ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

---------- ꕥ ----------

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: CÔNG NGHỆ JAVA

Đề tài: Lập trình game 2D chủ đề giải đố

Giảng viên: Đào Thị Lệ Thủy

Thành viên: 221230771 – Phạm Tiến Dũng

Hà Nội, 20 tháng 03 năm 2024

ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

---------- 🕮 ----------

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: CÔNG NGHỆ JAVA

Đề tài: Lập trình game 2D chủ đề giải đố

Giảng viên: Đào Thị Lệ Thủy

Thành viên: 221230771 – Phạm Tiến Dũng

Hà Nội, 20 tháng 03 năm 2024

LỜI NÓI ĐẦU

Hiện nay, toán học được coi là nền tảng của sự phát triển. Hàng triệu người từ các nền văn hóa khác nhau đều nỗ lực trong môn học này. Ở Việt Nam toán học cũng đã thể hiện được vị trí quan trọng kể từ khi đất nước bắt đầu phát triển. Tất cả mọi lĩnh vực đều cần có trình độ toán. Vì đó toàn học là một môn rất quan trọng mà chúng ta nên học và rèn luyện hàng ngày và đặc biệt là trẻ nhỏ.

Trong thế giới của công nghệ và giải trí, trò chơi điện tử không chỉ là một phần của cuộc sống hàng ngày mà còn là một phương tiện mạnh mẽ để thể hiện sự sáng tạo và kỹ năng lập trình. Trò chơi 2D, dù đơn giản về giao diện hình ảnh, nhưng lại mang trong mình sức hút và tiềm năng sáng tạo vô hạn.

Báo cáo này là kết quả của quá trình nghiên cứu và phát triển một trò chơi 2D, nơi em đã thử nghiệm và áp dụng các nguyên lý lập trình, kỹ thuật và nghệ thuật để tạo ra một trải nghiệm chơi game độc đáo và thú vị. Em đã đối mặt với nhiều thách thức trong quá trình này, từ việc xây dựng cấu trúc dữ liệu đến việc điều chỉnh hiệu suất và tối ưu hóa mã nguồn.

Báo cáo này không chỉ tập trung vào quá trình phát triển kỹ thuật mà còn đề cập đến quá trình thiết kế, ý tưởng sáng tạo và quản lý dự án. Em hy vọng rằng báo cáo này không chỉ là một tài liệu kỹ thuật mà còn là nguồn cảm hứng cho những ai đam mê lập trình game.

Cuối cùng, em xin gửi lời cảm ơn sâu sắc đến thầy cô cùng các bạn, người đã đóng góp và hỗ trợ một cách không ngừng nghỉ trong suốt quá trình này.

Mong rằng, báo cáo này sẽ mang lại kiến thức hữu ích và khám phá thú vị về thế giới của lập trình game 2D.

Trân trọng

Phạm Tiến Dũng

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU

Chào mừng bạn đến với thế giới của game giải đố, nơi mà sự sáng tạo và trí tuệ được thách thức mỗi ngày! Trong game giải đố, người chơi sẽ bước vào một thế giới phức tạp đầy bí ẩn và thách thức đầy kịch tính.

Trong cuộc hành trình này, người chơi sẽ phải tìm ra giải pháp cho các câu đố, các trò chơi trí tuệ, và các thử thách logic khác nhau. Từ việc giải mã mã điều khiển cổ xưa đến việc khám phá các phòng thí nghiệm khoa học kỳ bí, game giải đố đem lại cho người chơi những trải nghiệm thú vị và học hỏi không ngừng.

Với đồ họa tinh tế, âm nhạc hấp dẫn và cốt truyện hấp dẫn, game giải đố không chỉ là một trò chơi mà còn là một cuộc phiêu lưu đích thực. Hãy chuẩn bị tinh thần của bạn và bắt đầu hành trình khám phá những bí ẩn đang chờ đợi!

- Sản phẩm: Advanture Time
- Thể loại: Game Puzzle (giải đố)
- Lối chơi:
    - Game có nhiều màn chơi, trong mỗi màn chơi sẽ là một bản đồ với nhân vật nhất định (các nhân vật chỉ có thể di chuyển trên mặt đất)
    - Nhiệm vụ của người chơi là đi tìm những chiếc chìa khóa được bố trí khắp bản đồ, từ những chìa khóa tìm được người chơi sẽ mở được những cánh cửa bị khóa từ đó sẽ tìm được cái cầu thang giúp người chơi di chuyển đến màn tiếp theo
    - Để chiến thắng được trò chơi người chơi cần tìm đường để đi đến hòn đảo chứa rương kho báu

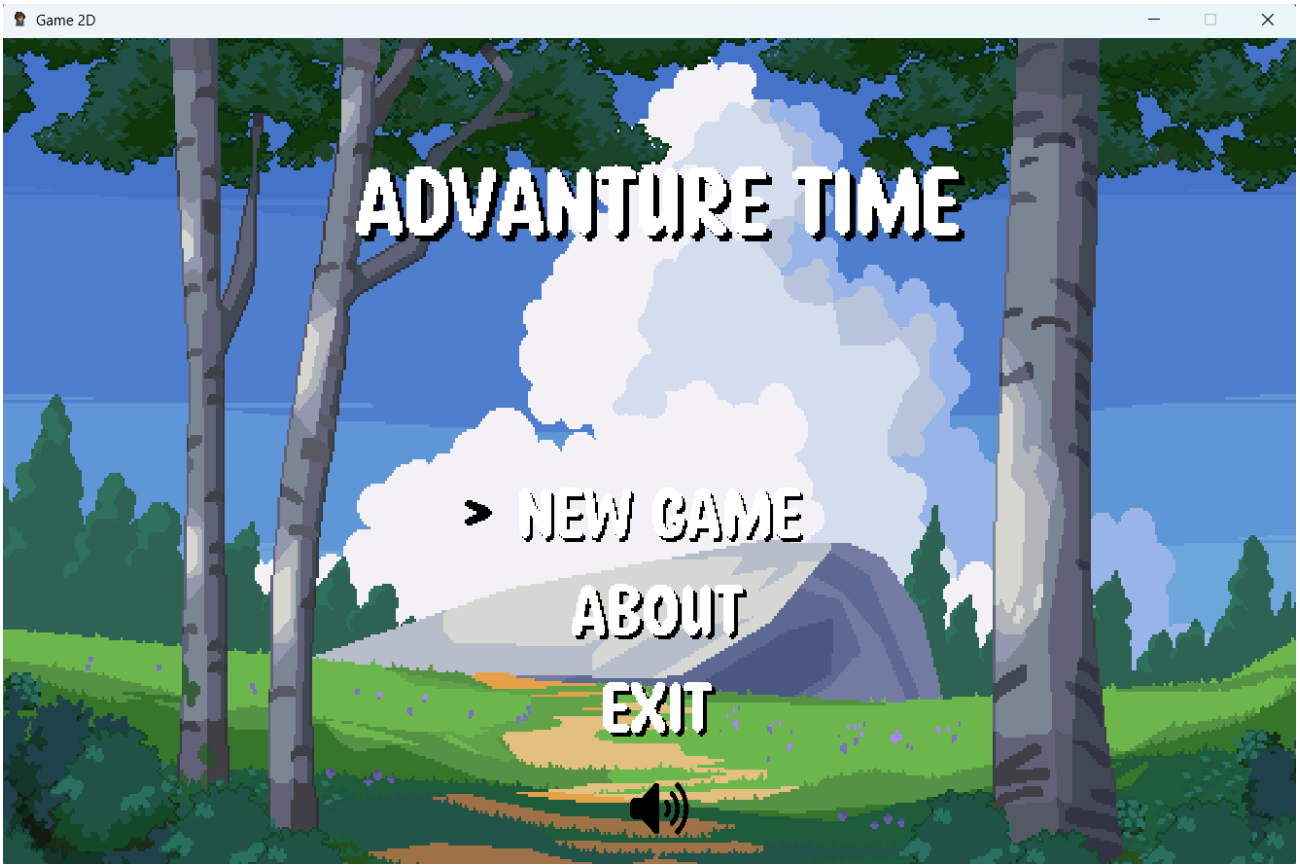
- Qua tìm hiểu ban đầu, nhận thấy game là một sản phẩm không quá khó
- Về thể loại, game puzzle rất dễ dàng tiếp cận với nhiều người, tập trung chủ yếu về tư duy giải đố, có khả năng kích thích sự hứng thú của người chơi.
- Khảo sát trải nghiệm người dùng:
    - Gameplay đơn giản, chỉ cần thao tác với một vài nút mũi tên trên bàn phím.
    - Màu sắc hợp lý (game với tông màu tươi sáng kết hợp với trầm tối).
    - Hình ảnh đơn giản và ưa nhìn
    - Âm thanh nhẹ nhàng, vui tươi.
    - Game đi từ dễ tới khó, chơi qua những màn khó thấy rất thú vị.

CHƯƠNG 2: CÁC TÍNH NĂNG CỦA GAME

2.1 Menu

2.1.1 Tính năng New Game trong Menu

Trong Menu chứa tính năng New Game cho phép người chơi tạo một game mới và bắt đầu trò chơi, nút Menu là một đoạn String được vẽ lên và ghi nhận sự kiện từ bàn phím để chuyển đổi giữa các GameState



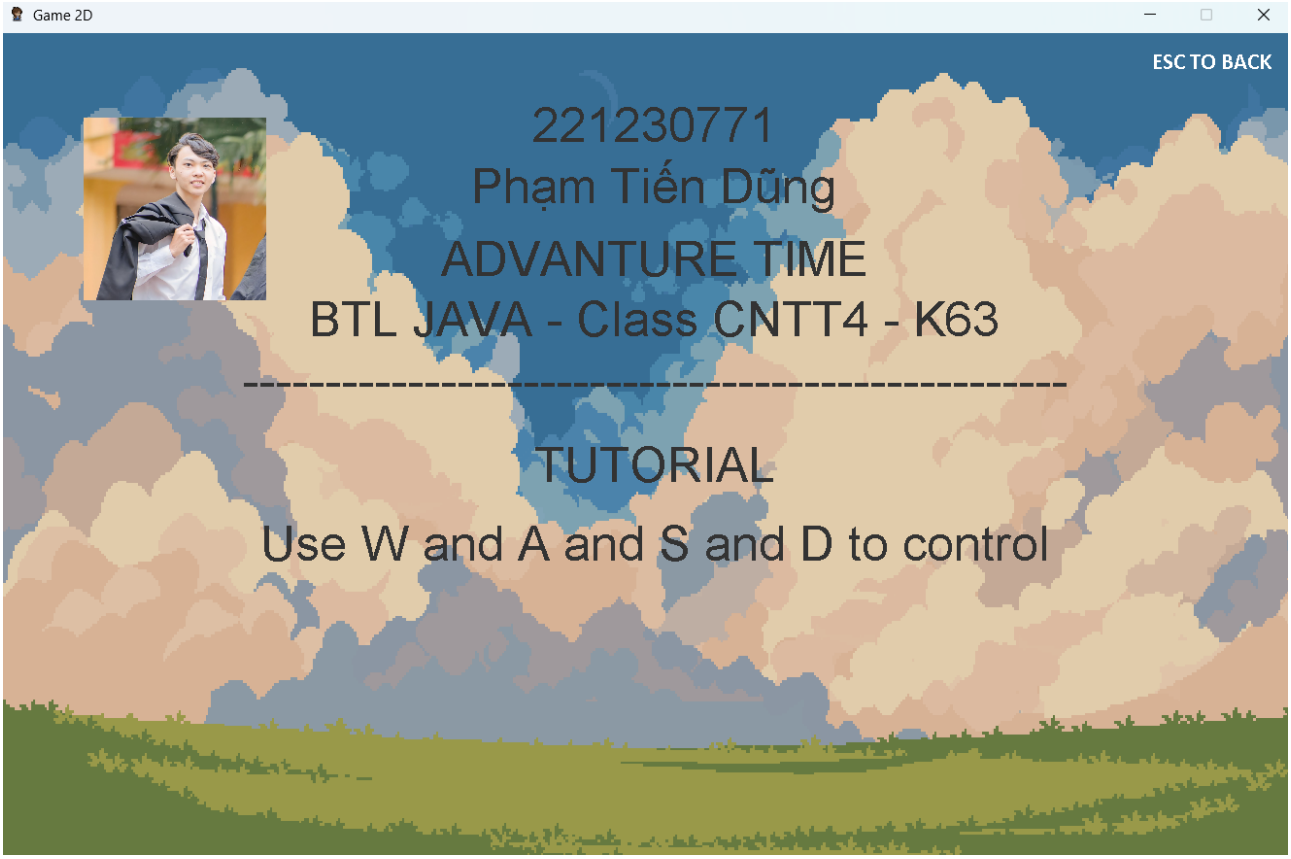
Hình 1: Menu của Game Advanture Time

Font chữ của New Game được sử dùng là một Font chữ custom do em tự thiết kế dựa theo hướng dẫn, Font chữ được thiết kế theo phong cách 2D

2.1.2 Tính năng About trong Menu

About chứa những thông tin về lập trình viên, số thứ tự nhóm, tên lớp học phần, tên lớp học, số báo danh và hướng dẫn chơi cơ bản của game vùng với một số thông tin quan trọng khác

Trong Menu này có một hình nền riêng, được đặt với Game State là AboutState, khi bấm ESC người dùng sẽ quay trở lại Menu của trò chơi



Hình 2: Menu About của Game Advanture Time

2.1.3 Tính năng EXIT (Thoát trò chơi)

Khi chọn EXIT người dùng sẽ thoát khỏi trò chơi, Menu này được em sử dụng lệnh System.exit để thực hiện buộc đóng trò chơi

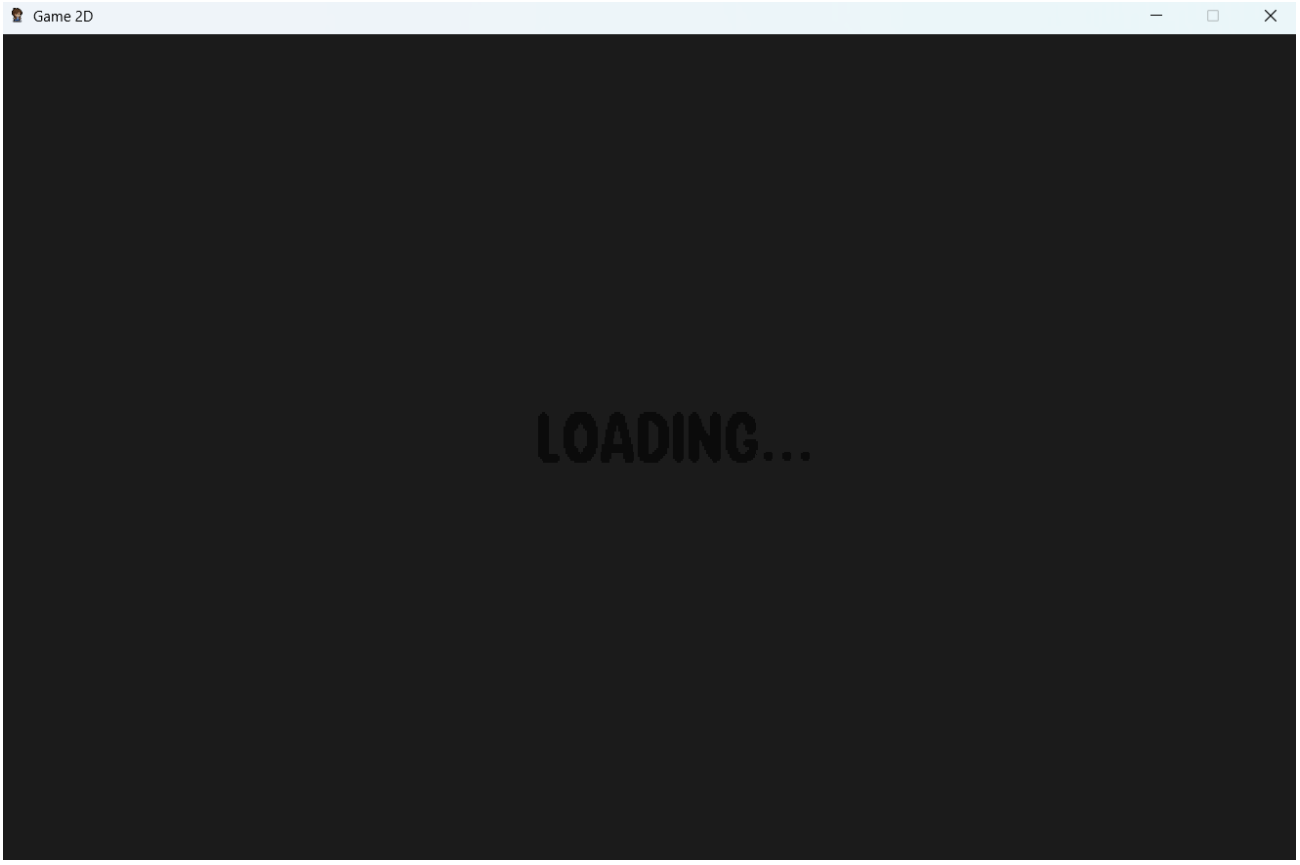
2.1.4 Tính năng bật tắt âm thanh

Khi muốn bật tắt âm thanh người dùng thực hiện đưa mũi tên trỏ vào hình chiếc loa ở màn hình Menu và nhân Enter để thực hiện tắt hoặc bật âm thanh, khi bật hoặc tắt âm thanh sẽ có hình ảnh tương ứng để người dùng dễ nhận biết (Hình 1)

2.2 Hiệu ứng chuyển cảnh (Scene Transition)

Hiệu ứng chuyển cảnh của game được em thiết kế là vẽ liên tục một màu được thiết lập từ trước rồi update() để giảm alpha của màu xuống qua từng lần update() từ đó tạo được hiệu ứng chuyển cảnh mượt mà

Khi chuyển động sẽ hiện chữ tương ứng dễ người dùng dễ dàng nhận biết cụ thể khi nhấn New Game sẽ hiện chữ Loading.. thể hiện đang load trò chơi, khi di chuyển vào Dungeon màn hình sẽ hiển thị Dungeon… Và tương tự như thế đối với các hoạt động khác của Player



Hình 3: Hiệu ứng chuyển canh của Game

Cách cài đặt của tính năng này như sau: đầu tiên ta thiết kế một Class SceneTransition với những thuộc tính và phương thức cần thiết, ta định nghĩa ra một biến Alpha là độ trong của màu rồi thực hiện giảm đần độ trong cho tới khi trong suốt thì dừng lại, từ đó sẽ tạo ra hiệu ứng chuyển cảnh

```
public void update() {
    if(isTransitioning && (System.nanoTime() - startTime) /
1000000 > transitionTime) {
        isTransitioning = false;
    }
}

public void draw(Graphics2D g2) {
    if(isTransitioning) {
        double elapsed = (System.nanoTime() - startTime) /
1000000;
        double alpha = 1 - elapsed / transitionTime;
        g2.setColor(new Color(color.getRed(), color.getGreen(),
color.getBlue(), (int) (255 * alpha)));
        g2.fillRect(0, 0, GameFrame.SC_WIDTH,
GameFrame.SC_HEIGHT);
        g2.setFont(pixelF2);
```

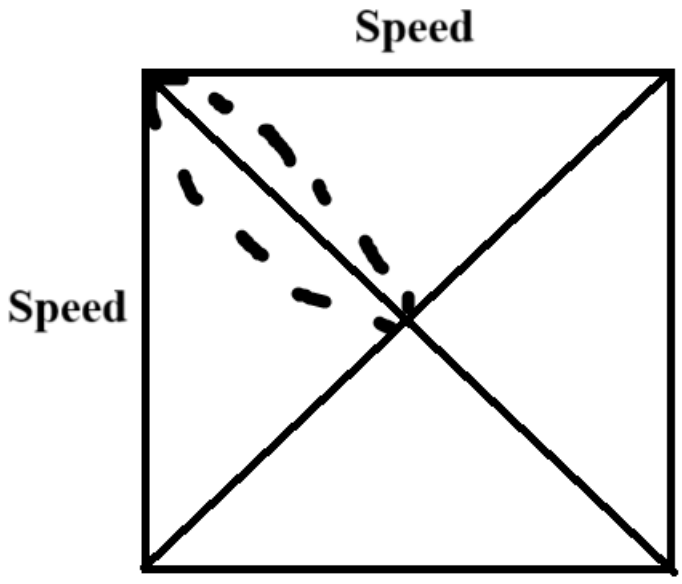
```
        }
}
```

2.3 Cơ chế di chuyển của nhân vật

2.3.1 Di chuyển cơ bản

Cơ chế di chuyển của nhân vật trong game được thực hiện bằng cách vẽ nhân vật dựa theo tọa độ của biến X,Y những biến này được lắng nghe sự kiện từ bàn phím để thay đổi phù hợp với sự di chuyển của nhân vật, trong game này em lựa chọn hệ trục tọa độ xy để tính toán, khi người chơi nhấn W thì nhân vật sẽ di chuyển lên trên tương đương với y-=speed và tương tự như thế với 3 hướng còn lại

Đối với việc di chuyển công đoạn khó nhất là tính toán để nhân vật di chuyển theo đường chéo và kiểm tra va chạm giữa nhân vật và các khối trên bản đồ. Để cho nhân vật di chuyển chéo em sẽ xét 4 hướng đó là Đông Bắc, Tây Bắc, Đông Nam, Tây Nam bằng việc lắng nghe sự kiện từ bàn phím, từ cộng trừ cho một nửa đường chéo của tam giác tạo bởi cạnh góc vuông speed



Hình 4: Minh họa cho cách tính đường chéo

2.4 Hiệu ứng di chuyển

2.4.1 Hoạt ảnh động khi đứng im

Khi nhân vật đứng im, nhân vật sẽ có hoạt ảnh thở việc này làm nhân vật nhấp nhô giúp game có đồ họa đẹp mắt

Hình 5: Hình ảnh nhân vật đứng im

Để thực hiện được việc này chúng ta cần có một mảng ảnh và vẽ chúng liên tục lên màn hình từ đó tạo nên ảnh động, ở đây em sử dụng 6 ảnh để vẽ nhân vật, cả 6 ảnh này cùng có kích thước là 48x48 để làm giảm dung lượng bộ nhớ, việc cài đặt cũng giống như cài đặt hiệu ứng chuyển cảnh, ở đây em tọa ra một Class Animation để quản lý Animation

```
public void setFrames(BufferedImage[] frames) {
    this.frames = frames;
    currentFrame = 0;
    startTime = System.nanoTime();
}
public void setDelay(long delay) {
    this.delay = delay;
}
public void setFrame(int frame) {
    currentFrame = frame;
}
public void update() {
    if (delay == -1) return;

    long elapsed = (System.nanoTime() - startTime) / 1000000;
    if (elapsed > delay) {
        currentFrame++;
        startTime = System.nanoTime();
    }
    if (currentFrame == frames.length) {
        currentFrame = 0;
    }
}
```

Thuật toán em sử dụng ở đây cũng giống như hiệu ứng chuyển cảnh đó là xây dựng hàm update() đùng để luôn phiên đổi ảnh từ 1 đến 6, hàm update animation sẽ được gọi đến trong game loop

2.4.2 Hoạt ảnh động khi di chuyển

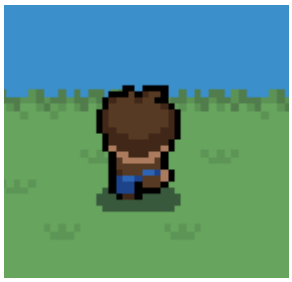
Khi nhân vật di chuyển sẽ đi theo 8 hướng khác nhau vì vậy em phải tạo ra 8 đối tượng Animation để quản lý hoạt động di chuyển của từng hướng nhưng đối mới di chuyển theo hướng Đông Bắc cũng giống đi sang phải vì vật em sử dụng chung Animation với đi qua phải vì vậy chúng ta chỉ cần 4 đối tượng Animation để thể hiện hoạt ảnh động

```
private Animation normal,left,right,up,down;
```

Tính cả hoạt ảnh đứng im ta sẽ có 6 đối tượng được đặt tên như trên, những đối tượng này sẽ được setImage trong hàm tạo của class Player khi Player được gọi đến trong GamePanel, khi nhân vật di chuyển chúng ta sẽ lắng nghe sự kiện di chuyển bằng interface KeyListener . Cách triển khai Code như sau:

```
normal.update();
if (inputManager.isRight==true ||
        inputManager.isLeft==true ||
        inputManager.isUp==true ||
        inputManager.isDown==true) {
     playerState =1;
     if (inputManager.isRight) {
         setDirection("right");
         right.update();
     }
     if (inputManager.isLeft) {
         setDirection("left");
         left.update();
     }
     if (inputManager.isUp) {
         setDirection("up");
         up.update();
     }
     if (inputManager.isDown) {
         setDirection("down");
         down.update();
     }
     if (inputManager.isRight && inputManager.isUp) {
        setDirection("rightUp");
    }
     if (inputManager.isRight && inputManager.isDown) {
        setDirection("rightDown");
    }
     if (inputManager.isLeft && inputManager.isUp) {
        setDirection("leftUp");
    }
     if (inputManager.isLeft && inputManager.isDown) {
        setDirection("leftDown");
     }
```

Ở đây chúng ta xét 4 phím và 4 tổ hợp phím tương ứng với các hướng di chuyển của nhân vật từ đó gọi đến hàm update() của animation phù hợp với hướng đang di chuyển từ đó tạo nên hiệu ứng chuyển động cho nhân vật



Hình 6: Nhân vật di chuyển lên trên

2.5 Cơ chế va chạm

2.5.1 Cơ chế va chạm khi gặp block trên map

Khi nhân vật va chạm với một khối trên map thì nhân vật sẽ không được đi chuyển theo cơ chế đó em đã viết ra một class CollisionChecker thực hiện nhiệm vụ trên, trong Class này chứa những biến để tính toán khoảng cách nhân vật và khung hình

```
int entityLeftWorldX = entity.getWolrdX() +
entity.getSolidArea().x;
int entityRightWorldX = entity.getWolrdX() +
entity.getSolidArea().x + entity.getSolidArea().width;
int entityTopWorldY = entity.getWolrdY() +
entity.getSolidArea().y;
int entityBottomWorldY = entity.getWolrdY() +
entity.getSolidArea().y + entity.getSolidArea().height;

int entityLeftCol = entityLeftWorldX / GameFrame.TILE_SIZE;
int entityRightCol = entityRightWorldX / GameFrame.TILE_SIZE;
int entityTopRow = entityTopWorldY / GameFrame.TILE_SIZE;
int entityBottomRow = entityBottomWorldY / GameFrame.TILE_SIZE;
```

Nhiệm vụ của các biến trên là tính toán các giới hạn của một đối tượng (entity) trong thế giới game dựa trên các giới hạn vật lý và kích thước của đối tượng.

entityLeftWorldX, entityRightWorldX, entityTopWorldY, entityBottomWorldY: Đây là các biến để xác định vị trí của đối tượng trong thế giới game. Chúng lấy vị trí của đối tượng (entity.getWolrdX(), entity.getWolrdY()) cộng với vị trí của vùng diện tích cố định (entity.getSolidArea().x, entity.getSolidArea().y) để xác định tọa độ thế giới của các cạnh của đối tượng.

entityLeftCol, entityRightCol, entityTopRow, entityBottomRow: Đây là các biến để xác định vị trí của đối tượng trong mạng lưới của thế giới game, thường được sử dụng để kiểm tra va

chạm với các đối tượng khác trong mạng lưới này. Cụ thể, chúng chia vị trí thế giới của các cạnh của đối tượng cho kích thước của mỗi ô trong mạng lưới (GameFrame.TILE_SIZE) để xác định hàng và cột tương ứng của ô mà đối tượng đang nằm trong đó.

Đầu tiên chúng ta sẽ vẽ một hình vuông có kích thước nhỏ hơn nhân vật, sau đó kiểm tra va chạm giữa hình vuông đó và các block trên World từ đó sẽ dừng chuyển động của nhân vật một cách hợp lý, khi nhân vật đi lên chúng ta sẽ xét góc trên bên trái và góc trên bên phải và tương tự với các hướng còn lại

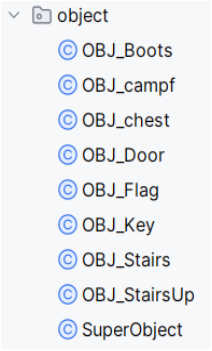
```
switch (entity.getDirection()){
    case "up":
        entityTopRow = (entityTopWorldY - entity.getSpeed()) /
GameFrame.TILE_SIZE;
        tilesNum1 =
gamePanel.getTileManager().getMapTileNum()[entityLeftCol][entityT
opRow];
        tilesNum2 =
gamePanel.getTileManager().getMapTileNum()[entityRightCol][entity
TopRow];

if(gamePanel.getTileManager().getTiles()[tilesNum1].isCollision()
==true ||
gamePanel.getTileManager().getTiles()[tilesNum2].isCollision()==t
rue){
            entity.setCollisionOn(true);
        }
        break;
```

2.5.2 Cơ chế va chạm khi gặp đối tượng

Cũng tương tự như va chạm với các khối trên bản đồ. Nhưng khi va chạm với đối tượng thì sẽ có đối tượng nhặt được và đối tượng không nhặt được vì thế ta phải giải quyết vấn đề này

Em tự định nghĩa ra một Package chứa những object có thể nhặt được trong thế giới, gồm các Class như sau:



Hình 7: Package object

SuperOject chứa toàn bộ những thuộc tính và phương thức chung của toàn bộ các Object trên, một vài Object có chứa Animation như Flag, Campfire thì Class đó sẽ được định nghĩa thêm thuộc tính Animation và phương thức vẽ Animation lên màn hình

Khi người chơi chạm vào object thì sẽ có sự kiện sảy ra ví dụ như chạm vào chìa khóa sẽ được thêm chìa khóa, chạm vào tăng tốc sẽ được cộng thêm tốc độ…..v.v..

```
switch (objName){
    case "key":
        gamePanel.playSE(2);
        hasKey++;
        gamePanel.getObj()[index]=null;
        gamePanel.getUi().showMessage("You got a key");
        break;
    case "door":
        if (hasKey>0){
            gamePanel.playSE(0);
            hasKey--;
            gamePanel.getObj()[index]=null;
            gamePanel.getUi().showMessage("You opened the door");
        }else {
            gamePanel.getUi().showMessage("You need a key");
        }
        break;
```

Đoạn code trên cho phép chương trình tính toán được việc người chơi có chạm vào chìa khóa không, nếu có thì chìa khóa được công thêm và object đó trên bản đồ sẽ biến mất, đối với cánh cửa thì nếu người chơi có chìa khóa khi chạm vào cửa thì cửa sẽ biến mất và trừ chìa khóa của người chơi và nếu người chơi không có chìa khóa thì sẽ không cho mở của và thông báo ra màn hình

2.6 Âm thanh trong Game

2.6.1 Nhạc nền

Game sử dụng nhạc nền vui tươi, và nó sẽ được lặp lại xuyên suốt quá trình chơi game, nhạc được chứa trong resource/sound, cách cài đặt như sau

```
public class Sound {
    private Clip clip;
    private boolean playing=true;
    private BufferedImage image;
    URL soundURL[]=new URL[5];
    private FloatControl volumeControl;
    private float volume = 1.0f; // Mức âm thanh mặc định
    public Sound(){
soundURL[0]=getClass().getResource("/resources/sounds/door/0.wav"
);
```

```
soundURL[1]=getClass().getResource("/resources/sounds/Background.
wav");
soundURL[2]=getClass().getResource("/resources/sounds/CollectKey.
wav");
soundURL[3]=getClass().getResource("/resources/sounds/MoveA.wav")
;
soundURL[4]=getClass().getResource("/resources/sounds/MoveDownSta
irs.wav");    }
```

Nhạc nền trong game 2D đóng vai trò quan trọng trong việc tạo ra một trải nghiệm chơi game sâu sắc và gây ấn tượng cho người chơi. Âm nhạc không chỉ làm tăng tính hấp dẫn của trò chơi mà còn có thể giúp tạo ra một không gian âm thanh phong phú và hòa mình vào thế giới của trò chơi.

Một bản nhạc nền thành công sẽ điều chỉnh cảm xúc và tạo ra bầu không khí phù hợp với từng phần của trò chơi. Ví dụ, nhạc nền có thể được điều chỉnh để tăng cường cảm giác hồi hộp và căng thẳng trong các tình huống hành động gay cấn hoặc tạo ra một bầu không khí bình yên và thư giãn trong các cảnh hướng dẫn hoặc khám phá.

Đồng thời, việc lựa chọn âm nhạc cũng cần phải phù hợp với chủ đề và đặc điểm của game. Ví dụ, một trò chơi phiêu lưu có thể chọn nhạc nền với giai điệu mở rộng và lôi cuốn, trong khi một trò chơi hành động có thể sử dụng nhạc nền nhanh nhẹn và mạnh mẽ để tạo cảm giác sức mạnh và quyết liệt.

Cuối cùng, việc tích hợp âm nhạc vào game cần phải được thực hiện một cách cân nhắc để không làm gián đoạn trải nghiệm chơi game của người chơi mà thay vào đó làm tăng thêm giá trị và sâu sắc cho trò chơi.

2.6.2 Hiệu ứng âm thanh (Sound Effect)

Hiệu ứng âm thanh trong game 2D là một phần quan trọng giúp tạo ra trải nghiệm chơi game sống động và hấp dẫn. Cùng với nhạc nền, hiệu ứng âm thanh đóng vai trò quan trọng trong việc tạo ra một không gian âm thanh đa dạng và chân thực, làm tăng tính tương tác và cảm giác thực tế cho người chơi.

Trong game sẽ được thêm vào những hiệu ứng âm thanh sau:

- Âm thanh nhặt chìa khóa
- Âm thanh mở cửa
- Âm thanh vào cầu thang
- Âm thanh lên cầu thang
- Âm thanh nhặt rương
- Âm thanh nhặt boot tốc độ

CHƯƠNG 3: XÂY DỰNG CÁC ĐỐI TƯỢNG

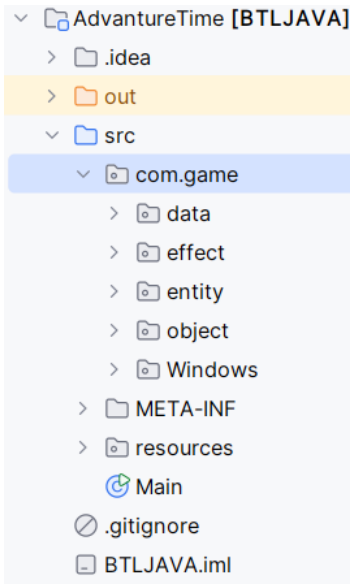
3.1 Giới thiệu về chương trình

Chương trình được thiết kế theo mô hình hướng đối tượng, mỗi đối tượng là một Class riêng và tuân thủ theo các tính chất của hướng đối tượng

- Tính kế thừa (Inheritance):
  - Kế thừa cho phép một đối tượng (class) có thể sử dụng lại các thuộc tính và phương thức của một đối tượng khác.
  - Đối tượng con (subclass) có thể kế thừa từ đối tượng cha (superclass), từ đó giúp tái sử dụng mã nguồn, giảm độ phức tạp của mã và tăng tính linh hoạt của chương trình.
- Tính đa hình (Polymorphism):
  - Đa hình cho phép các đối tượng cùng loại có thể được xử lý theo cách khác nhau dựa trên loại của chúng.
  - Đa hình cho phép một phương thức có thể được triệu gọi với nhiều hình thức khác nhau, tùy thuộc vào loại của đối tượng được triệu gọi.
- Tính đóng gói (Encapsulation):
  - Đóng gói đảm bảo rằng các dữ liệu (biến thành viên) và mã nguồn (phương thức) được bảo vệ bằng cách ẩn chúng khỏi sự truy cập trực tiếp từ bên ngoài đối tượng.
  - Đối tượng chỉ cung cấp một số cách để tương tác với dữ liệu và phương thức của nó thông qua các phương thức công cộng (public methods).
- Tính trừu tượng (Abstraction):
  - Trừu tượng cho phép che dấu các chi tiết phức tạp của một đối tượng và chỉ hiển thị các tính năng quan trọng mà người dùng cần phải biết.
  - Bằng cách này, trừu tượng giúp giảm sự phức tạp của chương trình và tăng tính tái sử dụng mã nguồn.

Hình 9: Mô hình lập trình hướng đối tượng

3.2 Cấu trúc chương trình



Hình 10: Cấu trúc chương trình

Lưu trữ hình ảnh và âm thanh được sử dụng trong trò chơi trong thư mục **resource**.

Class **Main**: Khởi chạy trò chơi.

3.2.1 Package data

Lưu trữ những dữ liệu map trong game, những dữ liệu này được lưu trữ bằng file txt, trong tương lai sẽ cải thiện để lưu vào cơ sở dữ liệu

Hiện tại Game có 3 màn chơi với 3 file txt để lưu trữ ma trận dữ liệu map

3.2.2 Package effect

- Class **Animation** quản lý hiệu ứng của các thành phần trong game
- Class **ImageLoader** xử lý việc load ảnh vào chương trình
- Class **ImageManager** quản lý hình ảnh
- Class **SceneTransition** quản lý hiệu ứng chuyển cảnh

3.2.3 Package Entity

- Class **Entity** định nghĩa các thuộc tính chung của vật thể trong Game
- Class **Player** quản lý nhân vật, các sự kiện liên quan đến nhân vật
- Class **Tile** định nghĩa các khối trong thế giới
- Class **TileManager** lưu trữ các khối và quy định va chạm

3.2.4 Package Object

- Class **OBJ_Boots** (extends **SuperObject**): định nghĩa vật phẩm tăng tốc độ
- Class **OBJ_campf** (extends **SuperObject**): định nghĩa lửa trại
- Class **OBJ_chest** (extends **SuperObject**): định nghĩa chiếc rương
- Class **OBJ_Door** (extends **SuperObject**) định nghĩa cái cửa
- Class **OBJ_Flag** (extends **SuperObject**) định nghĩa cái cờ
- Class **OBJ_Key** (extends **SuperObject**) định nghĩa cái chìa khóa
- Class **OBJ_Stairs** (extends **SuperObject**) định nghĩa cầu thang
- Class **OBJ_StairsUp** (extends **SuperObject**) định nghĩa cầu thang đi lên
- Class **SuperObject** định nghĩa những tính chất chung của vật thể

3.2.5 Package Windows

- Class **AssetSetter** đặt các vật phẩm lên map
- Class **CollisionChecker** kiểm tra va chạm
- Class **GameFrame** chứa các thông tin liên quan đến cửa sổ
- Class **GamePanel** lưu trữ những thành phần quan trọng của game
- Class **InputManager** xử lý sự kiện bàn phím
- Class **Sound** xử lý âm thanh
- Class **UI** quản lý menu
- Class **UtilityTool** tối ưu hóa việc đọc ảnh

3.3 Những Class và thuật toán quan trọng

- **GameFrame**: Chứa các thuộc tính của cửa sổ như width, height, tile size và chạy chương trình
- **GamePanel**: được implements Runnable và KeyListener
    - Class GamePanel thì quản lý và thực hiện việc update những đối tượng chính của game, chứa các thuộc tính quan trọng như CollisionChecker, InputManager, Sound, UI các thuộc tính này thực hiện những chức năng chính

của game như lấy sự kiện từ bàn phím, kiểm tra va chạm, vẽ giao diện và điều khiển âm thanh

- Hỗ trợ 1 số chức năng: kiểm tra xem có phải vừa mới ấn nút, vừa mới thả nút, hay đang nhấn giữ nút nào.

```
@Override
public void run() {
    int FPS=60;
    double drawInterval = 1e9 / FPS;
    double delta = 0;
    long lastTime = System.nanoTime();
    long currentTime;

    // calculate FPS
    long timer = 0;
    int drawCount = 0;

    while (isRunning) {
        currentTime = System.nanoTime();
        delta += (currentTime - lastTime) /
drawInterval;
        timer += (currentTime - lastTime);
        lastTime = currentTime;

        if (delta >= 1) {
            update();
            repaint();
            delta--;
            drawCount++;
        }
        // show FPS
        if (timer >= 1e9) {
            System.out.println("FPS: " + drawCount);
            timer = 0;
            drawCount = 0;
        }
    }
}
```

- Hàm run() của class GamePanel sẽ là gameloop, nó tính toán deltaTime theo FPS mình target, update(), repaint() mọi thứ theo FPS đó.
- Update() của Game sẽ là nơi đầu tiên được xử lý (nguồn), sau đó đi update tất cả mọi thứ nhỏ nhỏ hơn nó.

```
  public void update(){
      sceneTransition.update();
      closeSceneTransition.update();
```

```
        upstairsSceneTransition.update();
        downstairsSceneTransition.update();
        if (gameState == playState){
            player.update();
            flag.update();
            campf.update();
        }
        if (gameState == menuState){
            //menu
        }
    }
public void paintComponent(Graphics g){
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;
    if (sceneTransition.isTransitioning()){
        sceneTransition.draw(g2);
        g2.drawString("LOADING...",
GameFrame.SC_WIDTH/2-100, GameFrame.SC_HEIGHT/2-10);
    }else if (closeSceneTransition.isTransitioning()){
        closeSceneTransition.draw(g2);
        g2.drawString("COMING SOON",
GameFrame.SC_WIDTH/2-140, GameFrame.SC_HEIGHT/2-10);
    } else if
(downstairsSceneTransition.isTransitioning()) {
        downstairsSceneTransition.draw(g2);
        g2.drawString("DUNGEON...",
GameFrame.SC_WIDTH/2-110, GameFrame.SC_HEIGHT/2-10);
    } else if
(upstairsSceneTransition.isTransitioning()) {
        upstairsSceneTransition.draw(g2);
        g2.drawString("UPSTAIRS...",
GameFrame.SC_WIDTH/2-110, GameFrame.SC_HEIGHT/2-10);
    } else if (gameState == menuState){
        //draw menu
        ui.draw(g2);
    }else {
        // draw tile
        tileManager.draw(g2);
        // draw object
        for (int i=0;i<obj.length;i++){
            if (obj[i]!=null){
                obj[i].draw(g2,this);
            } }
        //draw player
        player.draw(g2);
        //draw UI
        ui.draw(g2);
        g2.dispose();
```

```
    }
}
```

- **InputManager**: Gồm các thuộc tính dùng để xử lý sự kiện bàn phím

```
public class InputManager {
    GamePanel gamePanel;
    public static boolean
isRight,isLeft,isUp,isDown,isSpace,isEnter;
```

  - isRight dùng để check xem Player có di chuyển sang phải hay không và tương tự những thuộc tính khác
  - Hàm processKeyPressed xử lý sự kiện nhấn từ bàn phím lấy vào một KeyCode và từ đó làm thay đổi tọa độ nhân vật, chuyển đổi giữa các state game và play Sound

```
public void processKeyPressed(int keyCode,GamePanel
gamePanel){
    this.gamePanel= gamePanel;
    if (gamePanel.gameState == gamePanel.menuState){
        switch (keyCode){
            case KeyEvent.VK_W:
                gamePanel.getUi().commandNum--;
                if (gamePanel.getUi().commandNum<0)
                    gamePanel.getUi().commandNum=3;
                break;
            case KeyEvent.VK_S:
                gamePanel.getUi().commandNum++;
                if (gamePanel.getUi().commandNum>3)
                    gamePanel.getUi().commandNum=0;
                break;
            case KeyEvent.VK_ENTER:
                if (gamePanel.getUi().commandNum==0){

gamePanel.getSceneTransition().start();
                    gamePanel.gameState =
gamePanel.playState;
                }
                if (gamePanel.getUi().commandNum==1){
                    gamePanel.gameState =
gamePanel.aboutState;
                }
                if (gamePanel.getUi().commandNum==2){
                    System.exit(0);
                }
                if (gamePanel.getUi().commandNum==3){
                    if
```

```
(!gamePanel.getMusic().isPlaying()) {

gamePanel.getMusic().playSound();} else if
(gamePanel.getMusic().isPlaying())
{gamePanel.stopMusic();}
                }break;}}
    if (gamePanel.gameState == gamePanel.aboutState){
        switch (keyCode){
            case KeyEvent.VK_ESCAPE:
                gamePanel.gameState =
gamePanel.menuState;
                break;} }
    if (gamePanel.gameState== gamePanel.playState){
        switch (keyCode){
            case KeyEvent.VK_W:
                isUp= true;
                break;
            case KeyEvent.VK_A:
                isLeft= true;break;
            case KeyEvent.VK_S:
                isDown=true; break;
            case KeyEvent.VK_D:
                isRight=true; break;
            case KeyEvent.VK_SPACE:
                isSpace=true; break;
            case KeyEvent.VK_ESCAPE:
                if (gamePanel.gameState ==
gamePanel.playState){ gamePanel.gameState =
gamePanel.menuState;} break;}}
```

- **Animation**: sẽ gồm 1 danh sách các frame, mỗi frame cho hiển thị trong khoảng thời gian nhỏ nhỏ (vd tầm 0.1s) , như vậy sau mỗi 0.1s nó sẽ chuyển sang vẽ frame tiếp theo, cho tới khi vẽ xong frame cuối cùng thì chuyển về vẽ frame đầu tiên.
- **Entity**: là Class viết ra để dùng chung cho hầu hết các Class trong Packages Entity ,Class này chứa những thuộc tính như sau

```
   public class Entity {
       private int wolrdX,wolrdY;
       protected Rectangle solidArea = new
   Rectangle(0,0,48,48);
       private int solidDefaultX,solidDefaultY;
       private boolean collisionOn = false;
       private String direction;
       private int speed;
       GamePanel gamePanel;
```

- Player: là Class quan trọng nhất trong game, nó chứa toàn bộ những tính năng có trong game, từ những sự kiện mà player thực hiện trong game mà chuyển hướng đến các Class khác để thực hiện trức năng tương ứng

```
public class Player extends Entity {
    private InputManager inputManager;
    private final int screenX=GameFrame.SC_WIDTH/2-
(GameFrame.TILE_SIZE/2);
    private final int screenY=GameFrame.SC_HEIGHT/2-
(GameFrame.TILE_SIZE/2);
    private int hasKey=0;
    private int playerState=0;
    private Animation normal,left,right,up,down;
```

- Load data form file:

```
public void loadMap(String filename){
    try {
        InputStream inputStream =
getClass().getResourceAsStream("/com/game/data/"+filena
me+".txt");
        BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(inputStream));
        int col=0;
        int row=0;
        while(col<gamePanel.getMaxWorldCol() &&
row<gamePanel.getMaxWorldRow()){
            String line = bufferedReader.readLine();
            while (col<gamePanel.getMaxWorldCol()){
                String numbers[] = line.split(" ");
                int num =
Integer.parseInt(numbers[col]);
                mapTileNum[col][row] = num;
                col++;

            }
            if (col==gamePanel.getMaxWorldCol()){
                col=0;
                row++;
            }
        }
        bufferedReader.close();
    }catch (Exception e){
        e.printStackTrace(); }}
```

- Data được lưu file text như sau:

- Chúng ta sử dụng hàm này để có thể load lại map khi đi vào dungeon hoặc đi lên cầu thang, bằng việc bắt sự kiện người chơi chạm vào cái cầu thang thì hàm này sẽ được gọi với tên file tương ứng
- Giá trị của các phần tử trong ma trận đã được thiết kế và định nghĩa ở phần mềm như sau:

Hình 11: Map editor

- SuperObject chứa toàn bộ những thuộc tính chung của OBJ gồm tên, ảnh, collision,…v.v.
- Các class còn lại quy định tên cho từng OBJ của riêng nó
- Đặc biệt Class OBJ_campf và OBJ_Flag là 2 đối tượng có chứa animation nên cần một số thuộc tính và phương thức đặc biệt như Animation, draw, update. Từ đó mà Animation mới có thể chạy được

- **ImageLoader**
  - Class này sẽ chứ những BufferedImage là biến static để dễ dàng gọi đến trong chương trình
  - Từng bức ảnh sẽ có tên tương ứng và khi gọi hàm để load file sẽ đưa vào tên của nó, ngoài ra với những đối tượng có animation thì việc load file ảnh sẽ phải đưa vào một mảng vì vậy, em đặt tên cho từng ảnh là index của chúng để dễ quản lý trong quá trình load file vào chương trình

```
public class ImageManager1 {
    private ImageLoader loader;
    private final String spritePath =
"/resources/sprites/";
    public static BufferedImage SPR_avatar;
    public static BufferedImage SPR_title_congra;
```

```
    public static BufferedImage SPR_sound_on;
    public static BufferedImage SPR_sound_off;
    public static BufferedImage[] SPR_background;
    public static BufferedImage[] playerUp, playerDown;
    public static BufferedImage[] playerLeft, playerRight;
    public static BufferedImage[] player;
    public static BufferedImage[] SPR_wall;
    public static BufferedImage[] SPR_fire_lamp;
    public static BufferedImage[] SPR_flag;
    public static BufferedImage SPR_stairs;
    public static BufferedImage SPR_stairsUp;
    public static BufferedImage SPR_key;
    public static BufferedImage SPR_key_door;
    public static BufferedImage SPR_bolt;
    public static BufferedImage[] SPR_cf;
    public static BufferedImage SPR_chest;
    public static BufferedImage SPR_black;
```

CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC, CẢI THIỆN VÀ NÂNG CẤP

4.1 Phương pháp – Ngôn ngữ - Công cụ

- Phương pháp:
  - Lập trình hướng đối tượng OOP
- Ngôn ngữ: Java
- Công cụ:
  - intelliJ IDEA: code
  - Graphics2D cùng các thư viện khác của java.awt
  - Adobe Ilustator: thiết kế một số asset cho game
  - Photoshop: chỉnh sửa 1 số ảnh

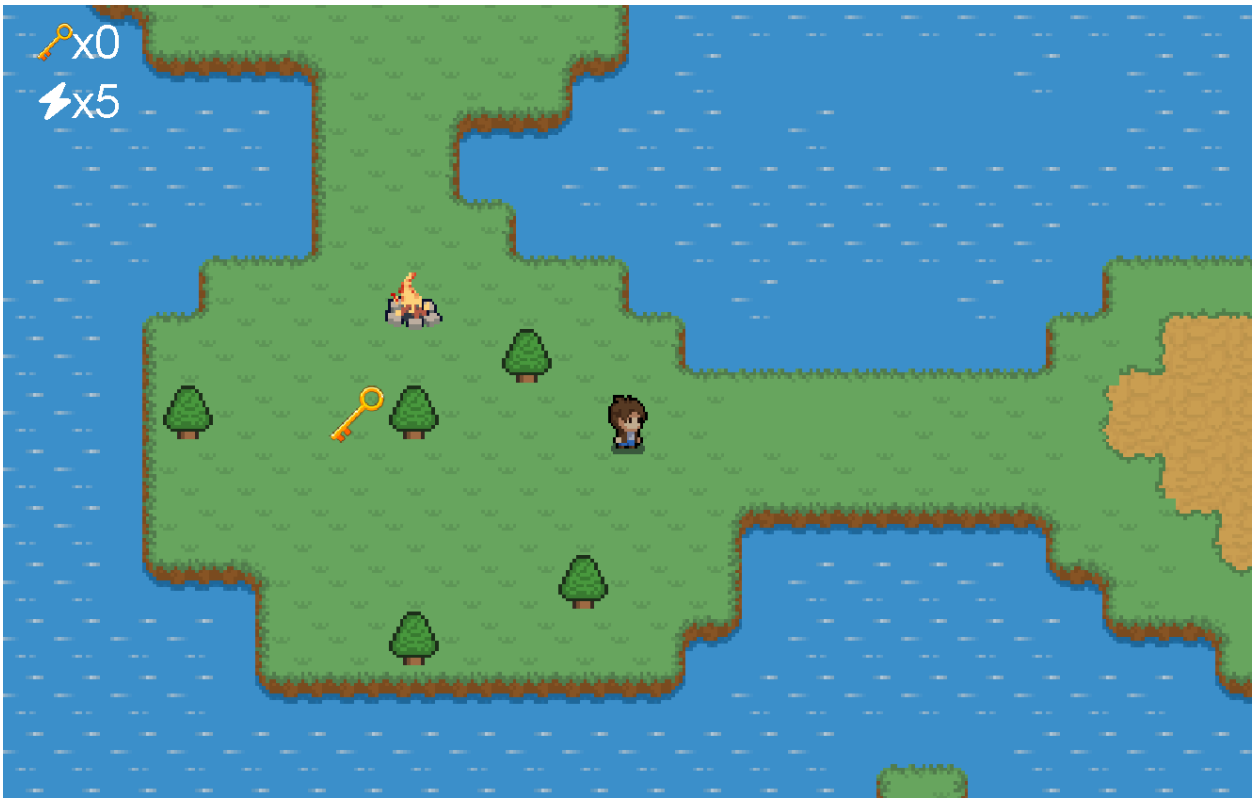
4.2 Giới thiệu hình ảnh và kết quả

- Menu:

- About:


ESC TO BACK
221230771
Phạm Tiến Dũng
ADVANTURE TIME
BTL JAVA - Class CNTT4 - K63
TUTORIAL
Use W and A and S and D to control

- Play:


x0
x5

4.3 Cải thiện và nâng cấp

- Trong tương lai trò chơi sẽ được thêm tính năng lưu game, cải thiện độ mượt mà, cải thiện game play, cải thiện đồ họa cùng nhiều màn chơi hơn
- Nâng cấp thêm cơ sở dữ liệu để có thể chơi online
- Nâng cấp tính năng mutiplayer
- Nâng cấp cơ chế giải đố

4.4 Những tài liệu tham khảo

- Series lập trình game 2d với java: https://youtu.be/om59cwR7psI
- Delta time handling: https://youtu.be/p7X64g6cOgQ
- Github tham khảo: https://github.com/NgTienHungg/SimpleMaze
- Github tham khảo: https://github.com/Dynamicslvl/Beat-em-up