Implementation of Graph in Strategy Video Game

Yodi Pramudito (13511095)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511095@std.stei.itb.ac.id

***Abstract*—This paper contains the impelementation of discrete structure course in strategy video game, specifically in Fire Emblem : Heroes of Light and Shadow. There are many mechanism of the game that can be explained using graph and tree theorem. This paper will explain the usage of graph as path finding method and tree in determining the enemy behavior in the mechanism of Fire Emblem.**

***Index* — Dijkstra's Algorithm, Fire Emblem, Graph, Strategy Game, Tree.**

I. Introduction

Strategy game is one of the oldest type of game known. One example of strategy game that already known well by most people around the world is chess. Strategy game is quite favorable for some people beacuse they have to think every move that they about to make in order to win the game. Strategy game is keep impoving as the time goes by, and now in the era of video game, strategy game is not only a simple board game that have a simple rule, digital entertainment era has bring strategy game into a whole new level of complexity.

In this paper the writer will explain some of mechanism in one of the famous strategy game called Fire Emblem. There are many mechanism in the game that can be explained using Graph and Tree therefore the writer choose this game as the object for this paper. In this paper the writer will explain the analysis of enemy behavior using decision tree and path finding mechanism using graph and Djikstra's algorithm.

II. Basic Theory

A. Graph

Graph is a model to representate discrete objects and their relations. Graph consist of *vertices* (*nodes*) and *edges* that connect those vertices. Each edge has either one or two vertices associated with it, called its *endpoints*. A graph G can be notated as below :

$$G = (V, E)$$

The graph G above is consist of V, a nonempty set of vertice and E, a set of edge.

The graph explained above is called *undirected graph*. Besides undirected graph there is also a graph that its edge has a direction assigned to it, it's called *directed graph*.

A directed graph (V,E) consist of V, a nonempty set of vertices and E, a set of directed edge. Each directed edge is associated with the ordered pair of vertices (u,v) which means it's start at *u* and end at *v*.

There are several classification of graph as shown in table below.

Type	*Edges*	*Multiple Edges Allowed?*	*Loops Allowed?*
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed grap	Both	Yes	Yes

There is some terms that used in graph. *Multiple Edge* is a condition where two vetrices associated with more than one edge. *Loop* is a term for an edge that only associated with one node. *Bridge* or *cut edge* is an edge in a connected graph which if the edge is removed then the graph is become unconnected.

There is also a special kind of graph which is called by *weigthed graph*. Weigthed graph is a graph which all of its edges has a value.

B. Dijkstra's Algorithm

Dijkstra's Algorithm is an algorithm that used to find the shortest path *a* to *z* in a weigthed graph where each value on the edge is represent the distance between two associated vertices. The Dijkstra's Algorithm is shown as below:

Dijkstra's Algorithm

procedure *Dijkstra* (*G*: weighted connected simple graph, with all weights positive)
{*G* has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$ where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in *G*}

for $i := 1$ **to** n
    $L(v_i) := \infty$
$L(a) := 0$
$S := \varnothing$
{the labels are now initialized so that the label of *a* is 0 and all other labels are ∞, and *S* is the empty set}
while $z \notin S$
    u := a vertex not in *S* with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in *S*
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
        {this adds a vertex to *S* with minimum label and updates the
        labels of vertices not in *S*}
return $L(z)$ {$L(z)$ = length of a shortest path from *a* to *z*}

C. Tree

Tree is a connected undirected graph which doesn't contain any circuit. Tree has a several properties, asume that G = (*V*,*E*) and G is a connected graph with *n* vertices then all of these following argument is equivalent :

- G is a tree
- Every pair of vertices in G is connected with a single path
- G is connected and have m = n-1 number of edges
- G doesn't contain any circuit and have m = n-1 number of edges
- G doesn't contain any circuit and an addition of one edge will create exactly one circuit
- Every edge on G is a bridge
- G is connected and all of its edge is a bridge

There are type of tree that being used in decision making, this tree is called decision tree. Decision tree consist of many vetrices that act as an conditional statement and the answers for the statement will determine the branch that it will access next. The action/result is given in the end (*leaf*) of the tree.

III. FIRE EMBLEM

Fire Emblem : Heroes of Light and Shadow is a game that developed by *Intelligent System* under Nintendo's console NDS. This game is released in July 2010.

Picture - 3.1 : main menu screen

A. Gameplay

Fire emblem is a turn based strategy game that have many stages. Each of the stages has its own objective to be cleared, in some of the stage we must defend against enemy attack, escape from the battle and sometime we must invade enemy's castle. Despite the variety of objective in each stages, in order to clear the stage we must lead the main character, the lord, into the yellow panel which usually located in enemy's castle.

The challenge of the game is to keep all allied units survive until the stage is cleared. Unlike many other strategy game where we still can use the units that have been lost in battle in the next stage, if an unit lost a battle in Fire Emblem then we will lost that unit permanently until the end of the game.

B. Panels

Fire emblem is a strategy game that is based on panel system. The strategy for winning fire emblem game is depend on the condition of surrounding panels.

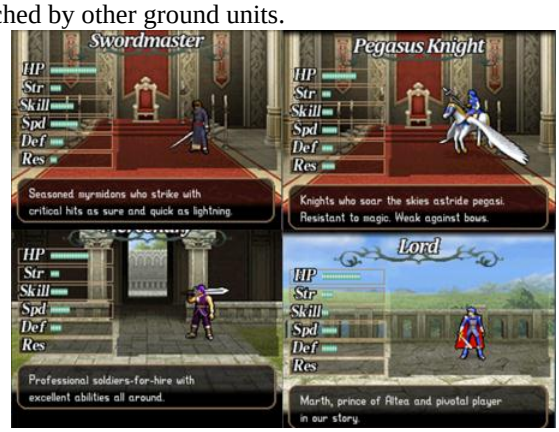
There are many kind of panel in fire emblem. Each panel represent the environment in that position. Each

kind of panel can give different kind of advantages and disadvantages. For example is forrest that can provide bonus evasion for the unit that standing above it but can decrease the unit movability for the unit that pass it.

Picture - 3.2 : types of panel

C. Unit type

There are many type of unit in Fire Emblem. Each kind of unit has different move range and movability on different kind of terrain. For example, most of ground unit's movement will be decreased while moving on the sand except for unit that use magic such as *mage* and *curate*. There also flying type unit which have a movability that doesn't affected by the kind of terrain it move on and can move to some panels that can't be reached by other ground units.



Picture - 3.3 : types of unit

D. Enemy

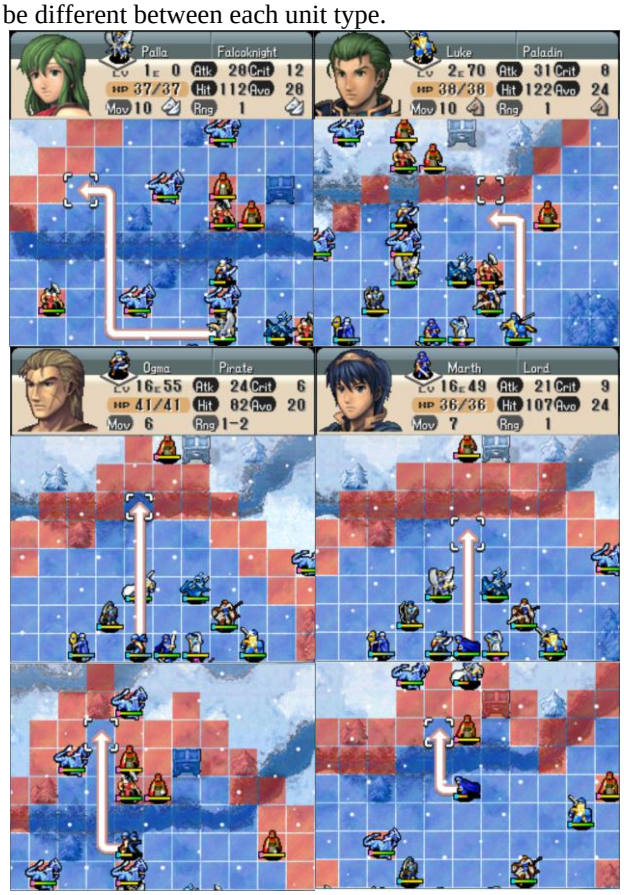
The main obstacles in clearing a stage in fire emblem is the enemy. Just like our own forces, the enemy forces is also consist of different kind of units. Besides the variety of unit type, each of enemy unit also has its own behavior. There are several kinds of enemy behavior in Fire Emblem, some of them are aggressive and some are defensive. There is also an enemy that always remain stationary. Some of them even have special kind of behavior.

IV. Analysis

A. Path finding in Fire Emblem

On the start of every unit's turn in fire emblem, that unit can decide whether it want to move to other panel or not. There are some algorithm that can explain the mechanism of the game to determine whether the unit able to reach a panel or not. In most strategy game the algorithm to determine the possible target panel can be done using BFS (Breath First Search), but in Fire Emblem there are some factor that can't be explained using BFS.

In Fire Emblem the unit movement characteristic can be different between each unit type.



Picture - 4.1 : unit type movability

The panel that highlighted in blue is the panel that can be selected as target panel for movement. The red highlighted panel is the panel that can be attacked, which doesn't related to the topic that being analyzed.

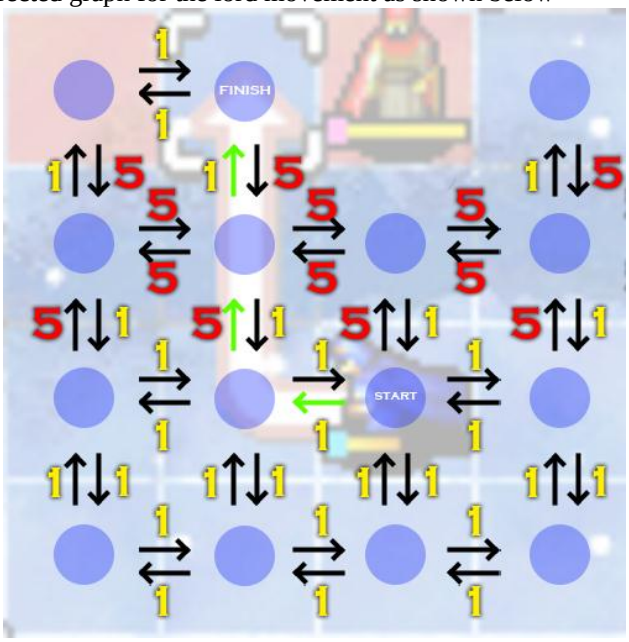
As seen in the Picture 4.1 above, starting from top left then clockwise, the movement of every unit is different. The pegasus knight (top left unit) has maximum movement range 10 and it can move freely arround the

field despite the terains it passed (arrow path). On the contrary, the paladin (top right unit) has maximum movement range 10, only walk for 5 panel and can't proceed to go north even further. This is because the paladin doesn't have the ability to enter river panel (as shown in picture 3.2). But there is some special unit that have the ability to pass river such as the lord (bottom right unit) which has maximum movement range 7. On the first picture the lord seems like it can't pass the river, but on the second picture the lord can pass the river but its maximum path (arrow path) is only 3 panel far. In order to count the movement cost to enter the river we need to use the formula below:

(*max movement* - *max path* + 1 = *movability cost*)

This means that in order to enter the river the lord must use 7-3+1 = 5 movability to enter the river panel. On the other hand, the pirate (bottom left unit) only need 6-5+1 = 2 movability to enter the river panel.

Knowing the fact above we can create a weigthed directed graph for the lord movement as shown below

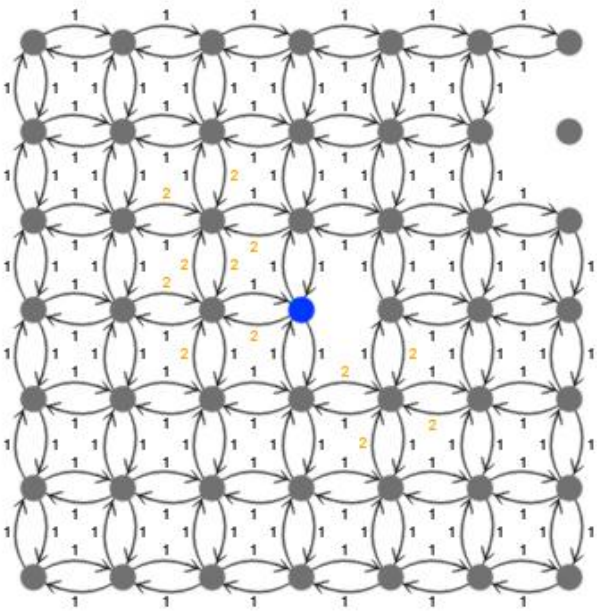


Picture - 4.2 : graph for *the lord* movement

In the graph above the blue dots represent a vetrices. The arrow represent the edge that associated with two vetrices. And the number shown is the value of each edge. The path taken by the lord is represented in green arrows. So the total of movement cost that required by the lord to move from *start* vetrices to the *finish* vetrices is 1+5+1 = 7, equal to the maximum movement posible for the lord.

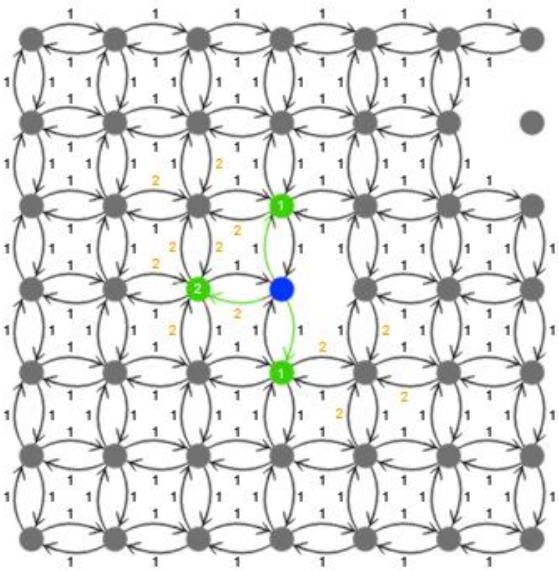
The graph used in every kind of unit is different. Based on the picture 4.2 the graph for the pirate unit will look the same but the edge that have 5 value will be having 2 value instead. And for the pegasus knight all the value will be 1. While for the paladin all the edge that associate with the river panel will be erased.

In order to know all possible panel that can be moved to, it is required to build a graph like aboves. To create a graph like in picture 4.2 we just simply represent all the panel as vertices an give all incoming edge a value as much as the cost needed for the unit to enter the panel. If entering the panel is imposible then there is no need in making the edges associated with the vetrices. By using a slightly modified Dijkstra's algorithm we can find all the possible destination panel. Asume the unit's maximum movement range is 3 and we already have a graph as shown below



Picture - 4.3 : phase 0 of modified Dijkstra's algorithm

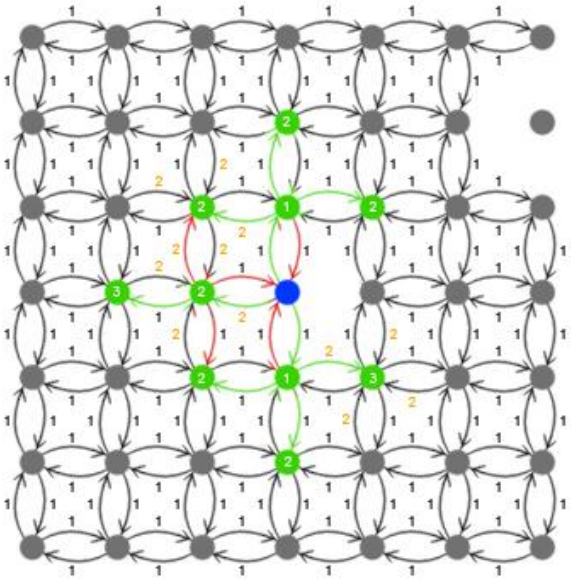
In the picture 4.3 the blue vetrices is the starting point of the unit, all the the value of the represent the cost of entering the panel. Each of grey dots hold a value ∞ and the blue dot hold a value 0.



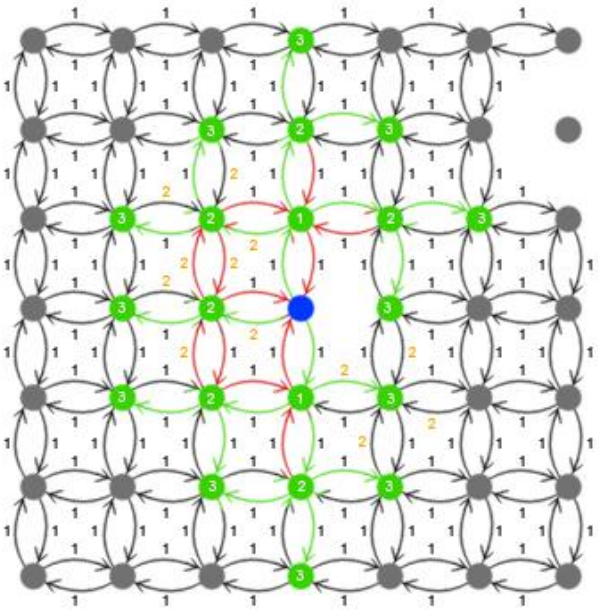
Picture - 4.4 : phase 1 of modified Dijkstra's algorithm

In picture 4.4 the blue dot is processed, all the vetrices that connected with it as the *end* will be given the smallest value between its current value and the sum of

start vetrices plus the edge value.



Picture - 4.5 : phase 2 of modified Dijkstra's algorithm



Picture - 4.6 : phase 3 of modified Dijkstra's algorithm

In picture 4.5 the graph is processed as in the phase 1. The red arrows is the process that doesn't the previous result, while the green arrows shows all the successful process. All the process will stop if all the remainings unprocessed vetrices already have a value equal to or larger than the unit's maximum movement range. The blue dots represents all posible target panel.

There is only a minor change in the Dijkstra's algorithm. The modified Dijkstra's algorithm can be seen in the textbox below.

```
.
.
.
while z ∉ S and L(u)<MaxMovementRange
        u := a vertex not in S with L(u) minimal
        S := S ∪ {u}
        for all vertices v not in S
.
.
.
```

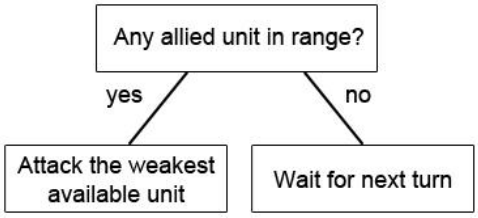
By the end of the algorithm, the possible vetrices is the one that have the value equal to or less than the maximum movement range of the unit.

B. Enemy's behavior

By analyzing several type of enemy behavior in the Fire Emblem game. The writer can conclude there are XX type of enemy behavior. These are some of the enemy behavior that can be found in the game :

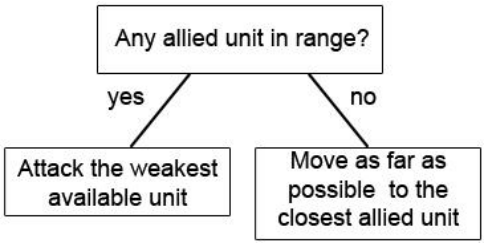
- Normal type

  This type of enemy is the most common type of enemy in early game. This type of enemy is only going to move and attack the weakest available allied unit if there is an allied unit in its range of attack. The decission tree for this type of unit is as shown below:



- Aggressive type

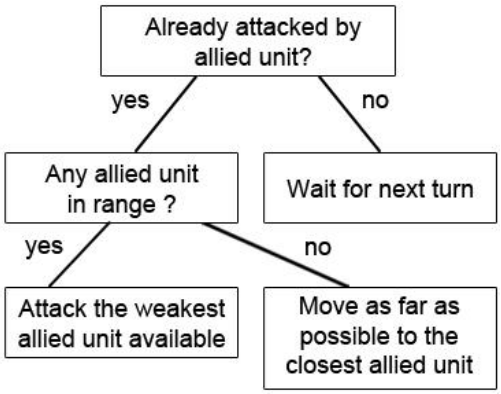
  This type of enemy is going to attack the weakest allied unit available in its range. If there is no allied unit availble in its range then it will move to the closest unit arround and wait for the next turn. The decission tree for this type of unit is as shown below:



- Defensive type

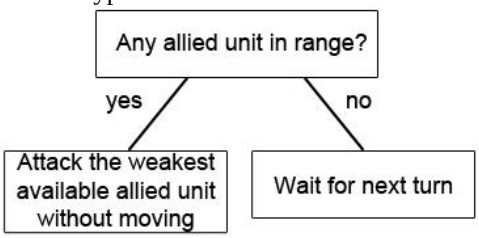
  This type of enemy usually available as an unit that can be recuited. This type of enemy will not take action at all unless it is being attack first by allied unit. Once it got attacked, its behavior change

into aggressive type. The decission tree for this type of unit is as shown below:



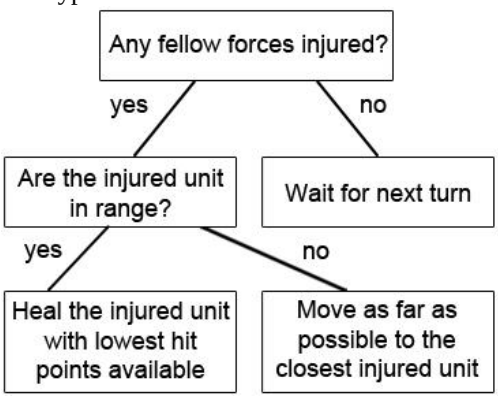
- Stationary type

  This type of enemy is usually found in the boss of each stage or as a balista. This type of enemy will not move from its position, but it will attack the wakest available allied unit arroud. The decission tree for this type of unit is as shown below:



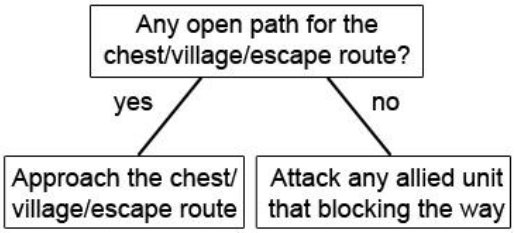
- Suprotive type

  This type of enemy seek their fellow forces with lowest health point arround and support them, if there is none available in its range then it will come closer to the nearest fellow forces with non-full health point, if it also not available then it will do nothing and wait for another turn. The decission tree for this type of unit is as shown below:



- Special type

  This type of enemy unit usually found as a thief. They only going to move closer to its objective (can be a vilage, chest or run away). If its objective panel is being block then it will attack the unit that blocking the way. The decission tree for this type of unit is as shown below:



V. CONCLUSION

The game mechanism of Fire Emblem can be explained using graph theorem and tree theorem. Dijkstra's algorithm can be used in path finding algotihm in strategy game that calculate the terain type to determine all possible objective panel for movement. Decision tree can be used as the base of every action taken by the enemy.

REFERENCES

[1] Rosen, Keneth H., *Discrete Mathematics and Its Applications, 7th ed.* New York: McGraw-Hill, 2012, pp. 641–651.

[2] Munir, Rinaldi, *Diktat Kuliah IF2091 Struktur Diskrit*. Bandung: Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2008

[3] http://asia.gamespot.com/fire-emblem-shin-monshou-no-nazo-hikari-to-kage-n/
accessed on 18 December 2012, 19.00 PM

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Desember 2012

Yodi Pramudito
13511095