên học máy (Machine Learning-Based PCG) mang lại tiềm năng to lớn trong việc cá nhân hóa nội dung, tối ưu hóa trải nghiệm người chơi và nâng cao khả năng thích ứng của trò chơi với hành vi người dùng

- **Game phiêu lưu thế giới mở:** ML-based PCG có thể học từ dữ liệu bản đồ có sẵn hoặc hành vi người chơi để sinh ra các vùng đất, nhiệm vụ phụ (side quests), hoặc hành vi NPC có tính thích nghi cao, qua đó làm tăng cảm giác chân thực và khả năng nhập vai của người chơi**.**
- **Game chiến lược và xây dựng:** Machine Learning giúp sinh các chiến lược đối thủ có thể học được từ cách chơi của người dùng, đồng thời cá nhân hóa bản đồ theo phong cách chơi quen thuộc.
- **Game hành động nhập vai (Action RPG)** : Thuật toán học máy có thể được huấn luyện từ dữ liệu người chơi để tạo ra các thử thách phù hợp với kỹ năng hiện tại, hoặc sinh các loot và vật phẩm có giá trị tương ứng với tiến trình của trò chơi.

1.4.Nghiên cứu về lý thuyết xác xuất, lý thuyết trò chơi

1.4.1. Tổng quan về lý thuyết xác xuất

Lý thuyết xác suất (Probability Theory) là áp dụng cho thuật toán không thể dự đoán một cách chắc chắn, mà thay vào đó được mô tả thông qua các khả năng xảy ra với những mức độ xác suất khác nhau. Đây là nền tảng lý thuyết cốt lõi của nhiều lĩnh vực hiện đại như thống kê suy diễn, học máy (machine learning), trí tuệ nhân tạo (AI), xử lý

tín hiệu số, cũng như trong thiết kế các hệ thống tạo nội dung tự động (procedural content generation).[17]

a) Không gian xác suất

Một không gian xác suất được định nghĩa dưới dạng bộ ba Ω,F,P), bao gồm:

- Ω (*sample space*): Tập hợp tất cả các kết quả có thể của một thí nghiệm ngẫu nhiên.
- F (*sigma-algebra*): Tập hợp các biến cố (hay sự kiện) có thể đo lường được trên Ω\OmegaΩ, thỏa mãn các tính chất của đại số sigma.
- P (*probability measure*): Một hàm ánh xạ từ F vào khoảng [0,1][0,1][0,1], thoả mãn tính chất sigma-additive P(Ω)=1.[27]

b) Các định lý nền tảng

Hai định lý quan trọng trong lý thuyết xác suất hỗ trợ cho việc suy luận thống kê và học máy là:

- Luật số lớn (Law of Large Numbers – LLN): khẳng định rằng trung bình của các quan sát độc lập sẽ hội tụ về kỳ vọng của biến ngẫu nhiên khi số lượng mẫu tiến đến vô hạn).[17];
- Định lý giới hạn trung tâm (Central Limit Theorem – CLT): phát biểu rằng tổng (hoặc trung bình) của nhiều biến ngẫu nhiên độc lập và có phân phối bất kỳ nhưng cùng kỳ vọng và phương sai hữu hạn sẽ xấp xỉ phân phối chuẩn khi số lượng biến đủ lớn.[29]

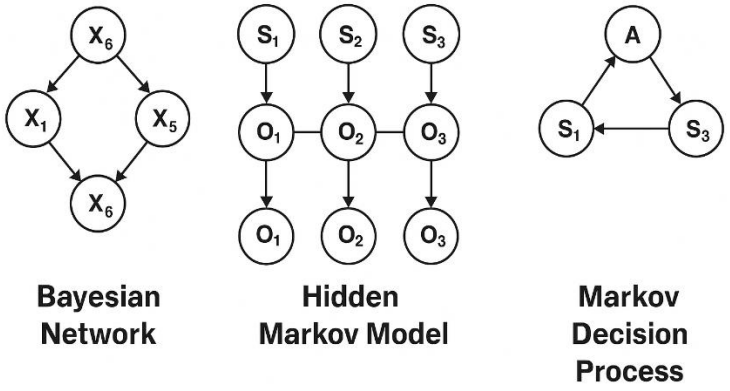
Những định lý này là cơ sở để suy luận thống kê và xây dựng mô hình học từ dữ liệu.

c) Các mô hình xác suất trong AI

Trong lĩnh vực trí tuệ nhân tạo, lý thuyết xác suất được áp dụng rộng rãi thông qua các mô hình đồ họa và chuỗi thời gian, nhằm mô tả mối quan hệ giữa các biến hoặc trạng thái trong hệ thống. Một số mô hình quan trọng bao gồm:

- **Mạng Bayes (Bayesian Networks)**: là mô hình đồ thị có hướng thể hiện mối quan hệ nhân quả giữa các biến ngẫu nhiên. Các cạnh trong đồ thị biểu diễn mối liên hệ xác suất có điều kiện, cho phép hệ thống suy luận dựa trên thông tin không đầy đủ).[17].
- **Mạng Markov (Markov Random Fields)**: là mô hình không có hướng, đặc biệt hữu ích trong các hệ thống có tính chất cục bộ cao như xử lý ảnh và thị giác máy tính. [17]

- **Mô hình Markov ẩn (Hidden Markov Model – HMM)**: là công cụ mạnh để mô hình hóa các quá trình chuỗi có trạng thái ẩn, như nhận dạng tiếng nói, phân tích dữ liệu thời gian hoặc xử lý ngôn ngữ tự nhiên.[17]
- **Markov Decision Process (MDP)**: là mô hình ra quyết định trong môi trường có tính ngẫu nhiên, có sử dụng khái niệm phần thưởng và xác suất chuyển trạng thái. MDP là nền tảng quan trọng trong học tăng cường (*reinforcement learning*) và các hệ thống học thích nghi. [17]



Hình 1. 2 Các mô hình xác suất trong AI

Những mô hình này đóng vai trò quan trọng trong NLP, xử lý tín hiệu, robot học và ra quyết định trong điều kiện bất định.

Lý thuyết xác suất không chỉ cung cấp ngôn ngữ toán học để mô tả sự bất định, mà còn đóng vai trò then chốt trong việc phát triển các mô hình thông minh và tự động. Các nguyên lý của xác suất giúp hệ thống phần mềm có thể học hỏi từ dữ liệu, dự đoán xu hướng, ra quyết định hiệu quả và sinh ra nội dung một cách linh hoạt.

1.4.2. Tổng quan về lý thuyết trò chơi

Lý thuyết trò chơi (Game Theory) là một lĩnh vực liên ngành trong toán học ứng dụng, kinh tế học và khoa học máy tính, chuyên nghiên cứu các mô hình tương tác chiến lược giữa các tác nhân có lý trí (rational agents). Lý thuyết này cung cấp một khung lý thuyết toàn diện nhằm phân tích các tình huống trong đó hành động của một đối tượng không chỉ phụ thuộc vào quyết định của chính nó mà còn phụ thuộc vào lựa chọn của các đối tượng khác trong cùng môi trường. Mục tiêu chính của lý thuyết trò chơi là xác định các chiến lược tối ưu cho mỗi tác nhân trong một trò chơi, dựa trên giả định rằng tất cả các bên đều hành xử một cách hợp lý và có hiểu biết đầy đủ về quy tắc của trò chơi [18]

Lý thuyết trò chơi được khởi nguồn từ công trình nền tảng "*Theory of Games and Economic Behavior*" (1944) của John von Neumann và Oskar Morgenstern, đánh dấu bước ngoặt trong việc ứng dụng các phương pháp toán học vào kinh tế học và ra quyết định [31]. Từ đó, lĩnh vực này đã được mở rộng và ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như quản trị kinh doanh, chiến lược cạnh tranh, đàm phán, sinh học tiến hóa, chính trị học, và gần đây là trong lĩnh vực trí tuệ nhân tạo và thiết kế hệ thống đa tác nhân (multi-agent systems). Một khái niệm quan trọng trong lý thuyết trò chơi là cân bằng Nash (Nash Equilibrium) – tình huống trong đó không người chơi nào có động lực thay đổi chiến lược của mình đơn phương vì họ đã đạt lợi ích tối đa, giả sử các người chơi còn lại không thay đổi lựa chọn của mình [19].

Trong lĩnh vực khoa học máy tính, đặc biệt là trí tuệ nhân tạo, lý thuyết trò chơi đóng vai trò quan trọng trong thiết kế các hệ thống đa tác nhân, nơi các thực thể phần mềm cần phối hợp hoặc cạnh tranh để đạt mục tiêu riêng. Các ứng dụng cụ thể bao gồm: đấu giá điện tử, lập lịch phân phối tài nguyên, điều phối giao thông thông minh, và các trò chơi chiến lược ảo (game AI). Gần đây, lý thuyết trò chơi cũng được tích hợp vào các mô hình học tăng cường nhiều tác nhân để xây dựng các hệ thống thích nghi trong môi trường động [20].

1.5. Kết luận chương

Chương 1 đã trình bày cơ sở lý luận và bối cảnh nghiên cứu cho việc ứng dụng kỹ thuật Procedural Content Generation (PCG) trong phát triển màn chơi trò chơi điện tử. Trước hết, chương đã phân tích xu thế phát triển nhanh chóng của ngành công nghiệp game hiện đại, trong đó yêu cầu về việc cung cấp nội dung đa dạng, phong phú, cá nhân hóa và có khả năng cập nhật liên tục được xác định là yếu tố then chốt. Bên cạnh đó, các hạn chế của phương pháp thiết kế màn chơi thủ công truyền thống, như chi phí phát triển cao, thời gian xây dựng kéo dài và khó khăn trong việc mở rộng quy mô, đã được chỉ ra rõ ràng.

Để giải quyết những thách thức này, chương đã giới thiệu kỹ thuật PCG như một giải pháp tiềm năng, cho phép tự động hóa quá trình tạo nội dung, nâng cao tính ngẫu nhiên, khả năng tái chơi, cũng như mở rộng phạm vi và quy mô thiết kế một cách hiệu quả. Tiếp theo, đề án đã phân loại và phân tích các nhóm thuật toán PCG phổ biến, bao gồm: thuật toán dựa trên quy tắc, thuật toán tối ưu hóa và tìm kiếm, thuật toán dựa trên ràng buộc và thuật toán học máy.

Ngoài ra, chương cũng đã đề cập đến các cơ sở toán học liên quan, cụ thể là lý thuyết xác suất và lý thuyết trò chơi, nhằm cung cấp nền tảng khoa học cho việc xây dựng và tối ưu hóa các thuật toán PCG, đảm bảo nội dung sinh ra vừa đáp ứng tính hợp lệ, vừa phù hợp với hành vi và trải nghiệm của người chơi.

Kết quả phân tích trong chương 1 là nền tảng quan trọng cho các nội dung nghiên cứu và triển khai kỹ thuật PCG trong các chương tiếp theo của đề án.

CHƯƠNG 2:NGHIÊN CỨU LỰA CHỌN MÔ HÌNH ỨNG DỤNG KỸ THUẬT PCG TÍCH HỢP VÀO PHÁT TRIỂN TỰ ĐỘNG TẠO RA CÁC MÀN CHƠI CHO GAME

Trong chương này, đề án sẽ nghiên cứu và phân tích lựa chọn từ các thuật toán trong PCG tìm hiểu ở chương 1 để nghiên cứu xây dựng một mô hình phần mềm ứng dụng kỹ thuật PCG có thể tích hợp hỗ trợ tạo ra các màn chơi tự động, đáp ứng nhu cầu của người chơi và yêu cầu của nhà phát triển.

2.1. Nghiên cứu Mô hình phần mềm sử dụng kỹ thuật PCG để tạo sinh nội dung tự đông màn chơi cho các game

Một hệ thống phần mềm áp dụng kỹ thuật Procedural Content Generation (PCG) thường được cấu trúc gồm bốn thành phần chính: (1) đầu vào (input data), (2) mô-đun sinh nội dung (Generator), (3) mô-đun đánh giá chất lượng (evaluation module), và (4) đầu ra (generated content). Các thành phần này phối hợp chặt chẽ với nhau nhằm đảm bảo quy trình sinh nội dung tự động diễn ra hiệu quả, có kiểm soát và đáp ứng được mục tiêu thiết kế của hệ thống[15].

Input Data vào đóng vai trò xác định các điều kiện khởi tạo và giới hạn cho quá trình tạo nội dung. Tùy theo phương pháp triển khai, đầu vào có thể bao gồm các giá trị seed, tập luật hoặc quy tắc thiết kế, mô hình học máy đã huấn luyện sẵn, hoặc thậm chí là dữ liệu hành vi người chơi thu thập được từ các phiên chơi trước. Các thông số đầu vào này ảnh hưởng trực tiếp đến đặc tính ngẫu nhiên, tính đa dạng và mức độ cá nhân hóa của nội dung được tạo ra.

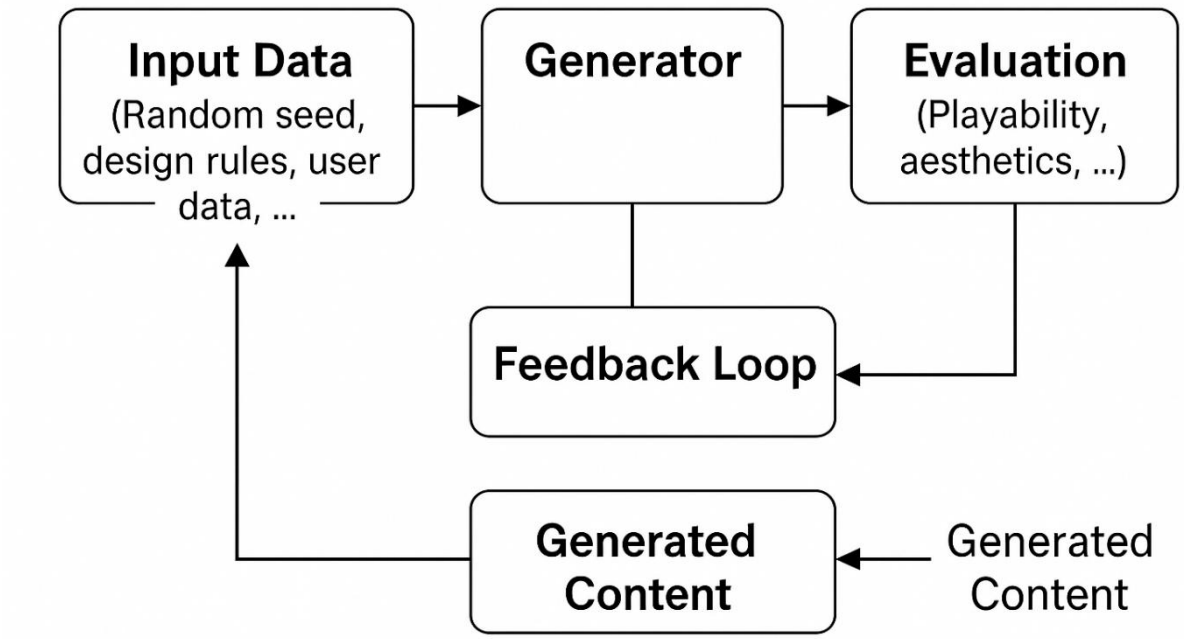
Generator (Mô-đun sinh nội dung) là thành phần cốt lõi, chịu trách nhiệm tạo ra các bản đồ, nhiệm vụ, vật phẩm hoặc đối tượng trong trò chơi dựa trên đầu vào đã cung cấp. Tùy theo yêu cầu hệ thống, mô-đun này có thể triển khai các thuật toán sinh nội dung như L-System, Perlin Noise, Markov Chains, hoặc các mô hình học máy hiện đại như Generative Adversarial Networks (GANs). Việc lựa chọn thuật toán phù hợp phụ thuộc vào loại nội dung cần tạo, mức độ phức tạp của thế giới trò chơi và yêu cầu về tính ngẫu nhiên hoặc tính logic.

Evaluation(Mô-đun đánh giá chất lượng) có vai trò đảm bảo rằng nội dung sinh ra đáp ứng các tiêu chí thiết kế nhất định. Các tiêu chí này thường bao gồm khả

năng chơi được (playability), độ khó hợp lý, tính nhất quán logic, và yếu tố thẩm mỹ. Đối với các hệ thống chuyên sâu hơn, mô-đun này còn thực hiện kiểm tra chéo với dữ liệu thực tế để phát hiện các trường hợp lỗi hoặc các kịch bản bất khả thi. Việc tích hợp mô-đun đánh giá chất lượng giúp đảm bảo đầu ra không chỉ hợp lệ về mặt kỹ thuật mà còn phù hợp với trải nghiệm người dùng.

Generated Content(Kết quả đầu ra) là nội dung trò chơi đã được tạo và kiểm duyệt, có thể được sử dụng trực tiếp hoặc tiếp tục tinh chỉnh thông qua vòng lặp phản hồi. Trong các hệ thống hiện đại

Vòng lặp phản hồi (feedback loop) thường được tích hợp để cải tiến quá trình sinh nội dung dựa trên phản hồi của người chơi hoặc phần mềm mô phỏng. Vòng lặp này cho phép hệ thống học hỏi và tự điều chỉnh nhằm tạo ra các nội dung ngày càng phù hợp hơn với mục tiêu thiết kế và hành vi người chơi.



Hình 2. 1Mô hình tổng quan hoạt động kỹ thuật PCG tạo sinh

Quy trình hoạt động của mô hình được tổ chức theo vòng lặp khép kín, gồm các bước sau:

- Bước 1: Người dùng (thường là nhà phát triển hoặc nhà thiết kế game) cấu hình các tham số đầu vào thông qua giao diện điều khiển.
- Bước 2: Bộ máy sinh nội dung áp dụng các quy tắc đã định nghĩa để tạo bản đồ màn chơi.
- Bước 3: Nội dung được chuyển sang module đánh giá để kiểm tra chất lượng

và tính khả thi.

- Bước 4: Nếu nội dung đạt yêu cầu, hệ thống xuất kết quả; nếu không, hệ thống sẽ tự động tái sinh (hoặc cho phép người dùng điều chỉnh quy tắc).

Trong mô hình tổng quát của hệ thống tạo sinh nội dung tự động bằng kỹ thuật Procedural Content Generation (PCG), các thuật toán PCG — bao gồm Rule-Based, Search-Based, Constraint-Based hoặc Machine Learning-Based — đóng vai trò trung tâm trong mô-đun sinh nội dung (Generation Engine). Đây là thành phần chịu trách nhiệm chính trong việc hiện thực hóa quá trình tạo nội dung mới từ dữ liệu đầu vào đã xác định. Việc triển khai các thuật toán PCG trong mô-đun sinh nội dung còn cho phép sử dụng cho nhiều thể loại game và triển khai cho nhiều nội dung và chức năng khác nhau

2.2.Phân tích khả năng ứng dụng các thuật toán PCG tạo màn chơi tự động trong các thể loại game theo cấu trúc màn chơi (level-based game)

Trò chơi theo cấu trúc màn chơi (level-based games) là nhóm game trong đó người chơi tiến triển qua các màn chơi riêng biệt, được thiết kế với độ khó và mục tiêu khác nhau, thường đòi hỏi kỹ thuật tạo nội dung đa dạng và khả năng điều chỉnh độ khó hợp lý Những thể loại này đòi hỏi hệ thống màn chơi có cấu trúc phức tạp, biến hóa linh hoạt và có khả năng mở rộng không giới hạn, vì vậy các thể loại game này có thể áp dụng các thuật toán PCG nhằm tạo ra hàng trăm đến hàng nghìn màn chơi khác nhau với chi phí thiết kế thủ công tối thiểu, đồng thời vẫn duy trì được tính nhất quán về logic và thẩm mỹ trong trò chơi tổng thể.

Trong nghiên cứu này, em sẽ phân tích áp dụng thuật toán PCG cho các thể loại game cấu trúc màn chơi, các thể loại game phổ biến nhất của loại game cấu trúc màn chơi là: ***Roguelike games, Platformer games, Puzzle games***

2.2.1 Ứng dụng các thuật toán PCG tạo màn chơi tự động cho thể loại game Roguelike

Roguelike là một thể loại trò chơi điện tử có đặc điểm nổi bật là thiết kế màn chơi theo kiểu ngẫu nhiên, hệ thống bản đồ được sinh động mỗi lần chơi, độ khó cao, và cơ chế "permadeath" (chết vĩnh viễn không lưu game). Ưu điểm của trò chơi thể loại này nằm ở khả năng tạo ra trải nghiệm mới mẻ, không lặp lại giữa các lần chơi, từ đó nâng cao giá trị chơi lại (replayability) cho người dùng. Việc áp dụng kỹ thuật PCG vào thể loại game này sẽ được áp dụng chủ yếu trong thành phần (2) Generation Engine – Bộ

máy sinh nội dung nhằm sử dụng các thuật toán phù hợp để sinh ra các bản đồ, hoặc vật phẩm hợp lệ và sinh ra map phù hợp nối tiếp tuyến đường hiện có

Hình 2. 2Game thể loại Roguelike

Mô hình phần mềm áp dụng PCG cho thể loại Roguelike vẫn tuân thủ theo cấu trúc gồm bốn thành phần chính theo hình 2.1:

(1) Input Parameters – Tham số đầu vào

Các tham số đầu vào ảnh hưởng trực tiếp đến đặc tính sinh bản đồ, bao gồm:

- Seed: giá trị hạt giống để đảm bảo tính ngẫu nhiên tái sử dụng.
- Mức độ khó: ảnh hưởng đến số lượng quái vật, vật phẩm, độ rộng hành lang.
- Kích thước bản đồ: chiều cao, chiều rộng cho từng tầng (floor).
- Tần suất sinh đối tượng: số lượng vật phẩm, cạm bẫy, kẻ địch.

(2) Generation Engine – Bộ máy sinh nội dung

- **Thuật toán chủ đạo**: kết hợp **Rule-Based** và **Search-Based PCG**.
- **Chi tiết triển khai**:

o Phân chia bản đồ bằng BSP (Binary Space Partitioning) để sinh các phòng và hành lang nối liền.

o Áp dụng Rule-Based để gán vị trí cửa, vật phẩm, điểm bắt đầu/kết thúc, quy định không gian hợp lệ.

- Search-Based PCG (như Genetic Algorithm hoặc Monte Carlo Tree Search) được dùng để tối ưu hóa tuyến đường hợp lý, tạo ra cấu trúc bản đồ đảm bảo khả năng di chuyển và mức độ thử thách phù hợp.

(3) Evaluation Module – Mô-đun đánh giá chất lượng

- Đảm bảo các yêu cầu:
- Playability: người chơi có thể di chuyển từ điểm bắt đầu đến mục tiêu.
- Tính ngẫu nhiên hợp lý: không trùng lặp bản đồ qua các lượt chơi.
- Tính cân bằng: tỷ lệ xuất hiện vật phẩm – kẻ địch – bẫy phù hợp với độ khó.

(4) Generated Content – Nội dung đầu ra

- Bao gồm toàn bộ bản đồ mỗi tầng, danh sách vật phẩm, vị trí quái vật và trạng thái môi trường (cửa khóa, cạm bẫy).
- Nội dung đầu ra có thể được ghi lại hoặc tái tạo bằng cách sử dụng seed ban đầu.

So sánh ưu điểm và nhược điểm ứng dụng các thuật toán của kỹ thuật PCG với thể loại game Roguelike

Bảng 2.1 So sánh ưu điểm và nhược điểm ứng dụng các thuật toán của kỹ thuật PCG với thể loại game Roguelike

Thuật toán PCG	Ưu điểm	Nhược điểm	Mức độ phù hợp
Rule-Based	Kiểm soát chặt nội dung, dễ mở rộng quy tắc, cấu hình tùy chỉnh linh hoạt	Thiếu tính khám phá cao nếu không kết hợp thêm ngẫu nhiên hóa	**Trung bình** (phù hợp hơn cho bố cục cơ bản và ràng buộc vị trí)
Search-Based	Tạo bản đồ tối ưu theo nhiều tiêu chí, đảm bảo tính khả thi của bản đồ	Chi phí tính toán cao hơn, khó tuning tham số	**Trung bình** (phù hợp cho tạo đường đi, cấu trúc dungeon phức tạp)
Constraint-Based	Đảm bảo các điều kiện logic/pháp lý nghiêm ngặt	Thiếu khả năng sáng tạo nếu ràng buộc quá cứng	**Trung bình–Cao**
Machine Learning-Based	Học hành vi người chơi, tự cải thiện bản đồ qua thời gian	Yêu cầu dữ liệu huấn luyện lớn, khó kiểm soát nội dung cụ thể	**Thấp–Trung bình** với game độc lập nhỏ

2.2.2 Ứng dụng các thuật toán PCG tạo màn chơi tự động cho thể loại Platforms game

Platformer là một trong những thể loại game phổ biến và có lịch sử phát triển lâu đời. Đặc trưng cơ bản của platformer là nhân vật người chơi sẽ di chuyển trong một môi trường được tổ chức theo chiều ngang hoặc chiều dọc, nhảy qua các vật cản, thu thập vật phẩm và né tránh kẻ thù để đạt đến đích. Nổi tiếng nhất của dòng game thể loại này là Super Mario. Với các trò chơi platformer hiện đại, số lượng màn chơi có thể lên đến hàng trăm, với các biến thể về địa hình, bẫy, hành vi kẻ thù và vật phẩm thưởng, vì vậy việc áp dụng kỹ thuật PCG vào thể loại game này sẽ được áp dụng chủ yếu mục tiêu tạo các cấu trúc platform chính như sàn, khoảng cách nhảy, độ cao tầng từ đó tạo ra các map nối tiếp nhau, ghép lại thành chuỗi hợp lý thành các map nói tiếp nhau.

Hình 2. 3Super mario – Game dạng Platformer nổi tiếng

Dựa trên mô hình phần mềm ứng dụng kỹ thuật PCG tiêu chuẩn, hệ thống tạo sinh màn chơi platformer có thể được xây dựng với các thành phần chính như sau:

1. **Input Parameters (Thông số đầu vào):**
    - Tập luật về cấu trúc địa hình (ground, gaps, walls, slopes)
    - Các quy tắc về vị trí chướng ngại vật và vật phẩm
    - Độ khó mong muốn, seed ngẫu nhiên và chiều dài màn chơi
    - Các mẫu mẫu màn chơi nhỏ (level chunks) tái sử dụng
2. **Generation Engine (Bộ máy sinh nội dung):**
    - Áp dụng thuật toán Rule-Based PCG hoặc Search-Based PCG để tạo các cấu trúc platform chính như sàn, khoảng cách nhảy, độ cao tầng.

- Chèn chướng ngại vật, kẻ thù và vật phẩm thưởng dựa trên quy tắc tương tác
- Có thể sử dụng cấu trúc chunk-based composition, nơi mỗi đoạn màn chơi được sinh riêng và sau đó ghép nối lại theo một chuỗi hợp lý.

3. **Evaluation Module (Mô-đun đánh giá chất lượng):**
   - Kiểm tra tính khả thi của màn chơi: đảm bảo người chơi có thể hoàn thành với kỹ năng cơ bản.
   - Đánh giá độ khó thông qua số lượng và mật độ vật cản, khoảng cách nhảy, mức độ tương tác.
   - Áp dụng mô phỏng hành vi người chơi để xác thực đường đi hợp lệ (path validation).
4. **Generated Content (Kết quả đầu ra):**
   - Màn chơi đầy đủ có thể được sử dụng trực tiếp trong game engine hoặc làm nền tảng cho tinh chỉnh thủ công sau đó.

So sánh ưu điểm và nhược điểm ứng dụng các thuật toán của kỹ thuật PCG với thể loại game platformer

Bảng 2.2 So sánh ưu điểm và nhược điểm ứng dụng các thuật toán của kỹ thuật PCG với thể loại game platformer

Thuật toán PCG	Ưu điểm	Nhược điểm	Mức độ phù hợp
Rule-Based	Khả năng sinh nhanh số lượng lớn màn chơi mà vẫn đảm bảo cấu trúc hợp lý	Tính sáng tạo bị giới hạn, hoàn toàn không có sự đổi mới về cấu trúc màn chơi	**Thấp** (Không phù hợp với dạng game này)
Search-Based	Có thể sinh ra cấu trúc sáng tạo mà vẫn đảm bảo logic	Tốc độ chậm hơn do phải tìm kiếm qua không gian trạng thái	**Trung bình** (phù hợp cho sáng tạo các cấu trúc màn chơi)
Constraint-Based	Đảm bảo sinh màn chơi hợp lệ theo các ràng buộc	Thiếu khả năng sáng tạo nếu ràng buộc quá cứng	**Thấp**(Không phù hợp với dạng game này)

Thuật toán PCG	Ưu điểm	Nhược điểm	Mức độ phù hợp
Machine Learning-Based	Học hành vi người chơi, tự cải thiện bản đồ qua thời gian	Yêu cầu dữ liệu huấn luyện lớn, khó kiểm soát nội dung cụ thể	**Thấp–Trung bình** không phù hợp với game độc lập nhỏ dạng casual **Cao** với game có tệp khách hàng lớn và dữ liệu đủ nhiều

2.2.3 Ứng dụng các thuật toán PCG tạo màn chơi tự động cho thể loại Puzzle game

Puzzle game là một thể loại trò chơi điện tử đặc trưng bởi yêu cầu người chơi phải vận dụng tư duy logic, khả năng phân tích, chiến lược hoặc ghi nhớ để giải quyết các bài toán được đặt ra trong game. Một điểm chung nổi bật của puzzle game là hệ thống màn chơi (level) thường được xây dựng dưới dạng phân cấp, nơi mỗi màn chơi là một câu đố độc lập hoặc là một phần trong chuỗi các thử thách có liên kết logic với nhau. Vì vậy có thể thấy, kỹ thuật PCG khi ứng dụng vào thể loại game này sẽ được áp dụng chủ yếu mục tiêu tạo các nội dung cho các màn chơi mới sao cho đảm bảo có thể chơi, cấu trúc logic hợp lệ và có thể tăng độ khó theo từng cấp độ màn chơi

Hình 2. 4 Game giải đố - Where is my Water?

Dựa trên mô hình phần mềm sử dụng kỹ thuật **Procedural Content Generation (PCG)** gồm bốn thành phần cơ bản:

1. Tham số đầu vào (Input Parameters)

Đối với trò chơi giải đố, các tham số đầu vào có thể bao gồm:

- **Độ khó (difficulty level)**: xác định số lượng bước giải, mức độ thử thách hoặc số lượng thành phần cần tương tác.
- **Tập luật logic (Rule sets)**: quy định mối quan hệ ràng buộc giữa các thành phần trong màn chơi (ví dụ: không có nước cạnh lửa, một đường đi duy nhất, v.v.).
- **Dữ liệu trạng thái ban đầu (Initial state configuration)**: định nghĩa điểm xuất phát và mục tiêu cần đạt được.
- **Mẫu chơi của người dùng (User behavior data)**: nếu hệ thống hỗ trợ cá nhân hóa.

2. Mô-đun sinh nội dung (Generation Engine)

Mô-đun sinh nội dung trong puzzle game đóng vai trò cốt lõi để xây dựng màn chơi đảm bảo các đặc tính:

- **Cấu trúc logic hợp lệ** (ví dụ: có thể giải được bằng một chuỗi hành động duy nhất).
- **Độ phức tạp tương ứng với độ khó mong muốn**.
- **Sự đa dạng về cách sắp xếp và kiểu thử thách**.

Thuật toán được đề xuất là **Rule-Based PCG**, kết hợp với **Constraint-Based PCG**. Cụ thể:

- **Rule-Based PCG**: sinh sơ đồ màn chơi dựa trên tập hợp quy tắc cấu trúc và ràng buộc nội bộ thuật toán có thể áp dụng để tinh chỉnh màn chơi theo các tiêu chí như số bước giải ngắn nhất, nhịp độ tăng dần độ khó.
- **Constraint-Based PCG**: đảm bảo các quy tắc không bị vi phạm và giúp kiểm tra khả năng giải được của mỗi màn chơi.

3. Mô-đun đánh giá chất lượng (Evaluation Module)

Mô-đun đánh giá đảm bảo rằng mỗi màn chơi:

- Có lời giải hợp lệ và duy nhất (hoặc một số giới hạn lời giải).
- Phù hợp với mức độ khó dự kiến.
- Không gây nhầm lẫn hoặc mâu thuẫn trong logic giải đố.
- Đảm bảo sự hài hòa trong thẩm mỹ nếu có yếu tố hình ảnh đi kèm.

Phương pháp kiểm tra lời giải thường sử dụng **backtracking, search tree** hoặc **SAT solver** để xác nhận tính khả thi và tính duy nhất của lời giải.

4. Nội dung đầu ra (Generated Content)

Đầu ra là các màn chơi hợp lệ, được lưu dưới dạng JSON hoặc cấu trúc tùy biến, có thể được hiển thị trực tiếp trong trò chơi. Với kiến trúc này, các màn chơi có thể:

- Tự động sinh mới khi người chơi vượt qua màn hiện tại.
- Được cá nhân hóa theo lịch sử chơi và độ thành thạo của từng người chơi.
- Được tái sử dụng và phát triển nhanh chóng cho các trò chơi có hàng trăm màn chơi.

So sánh ưu điểm và nhược điểm ứng dụng các thuật toán của kỹ thuật PCG với thể loại game Puzzle

Bảng 2.3 So sánh ưu điểm và nhược điểm ứng dụng các thuật toán của kỹ thuật PCG với thể loại game Puzzle

Thuật toán PCG	**Ưu điểm**	**Nhược điểm**	**Mức độ phù hợp**
Rule-Based	Dễ triển khai, kiểm soát chặt chẽ logic game. Dễ tùy biến theo mức độ khó.	Không thích hợp với thiết kế cần nhiều ngẫu nhiên.	**Rất cao** (Puzzle game đặc biệt phù hợp với thuật toán Rule-Based)
Search-Based	Có thể tối ưu hóa độ khó hoặc mục tiêu cụ thể (số bước giải, độ phức tạp, thời gian hoàn thành).	Thời gian tìm kiếm có thể lâu nếu không có điều kiện tốt	**Cao** (phù hợp cho tối ưu hóa độ khó các màn chơi, thời gian hoàn thành)
Constraint-Based	Đảm bảo sinh màn chơi hợp lệ theo các ràng buộc	Yêu cầu xác định rõ ràng các ràng buộc.	**Cao**(Rất phù hợp với dạng game này)
Machine Learning-Based	Học hành vi người chơi, tự cải thiện bản đồ qua thời gian	Yêu cầu dữ liệu huấn luyện lớn, khó kiểm soát nội dung cụ thể	**Thấp–Trung bình** không phù hợp với game độc lập nhỏ dạng casual **Cao** với game có tệp khách hàng lớn và dữ liệu đủ nhiều

Có thể thấy, Puzzle games là một trong những thể loại phù hợp nhất để ứng dụng kỹ thuật PCG, đặc biệt là các thuật toán **Rule-Based**, **Constraint-Based**, và khi cần thiết có thể kết hợp **Search-Based** để điều chỉnh độ khó. Mô hình PCG giúp tiết kiệm thời gian thiết kế, đảm bảo chất lượng logic của màn chơi, đồng thời dễ dàng mở rộng quy mô trò chơi lên hàng trăm màn mà vẫn duy trì tính mới mẻ và hấp dẫn.

2.3. Phân tích lựa chọn thể loại game và thuật toán phù hợp áp dụng đề án

2.3.1 Lựa chọn thể loại game phù hợp trong đề án

Dựa vào phân tích về khả năng ứng dụng kỹ thuật PCG vào trong các thể loại cũng như về phạm vi của đề án là các mô hình thuật toán PCG và các mô hình game có các màn chơi và sẽ ứng dụng mô hình PCG để tạo các màn chơi tự động và có thể ứng dụng tạo nhiều màn chơi liên tục hỗ trợ tạo nội dung cho từng màn chơi, đề án lựa chọn trò chơi giải đố (Puzzle Games) là môi trường ứng dụng chính của kỹ thuật PCG, cụ thể là trong việc tự động tạo sinh màn chơi.

Lý do lựa chọn thể loại này bao gồm:

- **Kiểm soát tốt logic màn chơi:** Puzzle game có tính chất ràng buộc logic cao, dễ định nghĩa và kiểm soát thông qua các quy tắc thiết kế cụ thể, phù hợp với các thuật toán PCG có thể kiểm tra tính hợp lệ như Rule-Based hoặc Constraint-Based.
- **Khả năng cá nhân hóa:** Các màn chơi có thể được sinh ra tương ứng với độ khó, hành vi hoặc khả năng suy luận của người chơi, tăng mức độ phù hợp và hấp dẫn.
- **Tối ưu tài nguyên phát triển:** Việc tự động sinh hàng trăm màn chơi giúp giảm tải đáng kể chi phí thiết kế thủ công, đặc biệt đối với game mobile có yêu cầu nội dung liên tục phù hợp với hướng phát triển và hướng nghiên cứu của đề án.

2.3.2 Phân tích lựa chọn thuật toán phù hợp

Mục tiêu chính của đề án là nghiên cứu và đánh giá khả năng áp dụng các thuật toán Procedural Content Generation (PCG) trong việc tạo màn chơi tự động, đồng thời triển khai thử nghiệm với một trò chơi cụ thể nhằm kiểm chứng hiệu quả thực tế của thuật toán. Sau quá trình phân tích khả năng ứng dụng các thuật toán PCG tạo màn chơi tự động trong các thể loại game theo cấu trúc màn chơi (level-based game), em nhận thấy rằng phương pháp dựa trên quy tắc (Rule-Based PCG) tích hợp thuật toán Constraint-Based là giải pháp phù hợp nhất để áp dụng trong giai đoạn đầu phát triển cho thể loại game Puzzle.

Quyết định lựa chọn thuật toán Rule-Based được đưa ra dựa trên một số tiêu chí kỹ thuật và thực tiễn như sau:

- Hiệu quả tính toán: Phương pháp không yêu cầu xử lý khối lượng dữ liệu lớn, không cần huấn luyện mô hình học máy, do đó cho phép thực thi nhanh chóng, tiêu tốn ít tài nguyên hệ thống. Điều này đặc biệt hữu ích khi tích hợp trực tiếp thuật toán vào một trò chơi sẵn có mà không gây ảnh hưởng đến hiệu suất tổng thể.
- Khả năng kiểm soát nội dung: Rule-Based cho phép các nhà thiết kế trò chơi (game designers) chủ động xác định và điều chỉnh quy tắc sinh nội dung, phù hợp với định hướng thiết kế và mục tiêu trải nghiệm mong muốn của người chơi.
- Đảm bảo level sinh ra có thể giải được: Constraint-Based cho phép các nhà thiết kế trò chơi (game designers) chủ động đảm bảo là quy tắc sinh nội dung, phù hợp với định hướng thiết kế và kết quả level sinh ra có thể giải được
- Tính đơn giản trong trển khai: Thuật toán có thể dễ dàng được lập trình bằng bất kỳ ngôn ngữ nào mà không yêu cầu thư viện chuyên dụng. Điều này làm giảm độ phức tạp trong triển khai hệ thống và phù hợp với các nhóm phát triển quy mô nhỏ.
- Tính tùy biến cao: Các tập luật (rules) có thể dễ dàng thêm vào, chỉnh sửa hoặc mở rộng tùy theo từng thể loại game, cho phép tái sử dụng linh hoạt và thích nghi với các yêu cầu thiết kế khác nhau.

Trong khuôn khổ đề án, thuật toán Rule-Based được thiết kế và áp dụng theo quy trình cụ thể như sau:

- Xác định tập hợp quy tắc (Rule Sets): Các quy tắc được phân chia thành ba nhóm chính:
  - Quy tắc cấu trúc: Định nghĩa mối liên hệ không gian giữa các đối tượng, đảm bảo sự hợp lệ về bố cục tổng thể của màn chơi.
  - Quy tắc vị trí: Xác định vị trí hợp lệ để khởi tạo hoặc đặt các đối tượng như người chơi (player), chướng ngại vật, hoặc vật phẩm trong màn chơi.
  - Quy tắc tương tác: Thiết lập điều kiện tương tác giữa các đối tượng, đảm bảo tính logic và khả năng chơi được của màn chơi.
- Thiết kế định dạng dữ liệu cho rule: Mỗi quy tắc được định nghĩa theo một cấu trúc dữ liệu chuẩn hóa dưới định dạng JSON. Các rule này được xử lý bởi một bộ xử lý luật (rule engine) hoạt động song song trong quá trình sinh bản đồ (map generation), nhằm đảm bảo việc kiểm tra và áp dụng các quy tắc diễn ra đúng thứ tự.

- Tích hợp theo chuỗi xử lý (pipeline):

o Giai đoạn đầu: Sau khi bản đồ thô được sinh ra bằng thuật toán phân chia không gian nhị phân (Binary Space Partitioning – BSP), hệ thống áp dụng các rule cấu trúc để đảm bảo các vật thể không bị chồng lấn và phân bố hợp lý.

o Giai đoạn tiếp theo: Rule vị trí được áp dụng để xác định vị trí đặt người chơi tại khu vực đủ không gian, đồng thời đảm bảo các đối tượng không vi phạm quy tắc không gian.

o Giai đoạn kiểm thử: Rule tương tác được áp dụng để kiểm tra và xác nhận tính logic của các điều kiện tương tác, đảm bảo người chơi có thể hoàn thành màn chơi mà không gặp lỗi logic.

- Điều chỉnh quy tắc theo độ khó (Difficulty Scaling): Mỗi cấp độ khó của trò chơi sẽ tương ứng với một tập luật có độ phức tạp tăng dần. Ví dụ, màn chơi ở cấp độ cao hơn có thể bổ sung thêm quy tắc ràng buộc như số bước di chuyển tối đa, số lượng kẻ thù, hoặc yêu cầu giải đố cao hơn.

- Tái sử dụng quy tắc trong nhiều mẫu template: Các quy tắc được trừu tượng hóa và thiết kế độc lập với logic cụ thể của từng màn chơi. Nhờ đó, chúng có thể dễ dàng tái sử dụng cho các loại màn chơi khác nhau hoặc tích hợp với nhiều kiểu trò chơi có lối chơi tương đồng

Việc áp dụng phương pháp Rule-Based không những giúp đơn giản hóa quá trình sinh màn chơi mà còn cho phép mở rộng dễ dàng trong tương lai, đặc biệt phù hợp với mục tiêu phát triển các game thuộc thể loại giải đố (puzzle)

2.4. Kết luận chương

Chương 2 của đề án đã tập trung trình các nội dung lý thuyết và kỹ thuật nền tảng liên quan đến việc ứng dụng kỹ thuật Procedural Content Generation (PCG) vào quá trình phát triển màn chơi cho trò chơi điện tử từ đó đưa ra khuyến nghị về thể loại game và thuật toán PCG phù hợp với mục tiêu của đề án.

Qua đánh giá, đề án xác định:

- PCG phù hợp để mở rộng nội dung và tăng tính chơi lại cho các trò chơi quy mô lớn (roguelike, sandbox, thế giới mở) nhưng chưa tối ưu để tự động tạo màn chơi mới hoàn chỉnh.
- Đối với puzzle game, **Rule-Based PCG** đáp ứng tốt yêu cầu về logic, khả năng kiểm soát và hiệu quả triển khai.

Trên cơ sở đó, đề án lựa chọn Rule-Based PCG làm phương án tiếp cận chính, tạo nền tảng cho các bước thiết kế, thử nghiệm và đánh giá hệ thống sinh màn chơi tự động ở các chương tiếp theo.

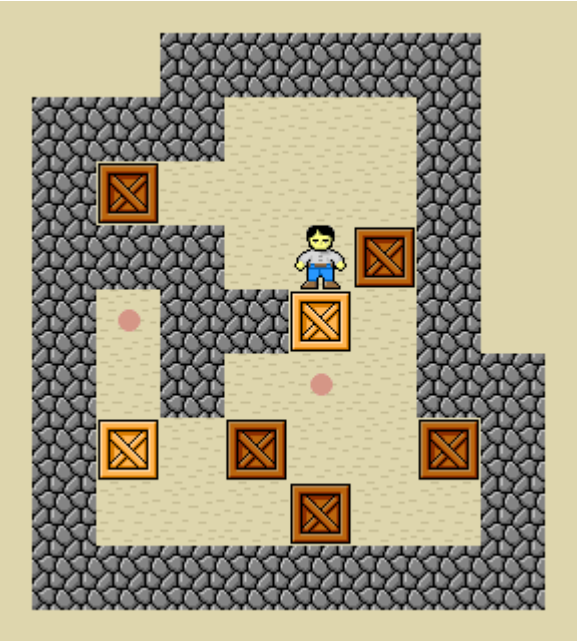
CHƯƠNG 3: TRIỂN KHAI VÀ XÂY DỰNG PHẦN MỀM

3.1.Giới thiệu game dự kiến tích hợp

Trong quá trình lựa chọn trò chơi mẫu để nghiên cứu và phát triển hệ thống tạo màn chơi tự động sử dụng kỹ thuật Procedural Content Generation (PCG),trò chơi Sokoban được lựa chọn nhờ vào tính chất đơn giản về luật chơi và tiềm năng tích hợp các thuật toán sinh nội dung tự động. **Sokoban** là một trò chơi giải đố cổ điển, trong đó người chơi có nhiệm vụ đẩy các thùng hàng đến đúng vị trí mục tiêu đã định trên bản đồ. Với cơ chế điều khiển cơ bản (chỉ di chuyển và đẩy thùng), trò chơi dễ dàng triển khai bằng các công nghệ web như HTML5, JavaScript và Canvas mà vẫn đảm bảo tính thử thách nhờ không gian chiến lược cao.

Luật chơi cơ bản:

- Mỗi lượt, người chơi có thể di chuyển hoặc đẩy 1 thùng hàng.
- Chỉ đẩy, không được kéo thùng.
- Không thể đẩy hai thùng cùng lúc.
- Trò chơi kết thúc khi tất cả thùng hàng được đặt đúng vào vị trí đích.



Hình 3.1 Game Sokoban cơ bản

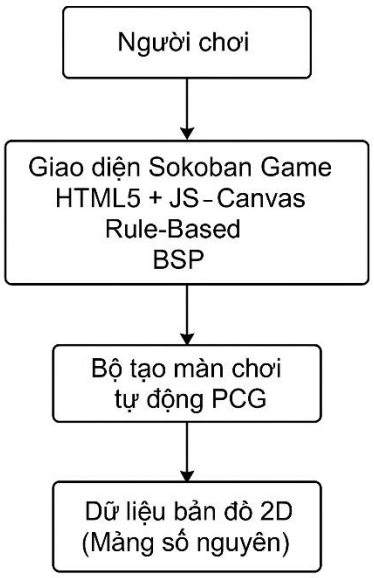
Về mặt kỹ thuật, Sokoban có cấu trúc bản đồ dạng lưới vuông 2D, số lượng đối tượng giới hạn (tường, người chơi, thùng, mục tiêu) và điều kiện chiến thắng rõ ràng, tạo điều kiện thuận lợi cho việc triển khai bằng HTML5, JavaScript và Canvas, các ngôn ngữ này cũng rất thích hợp để tích hợp các kỹ thuật PCG, cụ thể là thuật toán Rule-Based kết hợp phân hoạch không gian BSP, nhằm sinh ra các màn chơi có độ phức tạp

và khó khăn tăng dần theo từng cấp độ. Đồng thời, hệ thống còn có thể đánh giá khả năng giải được của mỗi màn chơi thông qua các thuật toán tìm kiếm đường đi như BFS, đảm bảo tính khả thi và hợp lý trong thử nghiệm. ,

Mục tiêu của chương trình khi tích hợp vào game Sokoban là xây dựng một công cụ sinh màn chơi tự động (level generator) tích hợp trực tiếp vào trò chơi Sokoban, sử dụng kỹ thuật Procedural Content Generation (PCG). Hệ thống cần đảm bảo:

- Các màn chơi sinh ra có thể giải được (solvable).
- Có sự gia tăng về độ khó qua từng cấp độ.
- Cấu trúc bản đồ thay đổi liên tục đa dạng, dựa theo cấu trúc

Sơ đồ mô hình kiến trúc tích hợp của phần mềm và game



Hình 3.2 Mô hình kiến trúc tích hợp của phần mềm và game

Các thành phần:

User: Là tác nhân tương tác trực tiếp với hệ thống trò chơi thông qua các thao tác điều khiển trên bàn phím (ví dụ: các phím mũi tên). Người chơi là đầu vào chính của hệ thống và là đối tượng trung tâm của trải nghiệm gameplay. Thông qua hành vi chơi, người dùng gián tiếp kích hoạt cơ chế chuyển level và sinh màn chơi tự động.

Sokoban Game: Đây là phần giao diện chính của hệ thống được xây dựng bằng tổ hợp công nghệ:

- **HTML5**: dùng để định nghĩa cấu trúc khung trang.
- **JavaScript**: xử lý logic trò chơi, cập nhật trạng thái bản đồ, kiểm tra điều kiện thắng/thua, và điều khiển hành động của người chơi.

- **Canvas API**: công cụ vẽ bản đồ 2D, hiển thị các ô tường, thùng, người chơi và mục tiêu.

Giao diện này đồng thời tích hợp thuật toán Rule-Based để kiểm tra và đảm bảo logic đúng đắn khi đặt vật thể, và sử dụng thuật toán BSP (Binary Space Partitioning) trong giai đoạn tiền xử lý để chia không gian bản đồ.

PCG Engine: Là thành phần cốt lõi trong kiến trúc hệ thống, có nhiệm vụ:

- Sinh ra bản đồ ngẫu nhiên (map) dựa trên các quy tắc xác định.
- Đảm bảo rằng mỗi màn chơi được tạo ra đều có cấu trúc rõ ràng, khả năng giải được và có độ khó phù hợp với từng cấp độ.

Dữ liệu map (JSON Data): Là kết quả đầu ra của quá trình sinh màn chơi, được biểu diễn dưới dạng một mảng hai chiều số nguyên, trong đó mỗi giá trị đại diện cho một loại đối tượng:

Dữ liệu này được truyền ngược trở lại giao diện Canvas để vẽ bản đồ màn chơi và xử lý logic di chuyển, tương tác trong game.

Quy trình hoạt động

- Người chơi khởi động trò chơi và tương tác với giao diện.
- Giao diện thực hiện gọi hàm sinh màn từ bộ tạo màn chơi tự động (PCG Engine).
- Bộ PCG sử dụng thuật toán BSP và Rule-Based để sinh ra dữ liệu bản đồ dạng mảng.
- Giao diện nhận dữ liệu bản đồ 2D, hiển thị trên Canvas và khởi tạo trạng thái game.
- Trong suốt quá trình chơi, nếu người chơi hoàn thành màn, giao diện sẽ gọi lại bộ PCG để sinh level tiếp theo.

3.2.Xây dựng và phát triển phần mềm tích hợp với game

3.2.1 Yêu cầu chức năng

- **Sinh màn chơi ngẫu nhiên có kiểm soát:** Phần mềm cần đảm bảo khả năng sinh ra các bản đồ màn chơi không lặp lại, nhưng vẫn tuân theo một bộ quy tắc logic để giữ tính nhất quán trong gameplay.
- **Tùy chỉnh theo tham số đầu vào:** Người phát triển game có thể cung cấp các tham số như:
    1. Kích thước bản đồ (rộng x dài)

2. Số lượng và loại thùng, số bước

- **Xác thực tính chơi được (playability):** Phần mềm cần đảm bảo rằng mọi màn chơi được sinh ra đều có thể hoàn thành được. Điều này yêu cầu phải có các bước kiểm tra logic như:
    1. Tồn tại đường đi từ vị trí xuất phát đến mục tiêu.
    2. Không có khu vực bị cách ly hoàn toàn.
    3. Không phát sinh lỗi như kẻ địch xuất hiện trong tường, bẫy không thể tránh, hoặc item không thể lấy.
- **Xuất dữ liệu dưới nhiều định dạng:** Màn chơi sinh ra cần có thể được lưu trữ và sử dụng ở nhiều môi trường phát triển game khác nhau:
    1. JSON cho game HTML5 hoặc Unity.
    2. Direct Memory Object (thay đổi trực tiếp bản đồ trong bộ nhớ game).
- **Tương tác người dùng**: Có thể thiết kế thêm giao diện đồ họa cho phép nhà phát triển:
    1. Nhấn nút tạo màn chơi.
    2. Xem trực tiếp kết quả màn chơi được sinh.

3.2.2 Yêu cầu phi chức năng

- **Hiệu suất:** Thời gian sinh màn chơi phải đảm bảo trong phạm vi 100ms - 1s để đảm bảo không làm gián đoạn dòng chơi hoặc gây trễ trong khởi động màn.

- **Tính ổn định và tin cậy:** Phần mềm không được phép sinh ra màn chơi bị lỗi (crash game, không có mục tiêu, không có người chơi, v.v.).

- **Khả năng mở rộng:** Cho phép tích hợp thêm các thuật toán PCG khác như Search-Based, Constraint-Based trong tương lai mà không cần viết lại toàn bộ phần mềm.

- **Khả năng tích hợp đa nền tảng:** Phần mềm cần hỗ trợ tích hợp vào nền tảng phổ biến hoặc có thể gọi từ server-side để sinh màn chơi động.

- **Bảo trì và nâng cấp:** Codebase rõ ràng, tài liệu đầy đủ, có sẵn cấu trúc module để dễ sửa lỗi và nâng cấp sau này.

3.2.3 Phân tích lựa chọn công cụ cài đặt hỗ trợ xây dựng phần mềm

Để hiện thực hóa mục tiêu xây dựng một hệ thống phần mềm hỗ trợ sinh màn chơi tự động dựa trên kỹ thuật Procedural Content Generation (PCG) theo phương pháp Rule-Based, em đã tiến hành khảo sát, phân tích và lựa chọn một cách có cơ sở các công

cụ, ngôn ngữ lập trình và định dạng dữ liệu phù hợp nhằm đảm bảo cả về mặt kỹ thuật lẫn khả năng mở rộng, tích hợp về sau.

Trong quá trình phát triển hệ thống, em đã lựa chọn bộ công cụ chính bao gồm HTML5, JavaScript và định dạng JSON. Đây là những công nghệ phổ biến, có mức độ tương thích cao với các nền tảng web hiện đại, đồng thời mang lại nhiều lợi thế trong quá trình xây dựng giao diện, xử lý tương tác và mô phỏng trực tiếp nội dung trò chơi.

Cụ thể, HTML5 đóng vai trò là nền tảng tạo dựng giao diện người dùng (User Interface – UI), cung cấp khả năng tương tác trực quan trên trình duyệt mà không cần cài đặt phần mềm bổ sung. HTML5 còn hỗ trợ các phần tử đồ họa như <canvas>, cho phép hiển thị bản đồ, nhân vật và các đối tượng trò chơi được tạo sinh trong thời gian thực.

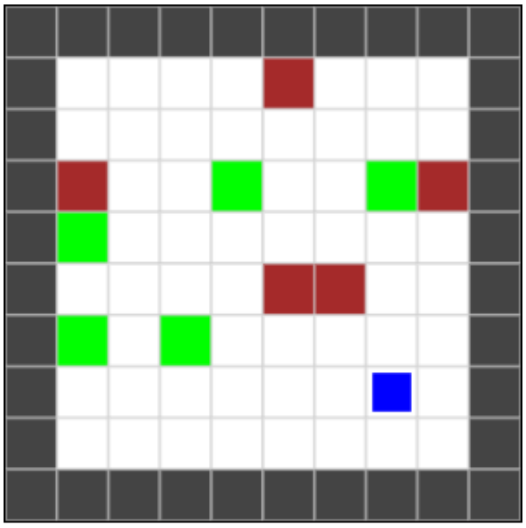
Ngôn ngữ JavaScript được sử dụng làm công cụ xử lý logic chính trong hệ thống. Với khả năng tương tác chặt chẽ với DOM và các thư viện đồ họa, JavaScript cho phép thực thi các thuật toán sinh màn chơi tự động ngay trên trình duyệt của người dùng.Ngôn ngữ này tạo điều kiện thuận lợi cho việc mô phỏng kết quả của các thuật toán PCG theo thời gian thực, bao gồm việc sinh các bố cục level mới, đánh giá tính hợp lệ và độ khó của màn chơi, cũng như theo dõi hành vi người dùng.

Việc lựa chọn HTML5, JavaScript và JSON không chỉ xuất phát từ tính phổ biến và hỗ trợ cộng đồng rộng rãi, mà còn dựa trên tính mở, linh hoạt, nhẹ và dễ triển khai của các công nghệ này. Bên cạnh đó, các công cụ này cũng hỗ trợ tốt cho mục tiêu triển khai và thử nghiệm phần mềm trực tiếp trên nền tảng Web, cho phép đánh giá khả năng áp dụng thực tế trong môi trường trình duyệt mà không cần cài đặt phức tạp.

Như vậy, nền tảng công nghệ được lựa chọn cho phép em vừa đảm bảo được tính tiện dụng, hiệu quả trong phát triển, vừa đáp ứng tốt các yêu cầu về mở rộng, tích hợp với các trò chơi khác trong tương lai.

3.2.4 Phát triển phần mềm công cụ sinh màn chơi tự động (level generator) tích hợp trực tiếp vào trò chơi Sokoban

Phát triển giao diện game Sokoban sử dụng HTML5, Canvas và Javascript



Hình 3.3 Hình ảnh giao diện màn chơi

Phát triển PCG Engine

Thuật toán chia bản đồ bằng BSP

Giải thích thuật toán:

```
function partitionBSP(x, y, width, height, depth) {
  if (depth <= 0 || width < 6 || height < 6) return [{ x, y, width, height }];

  const vertical = Math.random() > 0.5;

  const split = vertical
    ? Math.floor(width / 2) + Math.floor(Math.random() * 3 - 1)
    : Math.floor(height / 2) + Math.floor(Math.random() * 3 - 1);

  return vertical
    ? [...partitionBSP(x, y, split, height, depth - 1),
       ...partitionBSP(x + split, y, width - split, height, depth - 1)]
    : [...partitionBSP(x, y, width, split, depth - 1),
       ...partitionBSP(x, y + split, width, height - split, depth - 1)];
}
```

Hàm partitionBSP(x, y, width, height, depth) nhận vào một vùng bản đồ hình chữ nhật xác định bởi vị trí (x, y) và kích thước (width, height), cùng với tham số depth xác định độ sâu đệ quy (tức số lần tối đa vùng có thể bị chia nhỏ). Kết quả trả về là một mảng các vùng nhỏ không còn bị chia tiếp – tương đương với các "phòng" trong bản đồ game.

Khi độ sâu đệ quy đạt đến 0 hoặc vùng hiện tại quá nhỏ (rộng hoặc cao < 6 ô), thuật toán sẽ không chia tiếp nữa và trả về vùng hiện tại dưới dạng một "node" trong cây BSP. Sau đó, việc chọn hướng chia được thực hiện ngẫu nhiên. Nếu vertical là true, không gian sẽ được chia theo chiều dọc (tạo hai vùng trái – phải); nếu false, chia theo chiều ngang (tạo hai vùng trên – dưới). Điều này giúp tạo sự phong phú trong bố cục bản đồ.

Thuật toán chia tại khoảng giữa của chiều đang xét, cộng thêm một chút dao động ngẫu nhiên ±1 để tăng tính bất đối xứng của bản đồ. Điều này giúp tránh việc các phòng bị chia quá đều hoặc quá đối xứng – vốn dễ gây nhàm chán trong trải nghiệm người chơi.

Mỗi lần chia tạo ra hai vùng con, và thuật toán tiếp tục đệ quy trên từng vùng đó cho đến khi đạt điều kiện dừng.

Giải thích thuật toán Sinh màn chơi dựa vào thuật toán Rule-Based

Hàm generatePCGLevel có nhiệm vụ sinh ra một bản đồ (màn chơi) ngẫu nhiên có cấu trúc hợp lệ dựa trên một số quy tắc nhất định. Mỗi màn chơi sẽ bao gồm:

- Tường bao quanh bản đồ.
- Một vị trí người chơi.
- Một số lượng hộp (box) nhất định.
- Một số lượng vị trí đích tương ứng (goal).

Các đối tượng trong bản đồ được mã hóa theo các số nguyên như sau:

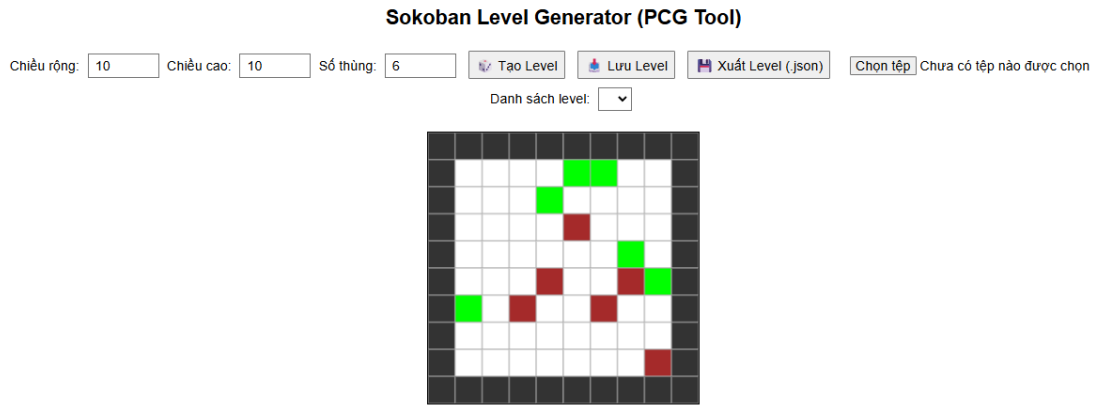
- 0: Ô trống (empty cell).
- 1: Tường (wall).
- 2: Vị trí người chơi (player start).
- 3: Hộp (box).
- 4: Vị trí đích (goal).

Sau đó, khởi tạo một mảng hai chiều có kích thước mapHeight x mapWidth, và gán giá trị mặc định là 0 cho toàn bộ ô – tương ứng với trạng thái "ô trống". Đây là bước đầu tiên để định nghĩa cấu trúc bản đồ

Các vòng lặp trên đặt giá trị 1 cho các ô ở hàng đầu, hàng cuối và cột đầu, cột cuối – tương ứng với tường bao của bản đồ. Việc này đảm bảo rằng người chơi hoặc hộp không thể thoát khỏi phạm vi màn chơi.

Một cặp tọa độ ngẫu nhiên px, py được chọn trong phạm vi bên trong tường (loại trừ viền ngoài). Vị trí này sau đó được đánh dấu bằng 2, đại diện cho vị trí bắt đầu của người chơi. Sau đó sử dụng một vòng lặp while, hệ thống cố gắng đặt boxCount số hộp vào các vị trí ngẫu nhiên không trùng lặp. Kiểm tra newMap[by][bx] === 0 nhằm đảm bảo không ghi đè lên các đối tượng đã tồn tại (như người chơi hoặc tường). Tương tự như bước đặt hộp, hệ thống đặt số lượng điểm đích bằng với số hộp. Mỗi điểm đích được đặt tại một vị trí trống khác biệt với các đối tượng đã được đặt trước đó. Dùng boxCount, rooms (tính toán số bước level trước) và hàm generateSolvableLevel () để tăng dần độ khó các màn chơi tiếp theo .

Phát triển công cụ có thể tự tạo level (theo yêu cầu đầu vào) và import vào màn chơi



Hình 3. 4 Level được tạo bằng công cụ

3.3.Triển khai thử nghiệm phần mềm và đánh giá kết quả

Hệ thống được triển khai và thử nghiệm trong môi trường thực tế để đánh giá hiệu quả. Quá trình này bao gồm cài đặt chạy phần mềm, triển khai chơi thử, hoàn thành các level ghi lại số bước thực hiện cho từng level và kiểm tra các level tự tạo để kiểm tra độ khó của từng level khi tăng dần. Việc triển khai kiểm thử hệ thống đóng vai trò quan trọng trong việc đánh giá chất lượng, độ ổn định và hiệu quả thực tế của phần mềm. Trong phạm vi đề án này, hệ thống được kiểm thử chủ yếu tập trung vào chức năng tạo

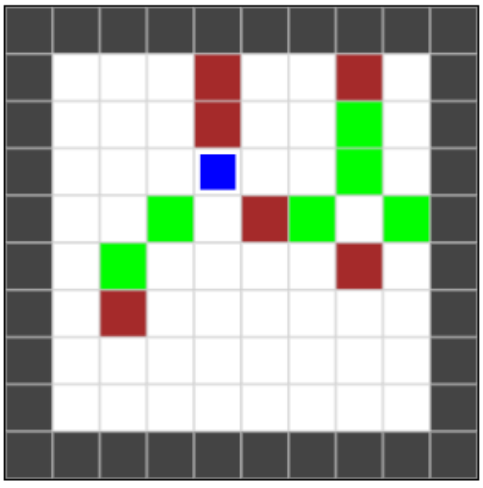
màn chơi tự động bằng kỹ thuật Procedural Content Generation (PCG) tích hợp vào trò chơi Sokoban. Các chỉ số quan trọng được đánh giá gồm:

- Tính ổn định và tính khả thi (solvability): Đánh giá tính ổn định và tính khả thi (solvability) của các màn chơi được tạo ra bằng thuật toán PCG..
- Hiệu suất xử lý Đánh giá hiệu suất thời gian sinh màn, khả năng mở rộng và khả năng duy trì chất lượng bản đồ..
- Tính chất tăng dần độ khó: Sinh liên tục 30 màn ngẫu nhiên để kiểm tra độ khó của từng màn chơi thông qua số bước và cảm nhận của người chơi.
- Đánh giá trải nghiệm người chơi (User Feedback): Mô phỏng người chơi bằng giải thuật tìm đường (BFS hoặc DFS) để thống kê số bước giải trung bình.

3.3.1 Đánh giá hiệu quả của hệ thống dựa trên kết quả thử nghiệm

Để phục vụ kiểm thử, hệ thống tiến hành sinh 30 màn chơi tự động bằng cách sử dụng phần mềm hoặc cách tạo Level bằng công cụ với số thùng tăng từ 2 đến 6. Mỗi màn chơi sẽ được đánh giá thông qua:

- Kích thước bản đồ
- Số thùng hàng
- Số mục tiêu
- Độ sâu BSP
- Số bước tối thiểu để hoàn thành (tính bằng thuật toán tìm đường)
- Mức độ đánh giá độ khó (Dễ, Trung bình, Khó)



Hình 3.5 Level tự động tạo trong trò chơi

Chi tiết đối tượng thử nghiệm: Đối tượng tham gia thử nghiệm bao gồm 10 người trong đó có 5 nam và 5 nữ. Về kinh nghiệm chơi game, nhóm thử nghiệm được chia thành ba mức: 5 người có kinh nghiệm chơi các trò chơi giải đố trên 2 năm, 3 người ở mức trung bình (1–2 năm) và 2 người mới bắt đầu chơi. Mỗi người chơi được yêu cầu hoàn thành 30 màn chơi (levels) do hệ thống PCG sinh ra, được chia thành hai buổi thử nghiệm liên tiếp (mỗi buổi kéo dài khoảng 60 phút). Trong quá trình thử nghiệm, dữ liệu liên quan đến tỷ lệ hoàn thành, thời gian trung bình, mức độ khó và phản hồi chủ quan của người chơi được ghi nhận đầy đủ để phục vụ đánh giá.

Trò chơi được trải nhiệm bởi nhóm người dùng , kết quả thử nghiệm 30 levels được đánh giá cụ thể như sau:

Bảng 3.1 Kết quả thử nghiệm 30 levels

Nhóm Level	**Số lượng thùng (box)**	**Số bước giải trung bình**	**Tỉ lệ màn chơi hợp lệ**	**Tỉ lệ màn chơi giải được**	**Đánh giá độ khó cảm nhận**
1–5	2	8–15 bước	100%	100%	Dễ
6–10	3	12–22 bước	100%	100%	Dễ – Trung bình
11–15	4	20–35 bước	100%	100%	Trung bình
16–20	5	28–45 bước	100%	100%	Khó
21–25	6	35–55 bước	100%	100%	Khó – Rất khó
26–30	6	45–65 bước	96.6% (5/6 màn hợp lệ)	83.3% (5/6 màn giải được)	Rất khó, cần có game sense tốt để giải

Kết quả thử nghiệm chi tiết:

- **Tính ổn định và tính khả thi (solvability):** Hầu hết các màn chơi được tạo ra đều đạt tính hợp lệ cao. Cụ thể, có 29/30 màn chơi hoàn toàn giải được, duy nhất một màn (ở nhóm 21–30) xảy ra hiện tượng deadlock khi 2 thùng bị đặt quá gần tường khiến việc đẩy ra mục tiêu không khả thi. Điều này cho thấy thuật toán đặt vật thể theo Rule-Based chưa hoàn toàn xử lý tất cả trường hợp "kẹt" (deadlock), đặc biệt ở mức độ khó cao khi không gian hẹp.

- **Tính chất tăng dần độ khó:** Mức độ khó được phản ánh rõ nét qua số bước giải trung bình tăng đều. Trong nhóm 1–10, trung bình chỉ cần **10–20 bước** để hoàn thành. Từ nhóm 16–30, số bước dao động từ **30 đến 65 bước**, chứng minh rằng việc tăng số thùng hàng, đồng thời tăng mật độ không gian hẹp, tạo nên độ khó thực tế
- **Hiệu suất xử lý**: Thời gian sinh một bản đồ (bao gồm phân vùng + đặt vật thể) trung bình chỉ khoảng **30–50 ms**, cho thấy hiệu suất thuật toán rất tốt, phù hợp với môi trường Web game thời gian thực. Ngay cả khi sinh liên tục 30 level, không có dấu hiệu sụt giảm hiệu năng.
- **Đánh giá trải nghiệm người chơi (User Feedback):**Mức độ khó tăng dần theo từng level, cấu trúc bản đồ có sự đa dạng, các level sẽ tạo ra các map khác nhau. Các ô nối nhau thành khu vực, ngóc ngách và tạo nhiều đường đẩy khác nhau. Điều này giúp mỗi màn chơi mang một trải nghiệm mới, tránh cảm giác lặp lại.

Phân tích ưu điểm và nhược điểm của hệ thống

Ưu điểm:

- Hệ thống sử dụng thuật toán PCG, kết hợp giữa phương pháp phân hoạch không gian BSP (Binary Space Partitioning) và kỹ thuật đặt vật thể dựa trên quy tắc (Rule-Based), cho phép tạo ra các màn chơi mới hoàn toàn mà không cần can thiệp thủ công.
- Nhờ BSP phân chia không gian thành các vùng ngẫu nhiên, bản đồ sinh ra có sự thay đổi đáng kể về bố cục mỗi lần chơi. Cùng với đó, quy tắc tăng dần số lượng thùng hàng và độ sâu phân chia không gian giúp điều khiển độ khó của game tăng dần theo cấp độ
- Toàn bộ hệ thống được xây dựng bằng HTML5, JavaScript và Canvas, hoạt động tốt trên hầu hết các trình duyệt hiện đại mà không cần cài đặt thêm. Thời gian sinh một màn chơi chỉ khoảng 30–50ms, đảm bảo hiệu suất đủ cao để áp dụng trong môi trường game thực tế và trên các thiết bị cấu hình thấp.
- Kiến trúc phần mềm được tổ chức theo mô-đun rõ ràng, giúp dễ dàng mở rộng số lượng màn chơi, thay đổi quy tắc sinh level, hoặc tích hợp vào các trò chơi khác cùng kiểu lưới.

Nhược điểm:

- Dù hệ thống sinh màn chơi dựa trên các quy tắc hợp lệ, nhưng chưa có cơ chế kiểm tra giải được (solvability) hoặc phát hiện trạng thái bế tắc (deadlock). Trong các

màn khó, đã ghi nhận có trường hợp thùng bị kẹt sát tường hoặc góc khiến người chơi không thể hoàn thành màn

- Hệ thống PCG hiện tại chỉ tăng độ khó tuyến tính thông qua tham số (số thùng, độ sâu BSP). Điều này chưa đủ linh hoạt để điều chỉnh độ khó dựa trên phong cách hoặc hiệu suất chơi thực tế của người dùng.
- Do tập trung vào thuật toán và tính năng lõi, phần giao diện (UI/UX) của hệ thống vẫn còn đơn giản, chưa có hiệu ứng đồ họa nâng cao, âm thanh, hoặc hướng dẫn người chơi rõ ràng.

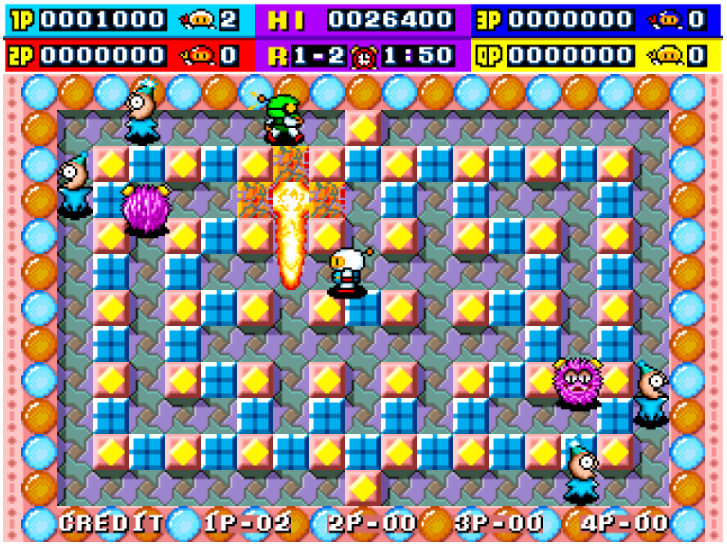
Tổng quan lại. phần mềm sinh màn chơi tự động bằng thuật toán PCG cho game Sokoban đã chứng minh được tính khả thi, hiệu suất cao và độ ổn định trong môi trường trình duyệt. Các ưu điểm nổi bật như khả năng tự động hóa, điều khiển độ khó hợp lý và khả năng mở rộng tốt cho thấy tiềm năng triển khai thực tế. Tuy nhiên để phần mềm hoàn thiện thuật toán và ứng dụng sâu rộng hơn, cần tiếp tục nghiên cứu giải pháp điều chỉnh độ khó dựa trên phong cách hoặc hiệu suất chơi thực tế của người dùng.

3.3.2 Đánh giá khả năng áp dụng hệ thống vào các game khác

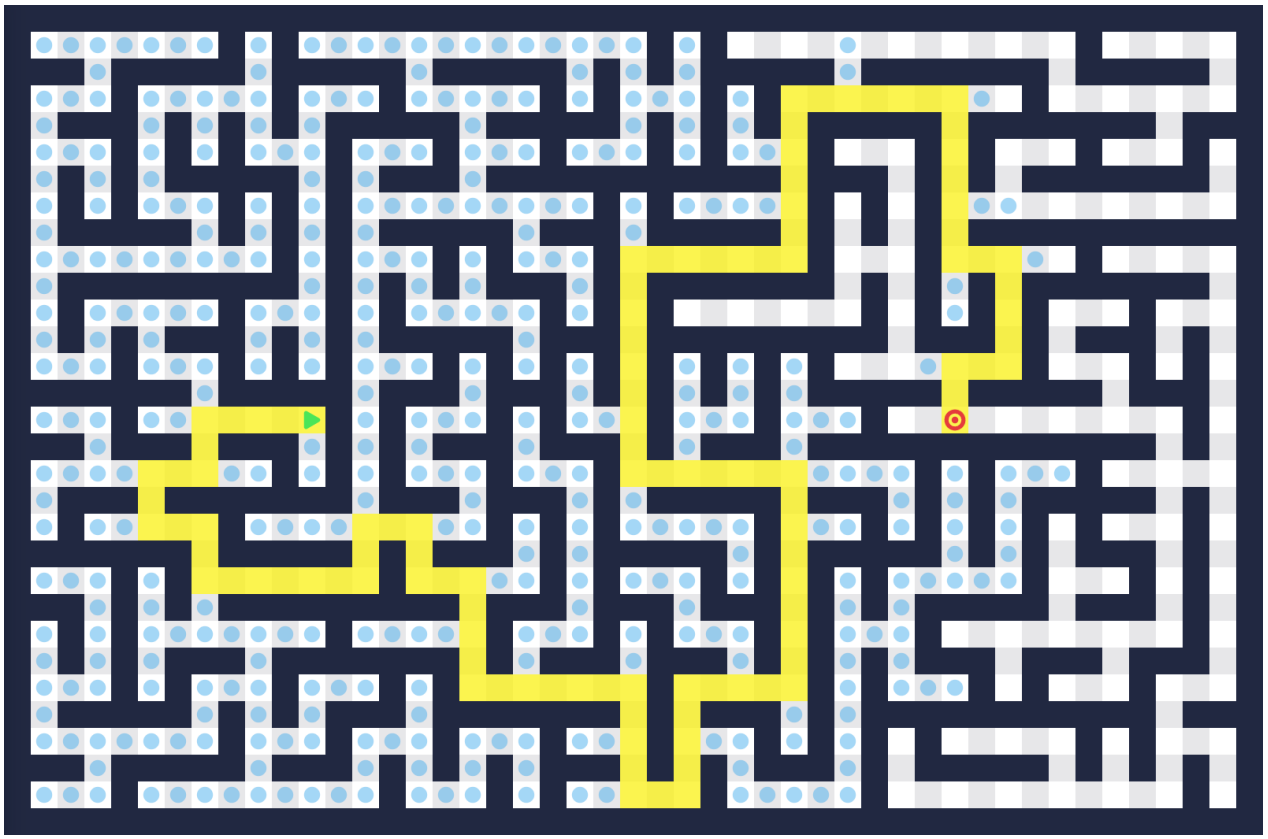
Trong bối cảnh các trò chơi giải đố (Puzzle Game) ngày càng phổ biến và phát triển mạnh mẽ, việc áp dụng một hệ thống hỗ trợ tự động sinh các màn chơi vào các trò chơi này là một bước tiến lớn, không chỉ hỗ trợ nhà phát triển mà còn mở ra nhiều tiềm năng nghiên cứu và ứng dụng trong lĩnh vực trí tuệ nhân tạo (AI). Một trong những yếu tố quan trọng nhất để đánh giá tiềm năng của hệ thống là khả năng áp dụng linh hoạt vào nhiều trò chơi khác nhau, bất kể sự khác biệt về dữ liệu đầu vào, cơ chế trò chơi hay môi trường trong từng trò chơi. Kỹ thuật tiềm năng ứng dụng rộng rãi của mô hình này đối với nhiều thể loại trò chơi thuộc nhóm puzzle game (trò chơi giải đố). Khác với các thuật toán học máy đòi hỏi dữ liệu huấn luyện lớn và mô hình phức tạp, mô hình BSP kết hợp Rule-Based có ưu điểm nổi bật là đơn giản, dễ kiểm soát, và đảm bảo tính khả thi (solvability) của level được tạo ra. Đặc biệt, mô hình này hoạt động tốt với các trò chơi sử dụng cấu trúc bản đồ dạng lưới (grid-based), nơi mà không gian chơi được chia thành các ô vuông hoặc vùng rời rạc rõ ràng.

Các trò chơi như Minesweeper, Bomberman, Tetris dạng phi tuyến, Maze Generator (mê cung ngẫu nhiên) hay các game dạng box-moving puzzles đều có thể tích hợp trực tiếp mô hình sinh nội dung này để tạo ra các màn chơi đa dạng và có độ khó tăng dần. Chẳng hạn, trong game Bomberman, BSP có thể được dùng để tạo cấu

trúc bản đồ gồm các phòng nối liền bởi hành lang, đồng thời Rule-Based sẽ xác định vị trí của người chơi, vật phẩm và chướng ngại. Đối với trò chơi Maze Generator, thuật toán này có thể sinh ra các mê cung có hướng đi rõ ràng, sau đó áp dụng Rule-Based để đặt điểm vào – điểm ra và vật cản. Ngoài ra, nếu kết hợp cùng các chiến lược phân tích độ khó (ví dụ: ước lượng độ dài đường đi, số bước giải tối thiểu), thuật toán có thể mở rộng để phục vụ nhu cầu tự điều chỉnh độ khótạo trải nghiệm chơi cá nhân hóa cho từng người dùng.



Hình 3. 6 Trò chơi Bomberman



Hình 3.7 Trò chơi Maze Generator

- **Thách thức khi áp dụng vào các trò chơi khác**

Mặc dù kỹ thuật Procedural Content Generation (PCG) sử dụng phương pháp phân hoạch không gian BSP kết hợp đặt vật thể theo quy tắc (Rule-Based) đã chứng minh được hiệu quả trong việc tạo màn chơi tự động cho trò chơi Sokoban, nhưng khi mở rộng ứng dụng vào các trò chơi puzzle khác, hệ thống phải đối mặt với nhiều thách thức kỹ thuật và logic đặc thù. Trước hết, điểm mạnh của thuật toán nằm ở khả năng sinh bản đồ dạng lưới và quy tắc định vị đơn giản, tuy nhiên nhiều trò chơi puzzle hiện nay có luật chơi phức tạp, không gian tương tác phi tuyến hoặc yêu cầu tương tác phụ thuộc trạng thái động, khiến cho các quy tắc đặt vật thể tĩnh trở nên không phù hợp

Với các game dạng puzzle không gian như Portal Puzzle, Escape Room, hay Lemmings, nội dung của màn chơi không đơn thuần là bản đồ 2D và vật thể, mà còn bao gồm cơ chế vật lý, hành vi động, điều kiện thời gian hoặc chuỗi hành động phức tạp. Việc áp dụng thuật toán BSP để chia không gian có thể tạo được bố cục ban đầu, nhưng không đảm bảo rằng chuỗi hành động có thể hoàn thành theo logic thiết kế

Ngoài ra, vấn đề khó nhất trong các game Puzzle là tính đo lường và điều chỉnh độ khó .Trong game xếp thùng như Sokoban, độ khó có thể ước lượng bằng số thùng hoặc số bước cần thực hiện, nhưng ở các trò chơi như Match-3 Puzzle, Logic Circuit Puzzle, hay Word Puzzle, độ khó có tính chủ quan cao và phụ thuộc vào ngữ cảnh người chơi. Việc gán một tham số cụ thể để điều chỉnh độ khó trong giai đoạn sinh nội dung là rất khó nếu không có hệ thống ứng dụng AI để phản hồi từ người dùng hoặc phân tích hành vi người chơi trong thời gian thực

3.3.3 Khả năng mở rộng và phát triển trong tương lai

- **Khả năng mở rộng và cải tiến**

Để đảm bảo hệ thống hoạt động hiệu quả trên nhiều trò chơi dạng Puzzles game khác nhau, cần thực hiện một số bước cải tiến và tối ưu hóa. Đầu tiên, để đáp ứng yêu cầu phức tạp của các trò chơi puzzle hiện đại, hệ thống cần được cải tiến về cả mặt cấu trúc thuật toán và năng lực kiểm thử nội dung. Cụ thể, việc bổ sung mô-đun phân tích ràng buộc logic (constraint analysis) hoặc kiểm tra giải được tự động (solution validator) sẽ giúp đảm bảo rằng các màn chơi được sinh ra không chỉ hợp lệ về mặt hình thức mà còn khả thi để người chơi hoàn thành

Ngoài ra, để phù hợp với các puzzle có nội dung phi tuyến tính hoặc không gian không đều, kiến trúc của hệ thống cần chuyển đổi từ dạng lưới vuông đơn giản sang dạng đồ thị (graph-based structure) hoặc dạng bản đồ có thể sinh theo nhiều chiều. Điều này mở ra khả năng áp dụng vào các game puzzle có yếu tố thời gian, trạng thái logic thay đổi liên tục, hoặc có hệ thống nhiệm vụ phân nhánh (multi-objective puzzle games).

Hơn nữa, hệ thống có thể nâng cấp hơn nữa bằng cách tích hợp học máy (machine learning) hoặc thuật toán tiến hóa (evolutionary algorithms) để tối ưu hóa quy trình sinh màn chơi, nhằm cá nhân hóa trải nghiệm cho từng người chơi hoặc tạo ra nội dung sáng tạo vượt ngoài khả năng thiết kế truyền thống. Các thuật toán như genetic algorithm, reinforcement learning, hay GANs có thể được huấn luyện để đề xuất bản đồ phù hợp với năng lực, thói quen và sở thích của người chơi, từ đó nâng cao tính thử thách, sự hấp dẫn và khả năng giữ chân người dùng lâu dài.

- **Tiềm năng phát triển trong tương lai**

Một trong những định hướng phát triển quan trọng trong tương lai của hệ thống là nâng cấp thuật toán sinh màn chơi từ rule-based lên các kỹ thuật PCG tiên tiến hơn như Constraint-Based hoặc Search-Based PCG. Các phương pháp này cho phép tạo nội dung có độ sâu logic cao hơn, phù hợp với các trò chơi puzzle có ràng buộc chặt chẽ về trạng thái và điều kiện giải. Việc tích hợp công cụ kiểm thử tự động để đánh giá độ khó, khả năng giải được và tính đa dạng cũng sẽ là bước đi quan trọng giúp hệ thống sinh ra các màn chơi chất lượng hơn, hướng tới khả năng tạo level theo yêu cầu cụ thể hoặc theo hồ sơ người chơi.

Bên cạnh đó, tiềm năng ứng dụng học máy (machine learning) vào quy trình sinh màn chơi cũng mở ra nhiều cơ hội đổi mới. Việc thu thập dữ liệu hành vi người chơi và sử dụng mô hình học tăng cường (reinforcement learning) hoặc các thuật toán tiến hóa (genetic algorithms) có thể giúp hệ thống tự động điều chỉnh độ khó, tối ưu hoá bố cục màn chơi, đồng thời tạo ra trải nghiệm cá nhân hoá.

Về mặt kỹ thuật, tiềm năng phát triển trong tương lai của hệ thống là khả thi, không chỉ giới hạn trong phạm vi trò chơi Puzzle game mà còn hướng tới việc trở thành một nền tảng tạo nội dung động, thông minh và thích ứng cho nhiều thể loại game và ứng dụng tương tác khác nhau.

3.4.Kết luận chương

Chương 3 đã trình bày quy trình phát triển, triển khai và thử nghiệm hệ thống sinh màn chơi tự động cho trò chơi Sokoban dựa trên kỹ thuật PCG. Nội dung bao gồm thiết kế kiến trúc tổng thể, lựa chọn công cụ lập trình, cài đặt thuật toán BSP kết hợp Rule-Based, Constraint-Based cùng mô tả chi tiết các thành phần hệ thống và mã minh họa.

Hệ thống được triển khai trên nền tảng web (HTML5, JavaScript, Canvas), đảm bảo khả năng sinh level động, lưu trữ dữ liệu và hỗ trợ thu thập thông tin kiểm thử. Các kết quả đánh giá bước đầu cho thấy hệ thống đạt hiệu suất tốt, tạo màn chơi hợp lệ, có độ khó trung bình phù hợp và đảm bảo khả năng giải được, chứng minh tiềm năng ứng dụng trong Sokoban cũng như các puzzle game có cấu trúc tương tự.

Cuối cùng, chương đã đưa ra các đánh giá ban đầu về hệ thống, bao gồm hiệu suất tạo màn chơi, tính hợp lệ của bản đồ sinh ra, độ khó trung bình và khả năng giải được màn chơi. Những đánh giá này cho thấy tiềm năng ứng dụng thực tiễn cao của hệ thống không chỉ trong trò chơi Sokoban mà còn ở các trò chơi puzzle có cấu trúc tương đương

KẾT LUẬN

Qua quá trình nghiên cứu và triển khai, đề án đã tập trung giải quyết bài toán tự động hóa tạo nội dung màn chơi trong game thông qua việc ứng dụng kỹ thuật Procedural Content Generation (PCG), góp phần tối ưu quy trình phát triển game hiện đại và nâng cao trải nghiệm người chơi. Việc ứng dụng kĩ thuật Procedural Content Generation để phát triển tự động màn chơi cho game không chỉ giúp tiết kiệm thời gian và công sức mà còn mang đến cho người chơi những trải nghiệm mới mẻ và đa dạng

Trong Chương 1, đề án đã làm rõ khái niệm, vai trò và các lý thuyết nền tảng liên quan đến PCG như lý thuyết xác suất và lý thuyết trò chơi, đồng thời khảo sát và phân tích các thuật toán tiêu biểu được sử dụng trong tạo nội dung game. PCG không chỉ giúp tiết kiệm thời gian và chi phí mà còn mang lại khả năng sinh ra nội dung phong phú, phù hợp với từng người chơi, từng lần chơi khác nhau. Việc hiểu rõ cơ sở lý luận là tiền đề quan trọng cho quá trình thiết kế và xây dựng hệ thống trong các chương tiếp theo.

Chương 2 của đề án đã tập trung trình các nội dung lý thuyết và kỹ thuật nền tảng liên quan đến việc ứng dụng kỹ thuật Procedural Content Generation (PCG) vào quá trình phát triển màn chơi cho trò chơi điện tử từ đó đưa ra khuyến nghị về thể loại game và thuật toán PCG phù hợp với mục tiêu của đề án. Từ các tiêu chí về hiệu quả triển khai, khả năng kiểm soát, mức độ phù hợp với mục tiêu nghiên cứu và yêu cầu kỹ thuật, đề án đã lựa chọn thuật toán dựa trên quy tắc (Rule-Based PCG) làm phương án tiếp cận chính để phát triển hệ thống sinh màn chơi tự động cho thể loại puzzle game.. Các kết quả nghiên cứu và triển khai ở chương này là nội dung quan trọng để bước sang giai đoạn triển khai, thử nghiệm, đánh giá và trên 1 game cụ thể.

Chương 3 trình bày việc triển khai thực tế hệ thống PCG vào một tựa game cụ thể. Dữ liệu thử nghiệm được thu thập để đánh giá hiệu quả hệ thống, từ đó giúp tinh chỉnh thuật toán và các quy tắc tạo nội dung. Kết quả cho thấy hệ thống không chỉ đạt hiệu suất sinh map tốt, mà còn có khả năng mở rộng, tùy chỉnh theo nhu cầu thực tế của nhà phát triển và phản hồi người dùng.

Với kết quả của đề án, có thể thấy đề án đã hoàn thành các mục tiêu đề ra, từ lý thuyết đến thực tiễn triển khai. Hệ thống ứng dụng kĩ thuật PCG tích hợp đã chứng minh được tính khả thi, hiệu quả và đóng góp thiết thực vào quy trình tạo các màn chơi cho trò chơi điện tử hiện đại và chứng minh tiềm năng ứng dụng cao trong ngành công

nghiệp game mà còn mở ra nhiều hướng nghiên cứu mới trong lĩnh vực trí tuệ nhân tạo và phát triển nội dung tự động

Tính mới của đề án là nghiên cứu ứng dụng kĩ thuật PCG vào trong viêc tạo các màn chơi tự động đáp ứng cho các game giải đố, khác biệt so với hướng nghiên cứu hiện nay là ứng dụng vào trong việc tạo các hoạt cảnh trong game hay tạo dựng các nội dung số hay các model 3D trong Unity/ Unreal để mở rộng bản đồ

- Kết hợp linh hoạt giữa các thuật toán PCG cổ điển và phương pháp thiết kế quy tắc hiện đại: Thay vì chỉ sử dụng một thuật toán sinh nội dung đơn lẻ, hệ thống được thiết kế để kết hợp thuật toán phân hoạch không gian (BSP) với các quy tắc tùy biến (Rule-Based Engine), tạo ra sự linh hoạt cao trong việc định hình cấu trúc và nội dung bản đồ.
- Định hướng tối ưu hóa trải nghiệm người dùng thông qua tính ngẫu nhiên có kiểm soát: Thay vì tạo ra màn chơi ngẫu nhiên hoàn toàn, hệ thống cho phép áp dụng tập luật được định nghĩa bởi nhà phát triển, giúp kiểm soát độ khó và tính hợp lý – điều mà nhiều hệ thống PCG tự động còn hạn chế.

Hướng phát triển trong tương lai là ứng dụng học máy vào quy trình sinh màn chơi cũng mở ra nhiều cơ hội đổi mới. Việc thu thập dữ liệu hành vi người chơi và sử dụng mô hình học tăng cường hoặc các thuật toán tiến hóa có thể giúp hệ thống tự động điều chỉnh độ khó, tối ưu hoá bố cục màn chơi, đồng thời tạo ra trải nghiệm cá nhân hoá. Hướng phát triển này sẽ giúp hệ thống tiến gần hơn tới các mô hình game hiện đại có tính thích nghi cao với người chơi .Về mặt kỹ thuật, tiềm năng phát triển trong tương lai của hệ thống là khả thi, không chỉ giới hạn trong phạm vi trò chơi Puzzle game mà còn hướng tới việc trở thành một nền tảng tạo nội dung tự động, thông minh và thích ứng cho nhiều thể loại game và ứng dụng tương tác khác nhau

Tổng quan, đề án đã nghiên cứu và phát triển, ứng dụng kĩ thuật PCG để tạo ra phần mềm có thể tích hợp vào trò chơi điện tử để các màn chơi tự động, đảm bảo sự đa dạng, thú vị và thách thức cho người chơi mà không cần sự can thiệp thủ công nhiều từ phía nhà phát triển

DANH MỤC CÁC TÀI LIỆU THAM KHẢO

[1] G. Smith, J. Whitehead and M. Mateas, "Tanagra: A Mixed-Initiative Level Design Tool," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201-215, 2011.

[2] N. Shaker, J. Togelius, and M. Nelson, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016.

[3] D. Braben and I. Bell, *Elite*, Acornsoft, 1984.

[4] J. Togelius et al., "The Procedural Content Generation Wiki," [Online]. Available: http://pcg.wikidot.com.

[5] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based Procedural Content Generation: A Taxonomy and Survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[6] K. Perlin, "An image synthesizer," *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[7] A. Volz et al., "Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network," in *GECCO '18*, pp. 221–228.

[8] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural Content Generation for Games: A Survey," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, pp. 1–22, 2013.

[9] C. Compton and M. Mateas, "Procedural Level Design for Platform Games," in *AIIDE*, 2006.

[10] J. Nielsen and J. Togelius, "Towards Constructive Procedural Content Generation," in *FDG*, 2011.

[11] G. N. Yannakakis and J. Togelius, "Experience-Driven Procedural Content Generation," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.

[12] Hello Games, *No Man's Sky*, 2016.

[13] Valve Corporation, *Left 4 Dead*, 2008.

[14] S. Khalifa, A. Pantaleoni, and J. Togelius, "PCGRL: Procedural Content Generation via Reinforcement Learning," in *AIIDE*, 2020.

[15] M. Guzdial and M. Riedl, "Game Engine Learning from Video," in *FDG*, 2016.

[16] C. Salge, C. Mahlmann, and J. Togelius, "Deep Interactive Evolution," in *GECCO*, 2018.
1.2
[17] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed., McGraw-Hill, 2002.
[18] Von Neumann, J., & Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton university press.
[19] Nash, J. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1), 48-49.
[20] Shoham, Y., & Leyton-Brown, K. (2009). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.
[21] Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10), 1095-1100.
[22] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
[23] Huang, M., Caines, P. E., & Malhamé, R. P. (2006). Large-population cost-coupled LQG problems with nonuniform agents: Individual-mass behavior and decentralized ϵ-Nash equilibria. *IEEE Transactions on Automatic Control*, 52(9), 1560–1571.
[24] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 9, no. 1, pp. 1–22, 2013.
[25] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.
[27] W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1, Wiley, 1968.
[28] S. M. Ross, *Introduction to Probability Models*, 11th ed., Academic Press, 2014.
[29] J. Freund and G. Simon, *Modern Elementary Statistics*, 12th ed., Pearson, 2006.
[30] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.
[31] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944.

[32] L. S. Shapley, “Stochastic Games,” *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[33] D. Fudenberg and J. Tirole, *Game Theory*, MIT Press, 1991.

[34] M. Osborne and A. Rubinstein, *A Course in Game Theory*, MIT Press, 1994.

[35] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge University Press, 2009.

[36] M. Bowling and M. Veloso, “Multiagent learning using a variable learning rate,” *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[37] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016.

[38] E. Smith, “Game Design Workflow in AAA Studios,” *Game Developer Magazine*, vol. 20, no. 4, pp. 10–15, 2018.

[39] J. Dormans, “Adventures in Level Design: Generating Missions and Spaces for Action Adventure Games,” *Proceedings of the FDG*, Monterey, CA, 2010.

[40] J. R. Parker, *Game Development Using Python and Pygame*. Mercury Learning and Information, 2011.

[41] T. Liapis, G. N. Yannakakis, and J. Togelius, “Sentient Sketchbook: Computer-Aided Game Level Authoring,” in *FDG*, 2013.

[42] A. M. Smith et al., “Tanagra: A Mixed-Initiative Level Design Tool,” in *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 2010.

[43] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Springer, 2018.

[44] A. K. Hoover et al., “Generating Levels That Teach Mechanics,” *IEEE Transactions on Games*, vol. 10, no. 2, pp. 145–155, 2018.

[45] J. Summerville et al., “Procedural Content Generation via Machine Learning (PCGML),” *IEEE Transactions on Games*, vol. 10, no. 3, pp. 257–270, Sep. 2018.

PHỤ LỤC

Phát triển giao diện game Sokoban sử dụng HTML5, Canvas và Javascript ứng dụng thuật toán PCG để tạo Level tự động

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Classic Box Pusher Game</title>
  <style>
    canvas {
      display: block;
      margin: 20px auto;
      border: 2px solid #000;
      background-color: #eee;
    }
    #status, #levelSelector {
      text-align: center;
      font-size: 18px;
      margin-top: 10px;
    }
    #levelSelector select {
      font-size: 16px;
    }
  </style>
</head>
<body>
  <canvas id="gameCanvas" width="320" height="320"></canvas>
  <div id="status"></div>
  <div id="levelSelector"></div>
```

```
<script>
  const canvas = document.getElementById('gameCanvas');
  const ctx = canvas.getContext('2d');
  const statusEl = document.getElementById('status');
  const levelSelector = document.getElementById('levelSelector');

  const tileSize = 32;
  const mapWidth = 10;
  const mapHeight = 10;
  let currentLevel = 0;

  let levels = [];
  let map = [];
  let player = { x: 0, y: 0 };

  function generatePCGLevel(boxCount = 2) {
      const newMap = Array.from({ length: mapHeight }, () => Array(mapWidth).fill(0));
    for (let x = 0; x < mapWidth; x++) newMap[0][x] = newMap[mapHeight - 1][x] = 1;
    for (let y = 0; y < mapHeight; y++) newMap[y][0] = newMap[y][mapWidth - 1] = 1;

    let px = Math.floor(Math.random() * (mapWidth - 2)) + 1;
    let py = Math.floor(Math.random() * (mapHeight - 2)) + 1;
    newMap[py][px] = 2;

    let placed = 0;
    while (placed < boxCount) {
      let bx = Math.floor(Math.random() * (mapWidth - 2)) + 1;
      let by = Math.floor(Math.random() * (mapHeight - 2)) + 1;
      if (newMap[by][bx] === 0) {
```

```
      newMap[by][bx] = 3;
      placed++;
    }
  }
  placed = 0;
  while (placed < boxCount) {
    let tx = Math.floor(Math.random() * (mapWidth - 2)) + 1;
    let ty = Math.floor(Math.random() * (mapHeight - 2)) + 1;
    if (newMap[ty][tx] === 0) {
      newMap[ty][tx] = 4;
      placed++;
    }
  }
  return newMap;
}

function isSolvable(map, player) {
  const queue = [{ x: player.x, y: player.y }];
  const visited = new Set();
  visited.add(`${player.x},${player.y}`);

  const directions = [
    [0, -1], [0, 1], [-1, 0], [1, 0]
  ];

  while (queue.length > 0) {
    const { x, y } = queue.shift();

    for (const [dx, dy] of directions) {
      const nx = x + dx;
      const ny = y + dy;
      const key = `${nx},${ny}`;
```

```
      if (
        nx >= 0 && ny >= 0 && nx < map[0].length && ny < map.length &&
        map[ny][nx] !== 1 && map[ny][nx] !== 3 &&
        !visited.has(key)
      ) {
        visited.add(key);
        queue.push({ x: nx, y: ny });
      }
    }
  }

  for (let y = 0; y < map.length; y++) {
    for (let x = 0; x < map[0].length; x++) {
      if (map[y][x] === 4 && !visited.has(`${x},${y}`)) {
        return false;
      }
    }
  }

  return true;
}

function generateSolvableLevel(boxCount = 2, maxAttempts = 100) {
  let attempts = 0;
  while (attempts < maxAttempts) {
    const map = generatePCGLevel(boxCount);
    let player = { x: 0, y: 0 };

    for (let y = 0; y < map.length; y++) {
      for (let x = 0; x < map[0].length; x++) {
        if (map[y][x] === 2) {
```

```
            player = { x, y };
            map[y][x] = 0;
          }
        }
      }

      if (isSolvable(map, player)) {
        map[player.y][player.x] = 2;
        return map;
      }

      attempts++;
    }

    return generatePCGLevel(boxCount);
  }

  function generateInitialLevels() {
    levels = [];
    for (let i = 0; i < 5; i++) {
      levels.push(generateSolvableLevel(2 + i));
    }
  }

  function updateLevelSelector() {
      levelSelector.innerHTML = '<label>Chọn màn chơi: </label><select id="levelSelect"></select>';
    const select = document.getElementById('levelSelect');
    for (let i = 0; i < levels.length; i++) {
      const opt = document.createElement('option');
      opt.value = i;
      opt.textContent = `Level ${i + 1}`;
```

```
    select.appendChild(opt);
  }
  select.value = currentLevel;
  select.addEventListener('change', (e) => {
    currentLevel = parseInt(e.target.value);
    loadLevel(currentLevel);
  });
}

function loadLevel(levelIndex) {
  map = JSON.parse(JSON.stringify(levels[levelIndex]));
  for (let y = 0; y < map.length; y++) {
    for (let x = 0; x < map[0].length; x++) {
      if (map[y][x] === 2) {
        player = { x, y };
        map[y][x] = 0;
      }
    }
  }
  drawMap();
  statusEl.textContent = `Level ${levelIndex + 1}`;
}

function drawMap() {
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  for (let y = 0; y < map.length; y++) {
    for (let x = 0; x < map[0].length; x++) {
      let cell = map[y][x];
      if (cell === 1) ctx.fillStyle = '#444';
      else if (cell === 3) ctx.fillStyle = '#a52a2a';
      else if (cell === 4) ctx.fillStyle = '#0f0';
      else ctx.fillStyle = '#fff';
```

```
      ctx.fillRect(x * tileSize, y * tileSize, tileSize, tileSize);
      ctx.strokeStyle = '#ccc';
      ctx.strokeRect(x * tileSize, y * tileSize, tileSize, tileSize);

      if (player.x === x && player.y === y) {
        ctx.fillStyle = '#00f';
        ctx.fillRect(x * tileSize + 4, y * tileSize + 4, tileSize - 8, tileSize - 8);
      }
    }
  }
}

function checkWin() {
  for (let y = 0; y < map.length; y++) {
    for (let x = 0; x < map[0].length; x++) {
      if (map[y][x] === 4) return false;
    }
  }
  return true;
}

function move(dx, dy) {
  let nx = player.x + dx;
  let ny = player.y + dy;
  if (map[ny][nx] === 1) return;

  if (map[ny][nx] === 3 || map[ny][nx] === 4) {
    let bx = nx + dx;
    let by = ny + dy;
    if (map[by][bx] === 0 || map[by][bx] === 4) {
      map[by][bx] = 3;
```

```
      map[ny][nx] = (map[ny][nx] === 4) ? 4 : 0;
    } else {
      return;
    }
  }

  player.x = nx;
  player.y = ny;
  drawMap();

  if (checkWin()) {
    statusEl.textContent = `Hoàn thành level ${currentLevel + 1}!`;
    setTimeout(() => {
      currentLevel++;
      if (currentLevel < levels.length) {
        loadLevel(currentLevel);
        updateLevelSelector();
      } else {
        const nextLevel = generateSolvableLevel(2 + currentLevel);
        levels.push(nextLevel);
        updateLevelSelector();
        loadLevel(currentLevel);
      }
    }, 1000);
  }
}

document.addEventListener('keydown', (e) => {
  if (e.key === 'ArrowUp') move(0, -1);
  else if (e.key === 'ArrowDown') move(0, 1);
  else if (e.key === 'ArrowLeft') move(-1, 0);
  else if (e.key === 'ArrowRight') move(1, 0);
```

```
  });

  generateInitialLevels();
  updateLevelSelector();
  loadLevel(currentLevel);
 </script>
</body>
</html>
```

Phát triển giao diện ứng dụng thuật toán PCG để tạo Level tự động cho Game Sokoban (Tool Generator)

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Sokoban Level Generator</title>
 <style>
  body { font-family: sans-serif; text-align: center; }
  canvas { border: 2px solid #000; margin: 20px auto; display: block; }
  input, button, select { font-size: 16px; margin: 5px; padding: 4px 8px; }
 </style>
</head>
<body>
 <h2>Sokoban Level Generator (PCG Tool)</h2>
  <label>Chiều rộng: <input type="number" id="mapWidth" value="10" min="5" max="30" /></label>
  <label>Chiều cao: <input type="number" id="mapHeight" value="10" min="5" max="30" /></label>
 <label>Số thùng: <input type="number" id="boxCount" value="3" min="1" max="10" /></label>
```

```
<button onclick="createLevel()">🎲 Tạo Level</button>
<button onclick="saveCurrentLevel()">📥 Lưu Level</button>
<button onclick="downloadJSON(savedLevels)">💾 Xuất Level (.json)</button>
<input type="file" id="loadLevels" />
<br/>
<label>Danh sách level:
<select id="levelList" onchange="selectSavedLevel()"></select>
</label>

<canvas id="canvas" width="320" height="320"></canvas>
<div id="status"></div>

<script>
const canvas = document.getElementById('canvas');
const ctx = canvas.getContext('2d');
const tileSize = 32;
let currentMap = [];
let savedLevels = [];

function generatePCGLevel(width, height, boxCount) {
const map = Array.from({ length: height }, () => Array(width).fill(0));
for (let x = 0; x < width; x++) map[0][x] = map[height - 1][x] = 1;
for (let y = 0; y < height; y++) map[y][0] = map[y][width - 1] = 1;

let px = Math.floor(Math.random() * (width - 2)) + 1;
let py = Math.floor(Math.random() * (height - 2)) + 1;
map[py][px] = 2;
let placed = 0;
while (placed < boxCount) {
let bx = Math.floor(Math.random() * (width - 2)) + 1;
```

```
    let by = Math.floor(Math.random() * (height - 2)) + 1;
    if (map[by][bx] === 0) {
      map[by][bx] = 3;
      placed++;
    }
  }
  placed = 0;
  while (placed < boxCount) {
    let tx = Math.floor(Math.random() * (width - 2)) + 1;
    let ty = Math.floor(Math.random() * (height - 2)) + 1;
    if (map[ty][tx] === 0) {
      map[ty][tx] = 4;
      placed++;
    }
  }
  return map;
}

function isSolvable(map) {
  let player = { x: 0, y: 0 };
  for (let y = 0; y < map.length; y++) {
    for (let x = 0; x < map[0].length; x++) {
      if (map[y][x] === 2) {
        player = { x, y };
        map[y][x] = 0;
      }
    }
  }
  const queue = [player];
  const visited = new Set([`${player.x},${player.y}`]);
  const dir = [[1,0],[-1,0],[0,1],[0,-1]];
  while (queue.length > 0) {
```

```
    const { x, y } = queue.shift();
    for (const [dx, dy] of dir) {
      const nx = x + dx;
      const ny = y + dy;
      const key = `${nx},${ny}`;
      if (
        ny >= 0 && ny < map.length && nx >= 0 && nx < map[0].length &&
        map[ny][nx] !== 1 && map[ny][nx] !== 3 && !visited.has(key)
      ) {
        visited.add(key);
        queue.push({ x: nx, y: ny });
      }
    }
  }
  for (let y = 0; y < map.length; y++) {
    for (let x = 0; x < map[0].length; x++) {
      if (map[y][x] === 4 && !visited.has(`${x},${y}`)) return false;
    }
  }
  return true;
}

function drawMap(map) {
  canvas.width = map[0].length * tileSize;
  canvas.height = map.length * tileSize;
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  for (let y = 0; y < map.length; y++) {
    for (let x = 0; x < map[0].length; x++) {
      let color = '#fff';
      if (map[y][x] === 1) color = '#333';
      else if (map[y][x] === 3) color = '#a52a2a';
      else if (map[y][x] === 4) color = '#0f0';
```

```
        ctx.fillStyle = color;
        ctx.fillRect(x * tileSize, y * tileSize, tileSize, tileSize);
        ctx.strokeStyle = '#aaa';
        ctx.strokeRect(x * tileSize, y * tileSize, tileSize, tileSize);
        if (map[y][x] === 2) {
          ctx.fillStyle = '#00f';
          ctx.fillRect(x * tileSize + 4, y * tileSize + 4, tileSize - 8, tileSize - 8);
        }
      }
    }
  }

  function createLevel() {
    const width = parseInt(document.getElementById('mapWidth').value);
    const height = parseInt(document.getElementById('mapHeight').value);
    const boxCount = parseInt(document.getElementById('boxCount').value);
    let map;
    let attempt = 0;
    do {
      map = generatePCGLevel(width, height, boxCount);
      attempt++;
    } while (!isSolvable(map) && attempt < 50);
    currentMap = map;
    drawMap(map);
    document.getElementById('status').textContent =
      isSolvable(map) ? `✅ Màn chơi có thể giải được (sau ${attempt} lần thử)`
: '❌ Không thể giải được.';
  }

  function saveCurrentLevel() {
    if (!currentMap || currentMap.length === 0) return;
```

```
    savedLevels.push(currentMap);
    updateLevelList();
    alert('📥 Level đã được lưu vào danh sách.');
  }

  function updateLevelList() {
    const select = document.getElementById('levelList');
    select.innerHTML = '';
    savedLevels.forEach((_, i) => {
      const opt = document.createElement('option');
      opt.value = i;
      opt.textContent = `Level ${i + 1}`;
      select.appendChild(opt);
    });
  }

  function selectSavedLevel() {
    const idx = parseInt(document.getElementById('levelList').value);
    if (!isNaN(idx)) {
      currentMap = savedLevels[idx];
      drawMap(currentMap);
       document.getElementById('status').textContent = `📋 Đang xem Level ${idx + 1}`;
    }
  }

  function downloadJSON(data, filename = 'levels.json') {
      const blob = new Blob([JSON.stringify(data, null, 2)], { type: 'application/json' });
    const link = document.createElement('a');
    link.href = URL.createObjectURL(blob);
```

```
    link.download = filename;
    link.click();
  }

  document.getElementById('loadLevels').addEventListener('change', (e) => {
    const file = e.target.files[0];
    const reader = new FileReader();
    reader.onload = function(event) {
      try {
        savedLevels = JSON.parse(event.target.result);
        updateLevelList();
        if (savedLevels.length > 0) {
          currentMap = savedLevels[0];
          drawMap(currentMap);
            document.getElementById('status').textContent = '📥 Level đã được tải!';
        }
      } catch (err) {
        alert('❌ Lỗi khi đọc JSON!');
      }
    };
    reader.readAsText(file);
  });
 </script>
</body>
</html>
```

BẢN CAM ĐOAN

Tôi cam đoan đã thực hiện việc kiểm tra mức độ tương đồng nội dung đề án tốt nghiệp qua phần mềm *https://kiemtratailieu.vn/* một cách trung thực và đạt kết quả mức độ tương đồng **5%** toàn bộ nội dung đề án tốt nghiệp. Bản đề án tốt nghiệp kiểm tra qua phần mềm là bản cứng đề án tốt nghiệp đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học Viện.

Hà Nội, ngày 20 tháng 06 năm 2025

Học viên thực hiện đề án

Trần Duy Hiếu

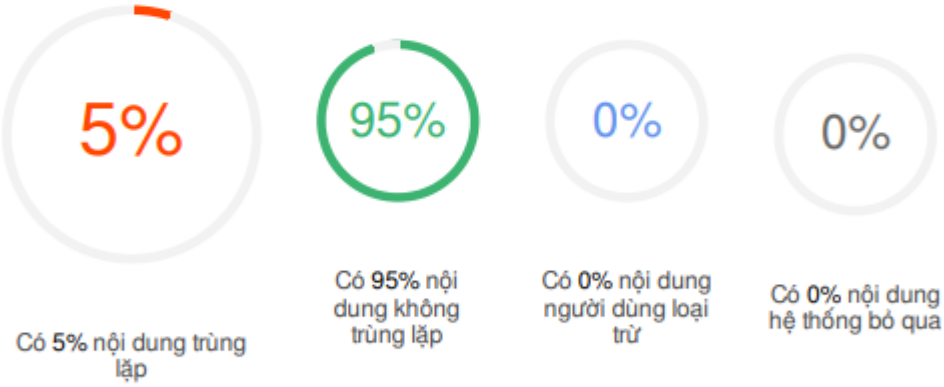
KiểmTraTàiLiệu

BÁO CÁO KIỂM TRA TRÙNG LẶP

Thông tin tài liệu

Tên tài liệu:	2024_KHMT_TranDuyHieu_DA
Tác giả:	Hieu TranDuy
Điểm trùng lặp:	5
Thời gian tải lên:	01:32 23/06/2025
Thời gian sinh báo cáo:	01:36 23/06/2025
Các trang kiểm tra:	82/82 trang

Kết quả kiểm tra trùng lặp



Nguồn trùng lặp tiêu biểu

arxiv.org 123docz.net tailieu.vn

Học viên	Người hướng dẫn khoa học
Trần Duy Hiếu	PGS. TS. Ngô Quốc Dũng