Đây là phần giải thích về code của em :

1.

```
const int IMAGE_WIDTH = 49;
const int IMAGE_HEIGHT = 49;
const int TILE_SIZE = 49;

// Kích thước của Button
const int BUTTON_WIDTH = 200;
const int BUTTON_HEIGHT = 90;

// Kích thước frame
const int frameWidth = 120;
const int frameHeight = 120;

// Kích thước của background
const int BACKGROUND_WIDTH = 931 ;
const int BACKGROUND_HEIGHT = 539 ;

// Toạ độ nhân vật
int characterX = 441 ;
int characterY = 98 ;

// Kích thước mảng
const int ROW = 8;
const int COL = 19;

// Toạ độ nhân vật trên mảng
int x = 2 ;
int y = 10 ;

// Tài nguyên
int NUM_GOLD = 10;

int gold_found = 0 ;

// Trạng thái nhân vật
char status = 's' ;

int gold_map1[ROW+4][COL+2];

// Mảng chứa trạng thái của các ô trong MAP có vật cản hay ko
bool gold_map2[ROW+4][COL+2];
```

Ở đây em khai báo các thông số trong game sử dụng đến

Mảng gold_map1 ở đây là mảng biểu thị cho map của em

Các phần tử có giá trị bằng :

2 sẽ là chứa vàng ở sau lớp đất

1 sẽ là ô trống bị che bởi lớp đất

4 sẽ là ô trống bị che bởi lớp đá

8 sẽ là ô chứa vàng bị che bởi lớp đá

-1 sẽ là ô chứa dung nham nếu người chơi di chuyển vào sẽ chết

6 ở đây là phần em giới hạn bản đồ mà nhân vật di chuyển

5 là ô mặt đất mà nhân vật tự do di chuyển trên đó

Còn mảng gold_map2 là mảng biểu thị cho việc ô đó có bị chặn không để nhân vật có thể di chuyển trên đó

Còn biến status là biến biểu thị hướng của nhân vật đang đứng ban đầu em gán biến bằng kí tự s

‘s’ là hướng xuống

‘w’ là hướng lên

‘a’ là hướng sang trái

‘d’ là hướng sang phải

Biến frameWidth và frameHight biểu thị cho kích thước của mỗi frame trong ảnh chứa các chuyển động của nhân vật

Biến CharacterX , CharacterY biểu diễn vị trí của nhân vật trên bản đồ

Còn biến x , y biểu diễn vị trí nhân vật trên mảng

Biến NUM_GOLD là biến khởi tạo số vàng cần tìm

Biến gold_found biểu thị số vàng đã tìm thấy

2

```
// Khởi tạo tài nguyên
void generate_gold_map()
{
    ifstream file ("MAP\\Goldmap1.txt") ;
    int m=0 , n=0 ;
    while (file >> gold_map1[n][m]) {
        m++; // tăng chỉ số cột
        if (file.peek() == '\n') { // nếu gặp dấu xuống dòng, chuyển sang hàng mới
        n++; // tăng chỉ số hàng
        m = 0; // reset chỉ số cột về 0
        }
    }
    file.close();
    int count = 0 ;
    srand(time(NULL));

    while (count < NUM_GOLD) {
        int i = rand() % 8 + 3 ;
        int j = rand() % COL + 1;
        if (gold_map1[i][j] != 2 && gold_map1[i][j] != 4 && gold_map1[i][j] != -1 && gold_map1[i][j] != 8) {
            gold_map1[i][j] = 2;
            count ++ ;
        }
        if (gold_map1[i][j] == 4)
        {
            gold_map1[i][j] = 8 ;
            count ++ ;
        }
    }

    for ( int i = 0 ; i<ROW+4; i++)
    {
        for ( int j = 0 ; j<COL+2 ; j++)
        {
            if (gold_map1[i][j]==5 || gold_map1[i][j]==-1) {gold_map2[i][j]= true ;}
            else {gold_map2[i][j]=false ; }
        }
    }
}
```

Đây là hàm khởi tạo MAP , mảng gold_map1 sẽ lấy giá trị từ file text làm như này có thể tái sử dụng khi chơi lại và tiếp đó là hàm khởi tạo các ô chứa vàng hoàn toàn ngẫu nhiên , còn ở mảng gold_map2 thì em khởi tạo nếu tại ô đó chứa dung nham hoặc là trên mặt đất thì có thể di chuyển

3.

```
// Tạo cửa sổ SDL
SDL_Window* window = SDL_CreateWindow("MAD DIGGER", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, BACKGROUND_WIDTH, BACKGROUND_HEIGHT, SDL_WINDOW_SHOWN);
SDL_Renderer* renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);

// Hàm Load texture
SDL_Texture* tx (string s , SDL_Renderer * renderer)
{
    SDL_Surface* sf = IMG_Load(s.c_str()) ;
    SDL_Texture* x = SDL_CreateTextureFromSurface(renderer , sf) ;
    SDL_FreeSurface(sf) ;
    return x ;
}

// Load background texture
SDL_Texture * background = tx("Background\\background.png",renderer) ;

// Load GameOver texture
SDL_Texture * menu = tx("Background\\Menu.png",renderer) ;

// Load Win texture
SDL_Texture * Win = tx("End\\Win.png",renderer) ;

// Load background texture
SDL_Texture * Coin = tx("MAP\\Coin.png",renderer) ;

// Load GameOver texture
SDL_Texture * gameover = tx("End\\GameOver.png",renderer) ;

// Load Land texture
SDL_Texture * Land = tx("MAP\\Land.png",renderer) ;

// Load Lava texture
SDL_Texture * Lava = tx("MAP\\Lava.png",renderer) ;

// Load Move texture
SDL_Texture * UP = tx("Move\\UP.png",renderer) ;
SDL_Texture * DOWN = tx("Move\\DOWN.png",renderer) ;
SDL_Texture * LEFT = tx("Move\\LEFT.png",renderer) ;
SDL_Texture * RIGHT = tx("Move\\RIGHT.png",renderer) ;

// Load ROCK texture
```

Đây là phần tạo cửa sổ và tạo render còn ở dưới là hàm tạo texture từ ảnh và phần tải ảnh liên quan đến game của em

4.

```
// Khai báo biến đếm ngược
int countdown = 61;

void printcd()
{
    // Tạo phông
    TTF_Font* font = TTF_OpenFont("BebasNeue-Regular.ttf", 28);

    // Tạo văn bản
    SDL_Color color = { 255, 255, 255 , 255};
    SDL_Surface* surface = TTF_RenderText_Solid(font,to_string(countdown).c_str(), color);
    // Tạo texture
    SDL_Texture* cd = SDL_CreateTextureFromSurface(renderer, surface);
    SDL_Rect dstrect1 = { 441 , 30 , surface->w, surface->h };
    SDL_RenderCopy(renderer, cd, NULL, &dstrect1);
    TTF_CloseFont(font);
    SDL_FreeSurface(surface);
    SDL_DestroyTexture(cd);
}
```

Đây là hàm in giá trị của biến coutdown lên màn hình

5.

```
void printsc()
{
    // Tạo phông
    TTF_Font* font = TTF_OpenFont("BebasNeue-Regular.ttf", 28);

    // Tạo văn bản
    SDL_Color color = { 255, 255, 255 , 255};
    SDL_Surface* surface = TTF_RenderText_Solid(font, "Gold:", color);
    SDL_Surface* sc = TTF_RenderText_Solid(font, to_string(gold_found).c_str(), color);
    // Tạo texture
    SDL_Texture* gold = SDL_CreateTextureFromSurface(renderer, surface);
    SDL_Texture* Score = SDL_CreateTextureFromSurface(renderer, sc);
    SDL_Rect dstrect1 = { 800 , 30 , surface->w, surface->h };
    SDL_Rect dstrect2 = { 860 , 30 , 10, 34 };
    SDL_RenderCopy(renderer, gold, NULL, &dstrect1);
    SDL_RenderCopy(renderer, Score, NULL, &dstrect2);
    TTF_CloseFont(font);
    SDL_FreeSurface(surface);
    SDL_DestroyTexture(Score);
}
```

Hàm này dùng để in số vàng tìm được lên màn hình

6.

```
// Cập nhật tọa độ nhân vật và trạng thái
void print1 ( SDL_Texture * status )
{
    SDL_Rect statusRect ={characterX, characterY, 49, 49};
    SDL_Rect Rec1 = { 0 , 0 , frameWidth , frameHeight } ;
    SDL_RenderCopy(renderer, status, &Rec1, &statusRect);
}

// In ra tài nguyên
void print2 ()
{
    for (int i = 1; i < 20; i++) {
        for (int j = 3; j < 11; j++) {
            // Tính vị trí của ảnh nhỏ
            int xPos = i * IMAGE_WIDTH - 49;
            int yPos = j * IMAGE_HEIGHT;

            // Tạo đối tượng SDL_Rect để định vị trí và kích thước của ảnh nhỏ
            SDL_Rect destRect = { xPos, yPos, IMAGE_WIDTH, IMAGE_HEIGHT};
            if (xPos!=characterX || yPos!=characterY)
            {
                if (gold_map2[j][i]==false)
                {
                    // Render ảnh nhỏ lên background tại vị trí hiện tại
                    if (gold_map1[j][i]==1 || gold_map1[j][i]==2) SDL_RenderCopy(renderer, Land, NULL, &destRect);
                    if (gold_map1[j][i]==4 || gold_map1[j][i]==8) SDL_RenderCopy(renderer, ROCK, NULL, &destRect);
                    if (gold_map1[j][i]==3) SDL_RenderCopy(renderer, ROCK1, NULL, &destRect);
                    if (gold_map1[j][i]==7) SDL_RenderCopy(renderer, ROCK2, NULL, &destRect);
                }
                if (gold_map2[j][i]== true && gold_map1[j][i]==2) SDL_RenderCopy(renderer, Coin, NULL, &destRect);
            }
            if (gold_map1[j][i] == -1) SDL_RenderCopy(renderer, Lava, NULL, &destRect);
        }
    }
}
```

Hàm print1 là hàm vẽ nhân vật lên màn hình

Hàm print2 là hàm in ra MAP

7.

```
SDL_Texture * funcion ()
{
    switch (status)
    {
        case 's' :
        return DOWN ;
        case 'w' :
        return UP ;
        case 'a' :
        return LEFT ;
        case 'd' :
        return RIGHT ;
    }
    return DOWN ;
}

void Load_break(SDL_Texture * st)
{
    for ( int i = 0 ; i<7 ; i++)
    {
        SDL_RenderClear(renderer);
        // Vẽ frame hiện tại lên màn hình
        SDL_Rect frameRect = {i*120, 0, 120, 120};
        SDL_Rect Rect1 = {characterX, characterY , 49, 49};
        SDL_RenderCopy(renderer, background , NULL, NULL);
        printcd() ;
        printsc() ;
        print2() ;
        SDL_RenderCopy(renderer, st, &frameRect, &Rect1);
        // Hiển thị lên màn hình
        SDL_RenderPresent(renderer);
        // Tạm dừng 100ms trước khi vẽ frame tiếp theo
        if (i!=6) SDL_Delay(10);
    }
}
```

Hàm funcion() là hàm trả về ảnh phù hợp với trạng thái của nhân vật

Hàm Load_break() là hàm hiện thị chuyển động khi đào vàng của nhân vật

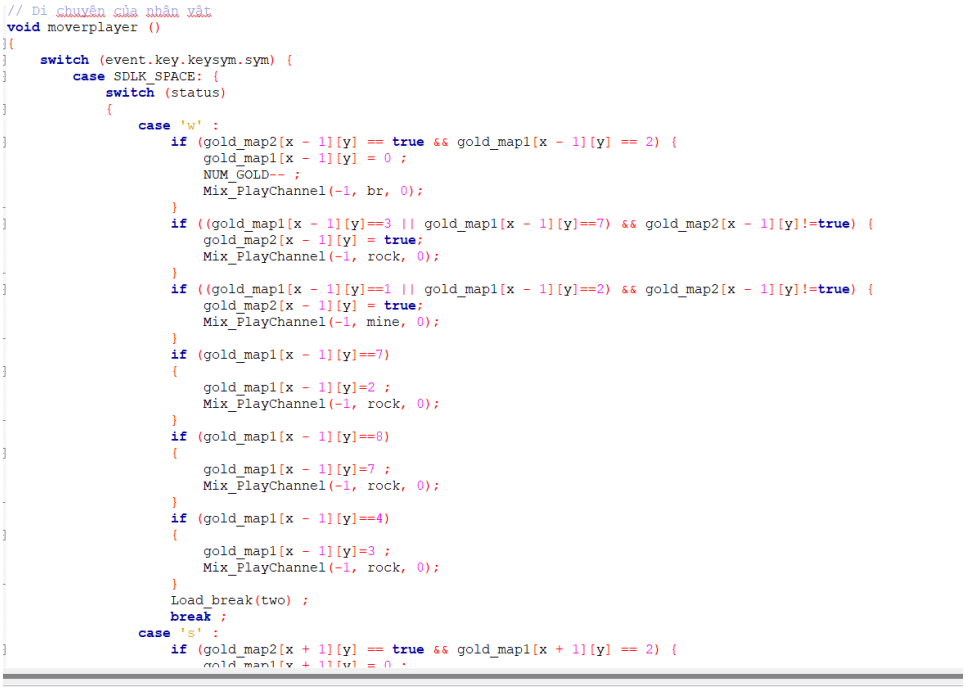
8.

```
void Update (SDL_Rect & Rect1)
{
    switch (status)
    {
        case 's' :
        characterY += 7 ;
        break ;
        case 'w' :
        characterY -= 7 ;
        break ;
        case 'a' :
        characterX-= 7 ;
        break ;
        case 'd' :
        characterX += 7 ;
        break ;
    }
}

void Load_mover(SDL_Texture * st)
{
    for ( int i = 0 ; i<7 ; i++)
    {
        SDL_RenderClear(renderer);
        // Vẽ frame hiện tại lên màn hình
        SDL_Rect frameRect = {i*120, 0, 120, 120};
        SDL_Rect Rect1 = {characterX, characterY , 49, 49};
        SDL_RenderCopy(renderer, background , NULL, NULL);
        printcd() ;
        printsc() ;
        print2() ;
        SDL_RenderCopy(renderer, st, &frameRect, &Rect1);
        Update(Rect1) ;
        // Hiển thị lên màn hình
        SDL_RenderPresent(renderer);
        // Tạm dừng 100ms trước khi vẽ frame tiếp theo
        if (i!=6) SDL_Delay(10);
    }

}
```

Hàm Update() là hàm cập nhật tọa độ của nhân vật trong khi di chuyển vì 1 ảnh có 7 frame mà mỗi lần di chuyển tăng tọa độ lên 49 thì mỗi khi load 1 frame thì tọa độ tăng lên 7.

Hàm Load_mover() là hàm load chuyển động của nhân vật

9.



```
// Di chuyển của nhân vật
void moverplayer ()
{
    switch (event.key.keysym.sym) {
        case SDLK_SPACE: {
            switch (status)
            {
                case 'w' :
                    if (gold_map2[x - 1][y] == true && gold_map1[x - 1][y] == 2) {
                        gold_map1[x - 1][y] = 0 ;
                        NUM_GOLD-- ;
                        Mix_PlayChannel(-1, br, 0);
                    }
                    if ((gold_map1[x - 1][y]==3 || gold_map1[x - 1][y]==7) && gold_map2[x - 1][y]!=true) {
                        gold_map2[x - 1][y] = true;
                        Mix_PlayChannel(-1, rock, 0);
                    }
                    if ((gold_map1[x - 1][y]==1 || gold_map1[x - 1][y]==2) && gold_map2[x - 1][y]!=true) {
                        gold_map2[x - 1][y] = true;
                        Mix_PlayChannel(-1, mine, 0);
                    }
                    if (gold_map1[x - 1][y]==7)
                    {
                        gold_map1[x - 1][y]=2 ;
                        Mix_PlayChannel(-1, rock, 0);
                    }
                    if (gold_map1[x - 1][y]==8)
                    {
                        gold_map1[x - 1][y]=7 ;
                        Mix_PlayChannel(-1, rock, 0);
                    }
                    if (gold_map1[x - 1][y]==4)
                    {
                        gold_map1[x - 1][y]=3 ;
                        Mix_PlayChannel(-1, rock, 0);
                    }
                    Load_break(two) ;
                    break ;
                case 's' :
                    if (gold_map2[x + 1][y] == true && gold_map1[x + 1][y] == 2) {
                        gold_map1[x + 1][y] = 0 ;
```

Đây là hàm xử lí sự kiện nhập từ bàn phím nếu nhấn Space thì sẽ kiểm tra hướng của nhân vật và ô ở trước mặt nhân vật để xử lí việc đào đất đá và in ra chuyển động đào

```
        case SDLK_LEFT: {
            status = 'a' ;
            if (gold_map2[x][y-1] == true)
            {
                y-- ;
                Load_mover(LEFT) ;
            }
            printcd() ;
            print2() ;
            print1(funcion()) ;
            break;
        }
        case SDLK_RIGHT:{
            status = 'd' ;
            if (gold_map2[x][y+1] == true)
            {
                y++ ;
                Load_mover(RIGHT) ;
            }
            printcd() ;
            print2() ;
            print1(funcion()) ;
            break;
        }
        case SDLK_UP : {
            status = 'w' ;
            if (gold_map2[x - 1][y] == true)
            {
                x-- ;
                Load_mover(UP) ;
            }
            printcd() ;
            print2() ;
            print1(funcion()) ;
            break;
        }
```

Đây là phần xử lí khi nhập sự kiện từ bàn phím là các phím mũi tên để di chuyển

10.

```
// Khai báo biến thời gian
Uint32 last_time = SDL_GetTicks();

void Gameplay ()
{
    SDL_RenderClear(renderer) ;
    bool quit = false;
    while (!quit) {
        SDL_RenderCopy(renderer, background, NULL, NULL);
        printcd() ;
        printsc() ;
        print2() ;
        print1(funcion()) ;
        // Cập nhật renderer
        SDL_RenderPresent(renderer);
        // Tính thời gian kể từ lần cuối cập nhật biến thời gian
        Uint32 current_time = SDL_GetTicks();
        Uint32 elapsed_time = current_time - last_time;

        // Nếu đã đủ thời gian, cập nhật biến đếm ngược và cập nhật biến thời gian
        if (elapsed_time >= 1000) {
            countdown--;
            last_time = current_time;
        }

        // Kiểm tra nếu đếm ngược bằng 0 thì dừng chương trình
        if (countdown == 0) {
            SDL_RenderCopy(renderer, background, NULL, NULL);
            printcd() ;
            printsc() ;
            print2() ;
            print1(funcion()) ;
            SDL_RenderCopy(renderer, gameover, NULL, NULL);
            SDL_RenderPresent(renderer);
            Mix_PlayChannel(-1, Over, 0);
            SDL_Delay(2000);
            return ;
        }
```

Đây là hàm chơi game và xử lí biến countdown ;

```
 if (elapsed_time < 1000) {
     // Xử lý sự kiện
     while(SDL_PollEvent(&event)) {
         if (event.type == SDL_QUIT) {
             Destroy();
         } else if (event.type == SDL_KEYDOWN) {
             SDL_RenderCopy(renderer, background, NULL, NULL);
             printsc() ;
             printcd() ;
             // Di chuyển
             moverplayer();
             SDL_RenderPresent(renderer);
             if (gold_map1[x][y]==2)
             {
                 gold_found++ ;
                 gold_map1[x][y]=1 ;
                 SDL_RenderClear(renderer) ;
                 SDL_RenderCopy(renderer, background, NULL, NULL);
                 printcd() ;
                 printsc() ;
                 print2() ;
                 print1(funcion()) ;
                 SDL_RenderPresent(renderer);
                 Mix_PlayChannel(-1, Gain, 0);
             }
             if (gold_map1[x][y]==-1)
             {
                 SDL_Rect statusRect ={characterX, characterY, 49, 49};
                 SDL_RenderCopy(renderer, Fire, NULL, &statusRect);
                 SDL_RenderCopy(renderer, gameover, NULL, NULL);
                 printcd() ;
                 // Cập nhật renderer
                 SDL_RenderPresent(renderer);
                 Mix_PlayChannel(-1, Over, 0);
                 SDL_Delay(2000);
                 return ;
             }
```

11 .

```
SDL_Texture * ins = Ins1 ;

void Instruction()
{
    SDL_RenderCopy(renderer,ins,NULL,NULL) ;
    SDL_RenderPresent(renderer) ;
    SDL_Event event1 ;
    if (ins == Ins1)
    {
        while (true)
        {
            while (SDL_PollEvent(&event1))
            {
                if (event1.type == SDL_KEYDOWN)
                {
                    switch (event1.key.keysym.sym)
                    {
                        case SDLK_RIGHT :
                        ins = Ins2 ;
                        SDL_RenderCopy(renderer,ins,NULL,NULL) ;
                        SDL_RenderPresent(renderer) ;
                        Instruction() ;
                        return ;
                        case SDLK_LEFT :
                        return ;
                    }
                }
            }
        }
    }
    else if (ins == Ins2)
    {
        while (true)
        {
            while (SDL_PollEvent(&event1))
```

Hàm này để xử lí phần instruction có thể đổi ảnh instruction khi nhấn phím mũi tên

12.

```
void Reset()
{
    // Toa độ nhân vật
    characterX = 441 ;

    characterY = 98 ;

    // Toa độ nhân vật trên mảng
    x = 2 ;
    y = 10 ;

    // Tài nguyên
    NUM_GOLD = 10;

    gold_found = 0 ;

    // Trang thái nhân vật
    status = 's' ;

    countdown = 61;

}
```

Hàm reset dùng để khởi tạo lại tài nguyên khi chơi lại

13.

```
void Menu()
{
    generate_gold_map();
    SDL_RenderCopy(renderer, menu, NULL, NULL);
    SDL_RenderPresent(renderer) ;
    SDL_Event event1 ;
    bool quit = false ;
    while (!quit) {
    while (SDL_PollEvent(&event1)) {
        // lấy các sự kiện mới nhất từ hàng đợi sự kiện
        switch (event1.type) {
            case SDL_QUIT:
                Destroy() ;
                return ;
            case SDL_MOUSEBUTTONDOWN:
                int a, b;
                SDL_GetMouseState(&a, &b);
                if (a >= 355 && a < 355 + BUTTON_WIDTH && b >= 172 && b < 172 + BUTTON_HEIGHT) {
                    // Chọn Play
                    Gameplay() ;
                    Reset() ;
                    Menu() ;
                    Destroy() ;
                    return ;
                }
                else if (a >= 355 && a < 355 + BUTTON_WIDTH && b >= 302 && b < 302 + BUTTON_HEIGHT) {
                    Instruction() ;
                    Reset() ;
                    Menu() ;
                    Destroy() ;
                    return ;
                }
                else if (a >= 355 && a < 355 + BUTTON_WIDTH && b >= 432 &&  b < 432 + BUTTON_HEIGHT) {
                    Destroy() ;
                    return ;
                }
            break;
            }
        }
    }
}
```

Hàm xử lí lựa chọn ở Menu

Em mới học nên code của em chưa được gọn hay tối ưu mong các thầy thông cảm em sẽ cố gắng cải thiện . Em xin cảm ơn ạ !