ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA CÔNG NGHỆ PHẦN MỀM

BÁO CÁO ĐỒ ÁN CUỐI KỲ

MÔN: LẬP TRÌNH TRỰC QUAN

Đề tài:

GAME MARIO BROS.

Giảng viên hướng dẫn : Phan Nguyệt Minh

Lớp : IT008.E21

Sinh viên thực hiện:

Trần Hữu Danh 12520054

Nguyễn Hữu Hiếu 12520136

Tăng Duy Khoa 12520207

Mục Lục

A. Giới Thiệu:

1. Mario

a. Lịch sử Mario:

Mario là một trong những nhân vật trong game nổi tiếng nhất thế giới. Ngày 12 tháng 9 năm 1985, công ty Nintendo đã phát hành tựa game mang tên Mario tại Nhật Bản. Cốt chuyện trong game xoay quanh Vương Quốc Nấm, trong đó có 1 anh chàng thợ sửa ống nước tên là Mario. Nhiệm vụ chính của Mario đó là phải cứu được công chúa Peach và bảo vệ Vương Quốc Nấm trước sự tấn công của con Quỷ Rùa (được lấy từ một câu chuyện dân gian của người Nhật). Trước sự thành công rực rỡ của tựa game Mario, công ty Nintendo đã liên tục phát triển các tực game Mario nổi tiếng cho đến ngày nay, các phiên bản Mario của Nintendo:

- Mario Bros.
- Super Mario Bros 1, 2, 3.
- Super Mario Land.
- Super Mario World
- Super Mario 64
- Super Mario Sunshine
- New Mario Galaxy
- Super Mario Bros.Wii

Nhân vật Mario được phát triển sao cho thích hợp với nhiều thể loại game khác nhau từ những game hành động đến những game thể thao ….. Chính vì vậy mà Mario thích hợp với mọi người chơi, không phân biệt giới tính, tuổi tác, ai cũng thấy thích nhân vật Mario.

Lấy ý tưởng từ tựa game Mairo của Nintendo, nhiều nhà phát triển đã tạo rất nhiều phiên bản Mario khác nhau cho hầu hết các nền tảng khác nhau, góp phần đưa Mario trở thành một trong những game nổi tiếng nhất thế giới.

Ở Việt Nam game Mario được biết nhiều đến từ các dòng máy chơi game NES(Nintendo Entertaiment System) và Famicom(hay còn gọi là máy chơi game 4 nút). Từ trươc khi các máy chơi game PS, PS2, PS3… trở lên phổ biến thì Mario đã trở thành một trong những game nổi tiếng và được nhiêu người chơi nhất Việt Nam.

b. Giới thiệu về đề tài Mario:

Cốt truyện: Mario là một anh chàng thợ sửa ống nước sống trong Vương Quốc Nấm, cuộc sống ở đây vốn bình yên, hạnh phúc, mọi người luôn giúp đỡ nhau, sống hòa thuận vui vẻ bên nhau. Nhưng một ngày nọ Vương Quốc Nấm bỗng xuất hiện nhiều kẻ xấu bao gồm bọn "Nấm Độc", "Rùa", "Hoa ăn thịt"…. với sự dẫn đầu của "Quỷ Rùa" tấn công vùng đất bình yên này. Tình hình trước mắt vô cùng nguy kịch, dân chúng ở đây vô cùng hoang mang lo sợ, cuộc sống của họ trước đây vốn rất hạnh phúc đầy thân thiện, không hề nghĩ đến chuyện sẽ có chiến tranh, nên không ai có thể chống lại "Quỷ Rùa"! Trước sự xâm chiếm của "Quỷ Rùa" và đồng bọn. Mario quyết định đứng lên để chống lại "Quỷ Rùa". Trên đường chống lại kẻ thù, Mario không hề đơn độc, Mario luôn có những sự trợ giúp từ các cư dân của "Vương Quốc Nấm" tiếp thêm sức mạnh cho Mario để dễ dàng chiến thắng kẻ địch và mang lại bình yên chu "Vương Quốc Nấm".

Cách chơi: Mario có 3 trạng thái sống đó là: Small < Big < Super. Khi Mario chạm vào "Nấm Lớn" hoặc "Hoa" sẽ nhận được sự trợ giúp và tăng 1 cấp trạng thái và tăng điểm. Nếu trạng thái là Super mà nhận được thì sẽ không tăng nữa.

Nếu Mario va chạm với Enemy phía "Quỷ Rùa" thì sẽ bị giảm 1 cấp trạng thái. Nếu trạng thái hiện tại là Small mà va chạm thì mất mạng. Nếu Mario giẫm lên đầu Enemy thì chúng sẽ bị tiêu diệt.

Người chơi dùng các phím “LEFT”, “RIGHT” để di chuyển Mario. Sử dụng phím “UP” để nhảy. Nếu Mario ở trạng thái “Super” thì người chơi có thể bắn ra đạn bằng cách nhấn nút “Z”. Đạn được bắn ra nếu va chạm với Enemy sẽ làm Enemy chết mà không cần Mario phải dẫm chân lên Enemy.

Mario chiến thắng khi vượt qua tất cả các Enemy và đến được lâu đài cuối mỗi bản đồ.

Map editor:

Công việc thiết kế Maps trở nên đơn giản, trực quan và sinh động hơn nhờ sự hỗ trợ của phần mềm MapEditor.

2. Mục tiêu:

a. Game:

Mục tiêu chính của Game là mang lại phút giây giải trí cho người chơi, giúp giảm thiểu căng thẳng trong học tập và làm việc. Game có cách chơi đơn giản, thú vị với hình ảnh và âm thanh vui nhộn, sinh động nhẹ nhàng, không mạng tính bạo lực cao và thích hợp cho cả nam và nữ, cho cả người lớn và trẻ em. GamePlay hấp dẫn phù hợp với cốt truyện.

Giao diện đơn giản, trực quan giúp người chơi dễ dàng điều khiển và chơi game.

Hệ thống va chạm vật lý trong game phù hợp với chân thực: gia tốc, trong lực, va chạm….

Quản lý tài nguyên game chặt chẽ giúp cho tránh việc thất thoát và trùng lặp tài nguyên sẽ làm giảm hiệu suất cửa game. Cấp phát tài nguyên trước mỗi state giúp game mượt mà không giật.

Tối ưu hóa các xử lý: Update, Draw, Collision, Phân hoạch không gian…. Nhằm tăng tốc độ xử lý hệ thống, đáp ứng được các máy tính có cấu hình yếu.

Game theo cấu trúc chuẩn, thích hợp việc làm việc nhóm, các State game hoàn toàn tách rời nhau để tránh gây lỗi dây chuyền. Thuận tiện việc nâng cấp và mở rộng.

b. MapEditor:

Mục tiêu chính của MapEditor là thiết kế map cho game một cách trực quan, sinh động và nhanh chóng để sử dụng 1 cách dễ dàng, thuận tiện. Người chơi cũng có thể tự thiết kế map cho mình chơi.Giảm bớt sự phụ thuộc việc chương trình game, dễ dàng trong việc tăng giảm độ khó game.

MapEditor phải hỗ trợ đầy đủ tất các các đối tượng, địa hình.

3. Tính năng:
   a. Game:
      i. New Game:

Nơi đây là nơi người chơi hoạt động chủ yếu, màn hình chơi được xử lý tại đây, trong main game có các thao tác chính: loadmap, tính điểm, Update, xử lý va chạm giữa các đối tượng trong game. Xử lý đồ họa, vẽ các đối tượng lên màn hình. Bên cạnh đó còn có các tính năng hỗ trợ khác:

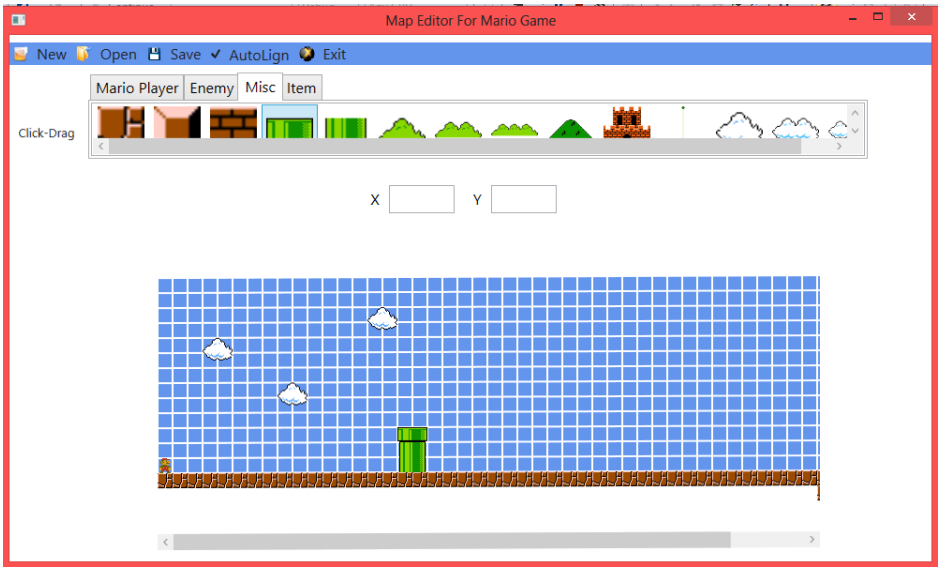
- Quản lý Resource game (hình ảnh và âm thanh): bằng design pattern.
- Quản lý Object trong game bằng cấu trúc dữ liệu QuadTree.
- Scroll map bằng Camera2D (MatrixTransform).

      ii. Option: Giúp người chơi thay đổi 2 trạng thái của game: Âm thanh hiệu ứng và Âm thanh nhạc nền.
      iii. About Us: Thông tin về nhóm và giao viên hướng dẫn môn học.
      iv. Exit Game: Thoát game khi không muốn chơi.
   b. MapEditor:

MapEditor là một công cụ để không những giúp người lập trình mà người chơi cũng có thể sử dụng để tạo ra map theo ý của riêng mình. Đa phần trong những game cần tính đa dạng và theo cốt chuyện. Thì mapeditor là một giải pháp quan trọng để làm cho người chơi thích thú hơn với những map do chính tạo ra.

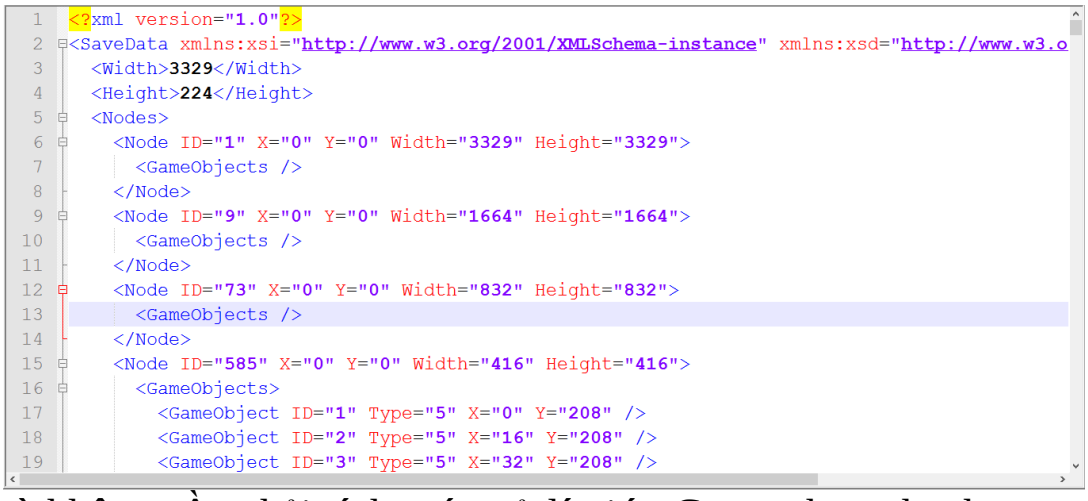
Cách thức thực hiện mapeditor:

Đầu vào mapeditor là một giao diện trực quan, có đầy đủ các object trong game và người chơi có thể tự chỉnh kích thước map, kéo thả các object vào màn chơi một cách dễ dàng như hình minh họa bên dưới



Đầu ra mapeditor là một file .xml được tổ chức để project Game có thể đọc và xử lí các dữ liệu một cách dễ dàng

Để làm tăng tính hấp dẫn và tạo sự trải nghiệm cho người chơi. Thì việc Game load lâu khi chơi là một điều cần tránh khỏi. Để làm Game không xử lí loading khi Game đang chạy. Nhóm đã quyết định xây dựng cây tư phấn trong phần build mapeditor. Và công việc của project Game chỉ cần load đọc ID để khởi tạo đối



tượng mà không cần phải tính toán xử lí giúp Game chạy nhanh hơn.

Với Node ID=”” chỉ số của các Node
X, Y là tọa độ của Node
Width, Height là kích thước của Node

GameObject là chỉ số các object tương ứng trong Game

B. Thiết Kế:

1. Kiến trúc chương trình:

a. Algorithm:

Trong này có 1 class SweptAABB giúp cho việc cài đặt và xét va chạm giữa các Object trong game được tốt hơn, tối ưu hơn trường hợp xét va chạm bằng Rectangle đơn thuần.

b. FrameWork:

- ❖ Enviroment: Trong đây chứa class GameMario.cs với các chức năng:
  - ➢ Quản lý các vấn đề liên quan đến hệ điều hành, flatform, phần cứng… Là cầu nối trung gian giữa môi trường xử lý các thiết bị nhập xuất trong game.
  - ➢ Không ảnh hưởng đến game chính nên tạo tính độc lập trong xử lý Game, thuận tiện cho việc Port Game.
  - ➢ Khởi tạo một số vấn đề tiên quyết cho Game:
    - ✓ Khởi tạo cửa sổ
    - ✓ Chọn và khởi tạo thiết bị đồ họa
    - ✓ Khởi tạo input
    - ✓ Tạo vòng lặp cho Game.
- ❖ GameState: Trong đây chứa 2 class GameState.cs và StateManager.cs với chức năng chính là quản lý sự chuyển đổi qua lại giữa các màn chơi trong game, tạo tính nhất quán, tránh xung đột code giữa các màn chơi. Dùng kiến thức hướng đối tượng và Singelton Pattern và cấu trúc dữ liệu List để thiết kế:
  - ➢ Class GameState.cs:
    - ✓ Là một class thuần ảo, không thể tạo đối tượng.

- ✓ Mỗi StateGame có 1 ID riêng, giúp dễ nhận biết và thao tác dễ dàng.
- ✓ Các phương thức đều là phương thức ảo để các class con kế thừa có chung 1 template làm việc, tạo sự dễ dàng trong việc thực hiện Update cũng như Draw các màn chơi:
  - InitState: Các khởi tạo cho màn chơi sẽ thiết lập tại đây (Âm thanh, Input, đồ họa, Resource…)
  - Update: Các Update của Game sẽ thực hiện tại đây.
  - HandleInput: Thực hiện việc Update các input của màn chơi.
  - Draw: Nhiệm vũ load hình ảnh lên màn hình Game tại State xác định.
  - ExitState: thoát khỏi State hiện tại để load State mới.

➢ Class StateManager.cs:

- ✓ Được thiết kế bằng Singleton Pattern nhằm mục đích tạo ra 1 đối tượng StateManager duy nhất trong suốt quá trình Game vận hành, để thuận tiện cho việc quản lý các màn chơi.
- ✓ Không thể nào tạo được thêm đối tượng từ StateManager.cs vì hàm tạo đã để trong chế độ protected. Tất cả thao tác với StateManager đều thông qua phương thức getInst(); và từ đó gọi

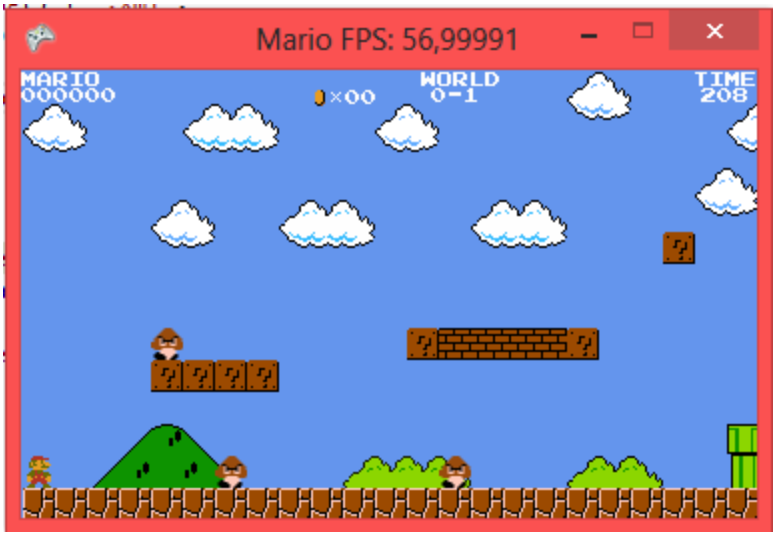
được các hàm Init, Update, Draw….
Phục vụ cho việc vận hành Game.

✓ Các phương thức:

- Init: Khởi tạo cho màn chơi vừa được Add vào ListState.
- Update: Cập nhật State hiện tại đang nằm ở vị trí vừa Add vào
- HandleInput: Cập nhật Input của State hiện tại
- Draw: Load hình ảnh của State hiện tại lên màn hình Game.
- AddScreen: Thêm 1 State vào trong ListState hiện tại.
- RemoveScreen: Remove 1 State ra khỏi List.

❖ QuadTree: Thiết kế cấu trúc dữ liệu QuadTree để phục vụ cho việc quản lý đối tượng, phân hoạch không gian 2D trong game.

Trong công việc làm Game, đối với xử lí các đối tượng là phần không hề đơn giản. Giả sử một Game, có một vài đối tượng và kích thước một map tuogw đối nhỏ, thì không cần quan tâm đến với việc xử lí update các nhân

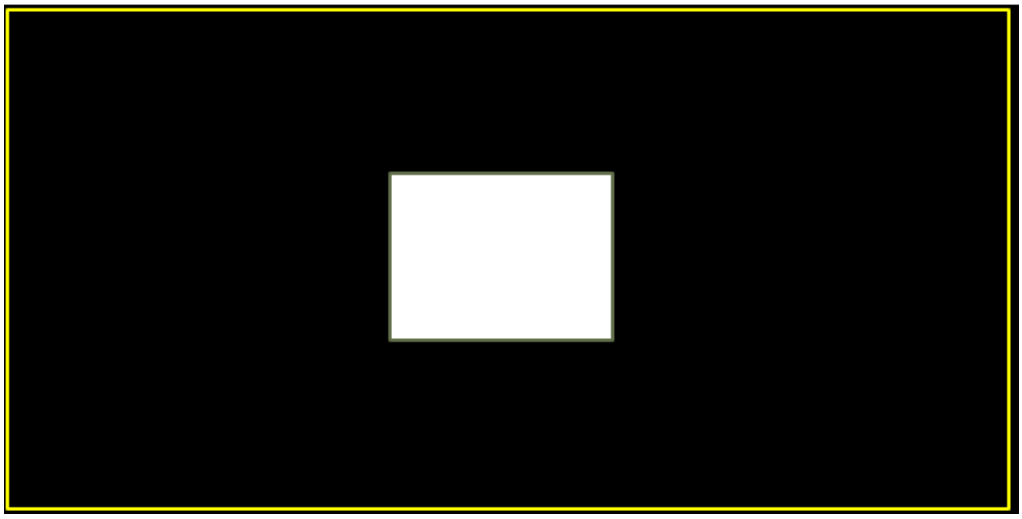


vật. Giả sử, một game có kích thước map 800 * 460 pixel thời gian xử lí sẽ nhanh hơn một game có kích thước 2000 * 2000 là điều hiển nhiên. Vấn đề đặt ra là làm thế nào để quản lí việc update các đối tượng thật hiệu quả. Tư tương chung cho các phương pháp quản lí đối tượng (hay còn gọi với tên chuyên ngành là phân hoạch không gian).Hiện nay có rất nhiều các phương pháp phân hoạch không gian, tuy nhiên ý tưởng chung cho các phương pháp này là vấn đề xử lí các đối tượng thật sự hiệu quả trong Game. Trong quá việc lập trình Game, ta có thuật ngữ viewport, viewport là phần nhìn thấy thế giới Game qua con mắt người chơi.
Phân hoạch không gian là công việc xử lí logic nhằm phục vụ cho việc update, draw các nhân vật chỉ nằm trong khu vực viewport (để làm tăng hiệu suất và tiết kiệm được tài nguyên cho việc xử lí các đối tượng không nằm trong viewport – là những đối tượng người chơi không nhìn thấy). Có một bài toán cho thấy lợi ích của phương pháp phân hoạch không gian. Giả sử một game có kích thước map là 2000 * 2000 có khoảng 1000 objects. Tuy nhiên chi mỗi viewport chỉ thể hiện 10 objects thì thật lãng phí khi xử lí một lúc 1000 objects. Với phân hoạch không sẽ update và draw khoảng 10 – 15 objects (trong trường hợp sai số trong tính toán)

Có nhiều phương pháp phân hoạch không gian, mỗi phương pháp đều có ưu và nhược điểm khác nhau. Từ tài liệu và kinh nghiệm, nhóm chỉ đề cập về phương pháp phân hoạch không gian quadtree (phương pháp phân hoạch cây tứ phân).

Ý tưởng thực hiện:

- Quadtree là một cấu trúc dữ liệu, gồm một node gốc và bốn node lá. Phân hoạc không gian quadtree là chia màn hình khoảng bốn phần. Sử dụng đệ quy để thực hiện việc xây dựng cây. Giả sử một map có kích thước là 2000 * 2000 và kích thước viewport là 800 * 600. Thì kích thước của mỗi node được xây dựng là 900 *



700 (lớn hơn kích thước viewport một chút)

Hình chữ nhật màu trắng, là kích thước của một viewport. Màn hình ngoài màu đen là kích thước của map.

Với quadtree sẽ phân hoạch trong màn hình thành nhiều phần. Được tổ chức với các node chứa một list objects và mỗi node có kích thước nhất định.

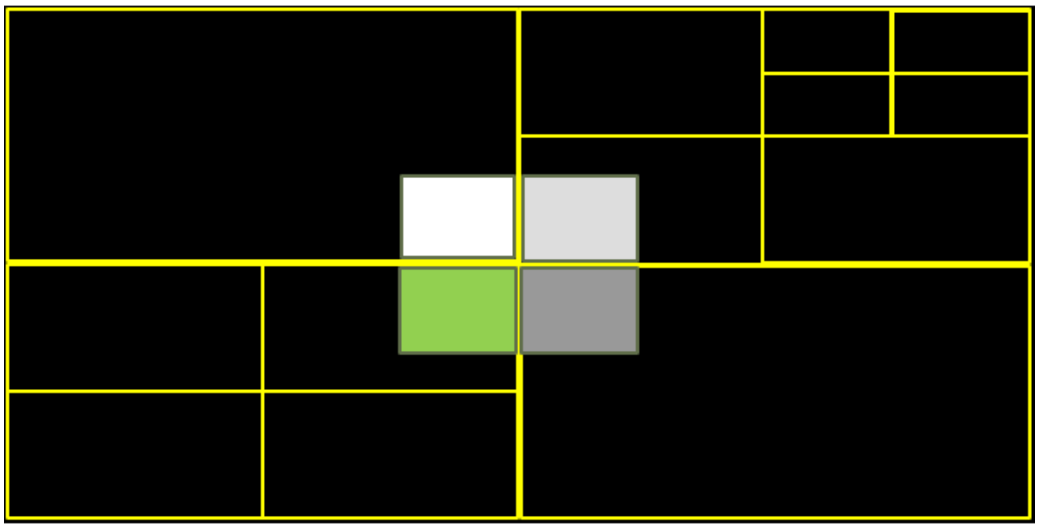
Cấu trúc của một Node

```
public class Qnode
{
  private int m_ID;
  // phạm vi của một node gồm tọa độ và kích thước
```

```
  private Rectangle bound
  // danh sách các đối tượng nằm trong Node
   private List<QGameObject> listObjects;

}
```

Sau khi thực hiện việc phân hoạch không gian, thế giới game sẽ được phân hoạch với hình minh họa phía dưới



- ***Thuật toán sử dụng:***
  Phân hoạch không gian tứ phân (quadtree) sử dụng đệ quy để xây dựng cây tứ phân.

  Điều kiện để 1 node có thể được chia tiếp là hình vuông bao node đó phải có kích thước lớn hơn hoặc bằng kích thước quy định sẵn (thường là lớn hơn viewport 1 tý) , thứ 2 là node đó phải có chứa đối tượng , đối tượng ở đây có là Ctreeobjec chứ không phải là GameObject. Và đây cũng chính là điều kiện dừng của hàm đệ quy này.

- Demo (do đây là bài báo cáo lập trình win, nên phần code sẽ được minh họa dưới dạng đồ án, nhóm không nói sâu hơn quá trình xây dựng quadtree)

Các class khác

- CResoureManager.cs: Dựa trên kiến thức hướng đối tượng và 2 loại Design Pattern tự tìm hiểu được là: Singleton Pattern và Factory Pattern. Tạo một đối tượng duy nhất của CResoureManager.cs để quản lý tất cả các tài nguyên hình ảnh của Game.
  - ✓ Điều đặc biệt của class này chính là hàm tạo được đặt trong private nên không thể chủ động tạo đối tượng của class từ bên ngoài.
  - ✓ Thể hiện static duy nhất của class được trả về trong phương thức GetInstance(). Thể hiện này chỉ tạo 1 lần duy nhất trong suốt quá trình vận hành Game. Đại diện cho hiện diện của *ResourceManager*.
  - ✓ Các Resource được xem như là thuộc tính của class để truy xuất thông qua đối tượng static được trả về trong GetInstance().
  - ✓ Phương thức Init() làm nhiệm vụ Load cho tất cả các Resource để sẵn sàng cho phương thức GetResource theo ID.
  - ✓ Khi khởi tạo chương trình ta cần chạy phương thức Init trước khi vào GameLoop.
- SoundManager.cs: Cũng với một kiến thức tương tự kiến thức dùng trong class CResoureManager.cs, nhưng có phần biến tấu để phù hợp với cách sử dụng.

- ✓ Trong class này định nghĩa 2 kiểu dữ liệu enum: ESong và ESound. ESong làm nhiệm vụ tổng hợp những ID của những Song trong game để làm nhạc nền. ESound tổng hợp những ID của Sounds Effect trong game.
- ✓ Phương thức static LoadContent để load âm thanh cần cho Game. Phương thức này cần thực hiện ngay trong phần Init của GameLoop.
- ✓ Phương thức PlaySound, PlaySong là phát âm thanh theo ID có sẵn trong phần tham số.

➢ CSprite.cs, CAnimation.cs: 2 class này phối hợp quản lý hiệu quả Texture2D, hiện thực các chuyển đông Animation và các hiệu ứng khác…

- ✓ Một số thuộc tính quan trọng trong class CSprite.cs và CAnimation.cs:
  - Texture2D m_Texture: lưu trữ hình ảnh từ bitmap
  - int m_FrameStart, m_FrameEnd, m_CurFrame: phục vụ cho việc vẽ chuyển động Animation
  - Vector2 m_Position, m_Origin, m_Scale: cập nhật vị trí, tâm xoay
  - SpriteEffects m_Effect: xử lý trường hợp lật gương của ảnh.
  - … và các thuộc tính khác
- ✓ Phương thức của CSprite.cs:

- Các hàm tạo: Hàm tạo mặc định, hàm tạo sao chép, hàm tạo có đối số thích hợp.
- Phương thức Init hiện thực được quá trình LoadContent cho Texture2D cần thiết.
- UpdateAnimation : Cập nhật chuyển động Animation.
- Update, Draw: Cập nhật trạng thái và vẽ hình.

➢ CObject.cs, CAnimationObject.cs: 2 class này đảm nhận việc làm nền tảng cho tất cả các đối tượng được tạo ra trong Game.

✓ CObject.cs là class tổng hợp những thuộc tính căn bản nhất của một đối tượng cần có: m_IDObject, m_Sprite, m_Position, m_Status…

- IDObject m_IDObject: Mỗi Object có 1 ID riêng để nhận biết. Dùng để xét va chạm và load hình cho Object đó trong Game.
- CSprite m_Sprite: lưu trữ Sprite hình của đối tượng.
- IDStatus m_Status: Lưu trữ trạng thái của Object
- Vector2 m_Position, m_StartPositon: là vị trí của đối tượng và vị trí lúc bắt đầu.
- Các hàm tạo: Hàm tạo có đối số, hàm tạo sao chép.

- Các phương thức Update: UpdateAnimation: để Update Animation cho Object, UpdateCollision: Để Update va chạm cho Object, UpdateMovement: Update di chuyển
- Phương thức Draw: Vẽ Object lên màn hình Game

✓ CAnimationObject.cs kế thừa từ CObject.cs để thêm các thuộc tính giúp ích cho việc di chuyển, va chạm và Animation. Các thuộc tính: m_Velocity, m_Accel, m_GameTime, m_Direction giúp ích cho việc di chuyển và va chạm cũng như cập nhật trạng thái.

- int m_FrameStart, m_FrameEnd, m_CurFrame, m_TimeAnimation: Những thông số hỗ trợ cho việc vẽ Animation Game.
- Phương thước UpdateAnimation: Mô phỏng chuyển động Animation cho Object.

➢ CCamera2D.cs : class này quản lý việc Scroll map bằng MatrixTransform.

✓ Một số thuộc tính của class CCamera2D:

- Vector2 m_Position: Vị trí của Camera

- Vector2 viewSize: kích thước của Camera

✓ Một số phương thức của class CCamera2D

- Hàm tạo: cài đặt một số thông số cần thiết cho CCamera2D: viewSize, m_Position
- Phương thức Update: Thực hiện việc cập nhật CCamera2D theo Object

➢ CInput.cs: class thực hiện việc cập nhật các xử lý liên quan đến Input trong Game.

✓ Một số thuộc tính:

- KeyboardState m_CurKey, m_LastKey: Các trạng thái của bàn phím

✓ Một số phương thức

- Phương thức khởi tạo CInput: Khởi tạo trạng thái của bàn phím
- bool KeyReleased(Keys key): xác định trạng thái phím key có đang Release không?
- bool KeyPressed(Keys key): xác định trạng thái phím key có đang Pressed không?
- bool KeyDown(Keys key): xác định trạng thái phím key có đang Down không?

➢ GlobalSetting.cs, GlobalValue.cs: Gom nhóm tất cả các cài đặt, các giá trị cần thống nhất và

dùng ở bất kỳ vị trí nào trong chương trình. Ta cần đến 2 class này.

c. Map:

Map là thế giới của Game. Với map, người chơi có thể nhập vai vào một chú Mario để tiêu diệt các enemy dựa theo cốt truyện. Các map được tổ chức và tạo ra từ mapeditor (một công cụ, người chơi có thể tự sáng tạo)

d. Object:

Nơi đây tổng hợp tất cả các thiết kế các class Object có trong Game.

❖ Buttons: Các nút chọn trong các State Menu, Option…

❖ Character: Đối tượng chính của Game đó là Mario và đạn của Mario

❖ Enemy: Các Enemy của phe "Quỷ Rùa" được thiế kế trong class này.

❖ Item: Các hỗ trợ như: Hoa, Nấm lớn, Nấm tưởng mạng, Ngôi sao bất tử, tiền

❖ Miscellaneous: Chứa các đối tượng còn lại để xấy dựng nên thế giới Game sinh động.

e. State:

Tập hợp tất cả các class thể hiện cho mỗi StateGame khác nhau. Mỗi StateGame là 1 class được kế thừa từ class StateGame, khi cần chạy 1 StateGame cụ thể chỉ cần gọi hàm Update.

❖ MainGame: Hầu hết các hoạt động của Game sẽ thực hiện tại đây.

➢ Các thuộc tính chính của State MainGame;

✓ Mario Mario: Object chính của game. Đi xuyên suốt game để chiến đấu với Enemy.

- ✓ CCamera2D cam: đối tượng CCamera2D để scroll map theo đối tượng Mario.
- ✓ List<CAnimationObject> list: Một danh sách các Object cần có trong game: Enemy, các phụ trợ Item, và các Object tĩnh làm sinh động thế giới Game.
- ✓ CInput m_input: Cập nhật InputState cho các sự kiện trong Game.
- ✓ QuadTree quadTree: Khai báo một cấu trúc dữ liệu QuadTree để quản lý không gian hiển thị trong Game.
- ✓ CMap map: dùng để Load Map vào game.

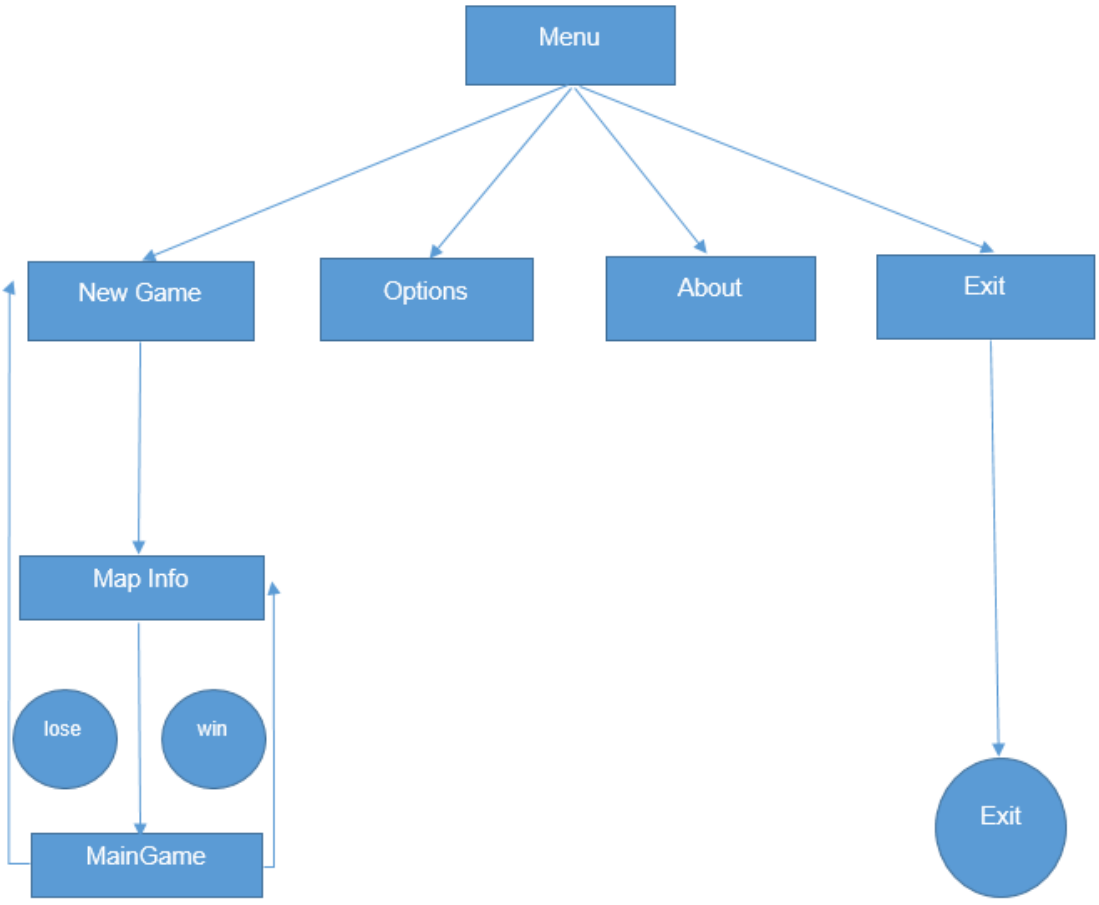
➢ Một số phương thức:

- ✓ InitState: Khai báo mọi thứ chuẩn bị để Game khởi hành. Thực hiện việc Load Map, tạo đối tượng Mario, khởi tạo đối tượng QuadTree, khởi tạo Input cho Game.
- ✓ HandleInput: Thực hiện việc cập nhật các Input trong MainGame, cập nhật các trạng thái của Mario theo Input. Thực hiện các Update của các Object khác.
- ✓ Update: Thực hiện việc Update chung.
- ✓ Draw: vẽ các đối tượng trong Game lên màn hình.

❖ Những class State còn lại: Những Class này thiết kế khá đơn giản, chủ yếu để thể hiện tính đầy đủ và đẹp của Game. Logic tương tự như MainGame.

2. GameFlow:

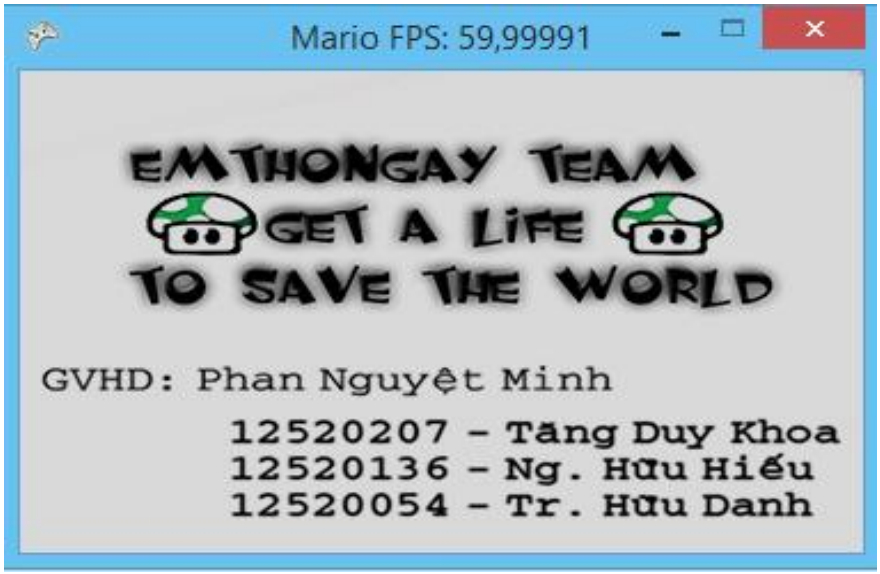
GameFlow là mô hình thể hiện mối liên hệ giữa các StateGame, GameFlow giúp quan sát, thấy được vị trí của mỗi StateGame để thuận tiện cho công việc làm việc nhóm.



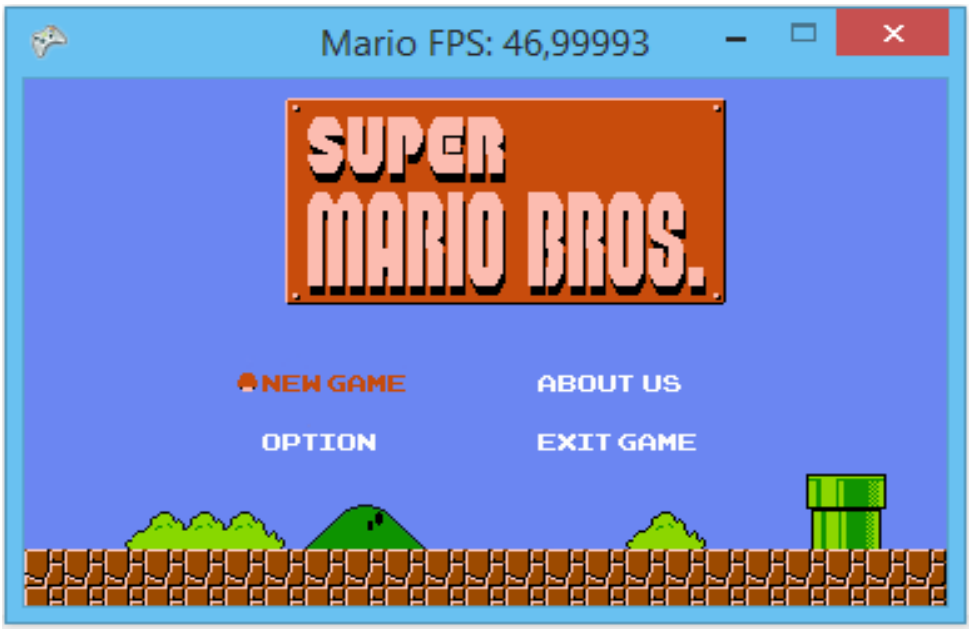
3. Giao diện:

a. Game:

❖ Intro: Là một StateGame nhỏ dùng để mở đầu sau đó vào MenuState. Giới thiệu sơ qua nhóm và tên đồ án.

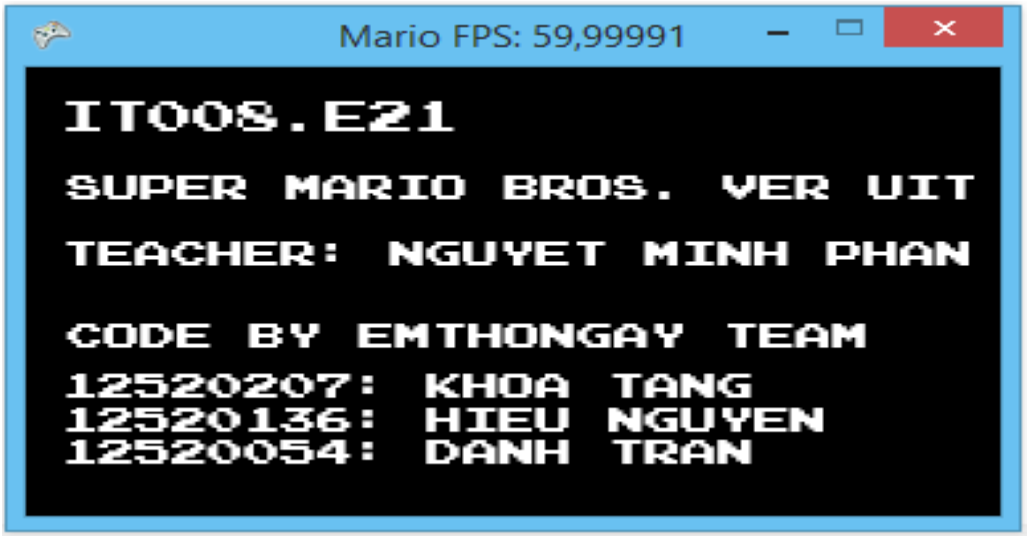


❖ MenuState: Nơi đây có những lựa chọn để đưa người chơi sang các State khác. Người chơi sử dụng các phím LEFT và RIGHT để chuyển icon sang cách vị trí tên StateGame mà mình muốn chuyển đến, sau đó nhấn SPACE để chọn di chuyển đến State đó.



❖ OptionState: Nơi đây cho những cài đặt về âm thanh gồm âm thanh nền và âm thanh hiệu ứng. Nếu muốn tắt âm thanh nào, người chơi chỉ cần di chuyển icon về dòng đó rồi nhấn SPACE. Khi cần trở về Menu thì chọn vào Return.

❖ AboutState: Nơi đây có thông tin về nhóm và tên đồ án + giáo viên hướng dẫn. Khi cần trở về MenuState người chơi nhấn phím BackSpace.



❖ MainGameState: Là màn chơi chính trong Game,

người chơi sẽ đóng vai mình là một chú Mario để giải cứu công chúa.

❖ GameOverState: Nếu người chơi sử dụng ba mạng của chú Mario, người chơi phải bắt buộc chơi lại

❖ WinGameState: Nếu trải qua ba màn chơi và tiêu

diệt được boss để vào lâu đài giải cứu công chúa

C. Cài đặt thử nghiệm:

1. Thử nghiệm thứ nhất:

a. Cấu hình máy thử nghiệm:

- ❖ CPU: Core2 Duo E7400 2.8Ghz
- ❖ RAM: 2G
- ❖ VGA: ATI Radeon HD 5450(512mb)
- ❖ OS: Windows 7 x32

b. Kết quả:

- ❖ FPS: Duy trì ở mức 60

- ❖ CPU: Khoảng 20%
- ❖ RAM khoảng: 20 – 23MB

2. Thử nghiệm thứ hai:
   a. Cấu hình máy thử nghiệm:
      - ❖ CPU: Core 2Dou T5670 1.8Ghz
      - ❖ RAM: 2G
      - ❖ VGA: onboard
      - ❖ OS: Windows 7 x32
   b. Kết quả:
      - ❖ FPS: Duy trì 60
      - ❖ CPU: Khoảng 22%
      - ❖ RAM khoảng: 20MB
3. Đánh giá:

Game sử dụng ổn định và duy trì fps 60, không yêu cầu có card màn hình rời, đáp ứng nhu cầu của nhiều máy khác nhau. Khả năng xử lý tốt, ít lỗi trong quá trình chạy Game.

4. Cấu hình đề xuất:
   a. Phần cứng:
      - ❖ CPU core dual 1Ghz
      - ❖ RAM: 1G
      - ❖ VGA: onboard
   b. Phần mềm:
      - ❖ OS: Windows 7
      - ❖ .Net FrameWork 4.0
      - ❖ XNA 4.0 Redist

D. Kết luận và hướng mở rộng:

1. Ưu điểm:
   - ❖ Game chạy tốt ở nhiều máy có cấu hình yếu.
   - ❖ Đồ họa trong Game gần gũi giống như phiên bản đầu tiên của Mario Bros. giúp người chơi hồi nhớ lại kỷ niệm đẹp của tuổi thơ.

- ❖ GamePlay đơn giản, không có hình ảnh bạo lực, mang đậm phong cách hoạt hình, thích hợp mọi lứa tuổi.
- ❖ Có thêm Mapeditor hỗ trợ trong tạo Map, giúp cho làm Map sinh động, trực quan, độc lập.
- ❖ Game được viết theo cấu trúc FrameWork, thuận tiện cho việc nâng cấp và phát triển Game sau này.

2. Khuyết điểm:
   - ❖ Xử lý va chạm còn vài lỗi nhỏ, trong Game còn 1 số lỗi nhỏ về va chạm và di chuyển.
   - ❖ Yêu cầu phần mềm phải có .Net framework 4.0 và XNA 4.0
   - ❖ Mapeditor còn đơn giản, chưa có khả năng bắt ngoại lệ nhiều.
3. Mở rộng:
   - ❖ Thêm nhiều Map mới với nhiều GamePlay khác nhau để tạo sự mới mẻ trong Game.
   - ❖ Thay đổi Resource qua từng màn chơi để tạo sự mới mẻ.
   - ❖ Sử dụng lại FrameWork này để phát triển game Nấm Lùn đa màu sắc hơn.
   - ❖ Thêm chức năng SaveGame, LoadGame.
   - ❖ MultiPlayer qua mạng LAN.