Tactic AI

What's Tactic AI?

Tactic AI stands for Tactical Artificial Intelligence, but what does that mean? In RTS games like Age of Empires II, the way the enemy units act and react to our own troop is handled by tactic AI. On the other hand we have strategic AI, which would kind of represent the "enemy player", as it has control over enemy resources and such. The tactic AI in games defines -for example- which path will a enemy follow to get to you, taking into account your weapon, his weapon, and many other variables. You can probably see that almost every game with some kind of enemy has tactic AI. With that in mind, there's endless ways to program our AI, and nowadays it's mostly done through scripting, pre-generated paths... However, in here we will do it the old-fashioned way.

How to get started

First and foremost, you'll obviously need two units: one of your army and an enemy. They can be very basic (just a melee unit), or very complex (a flying mage) and that will decide the complexity of your AI when you program it, the possibilities are endless! For this research I'd recommend that they have at least a radius (how far can they spot enemies), HP and attack values.

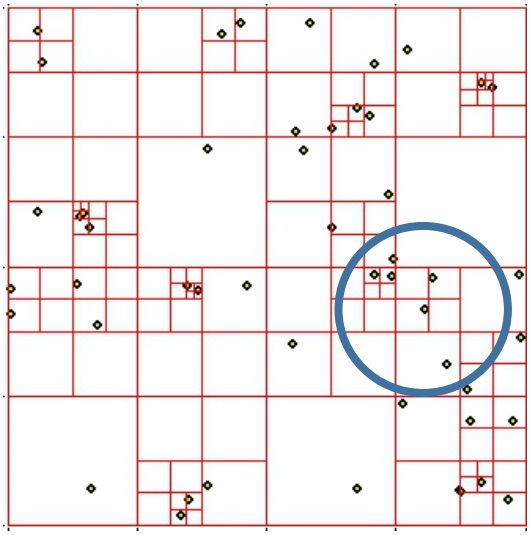
Identifying enemies - Brute Force Way

So now that you've got your units, the first step would be to check if there are any enemies around them. The brute force way to do is to check manually all the tiles inside their radius, starting from their position and going out. For this reason, we'll use the BFS algorithm with some tweaks as it serves the purpose just right.
For those of you who don't know or have forgotten what BFS is and how it works just read this link:
BFS Algorithm article

You should check in each tile if there's a unit there, and stop calculating as you've already found the nearest enemy. However, this way to do it is very expensive resource-wise, and you'll have problems when many unis are deployed on the field. For now, just limit the time the calculations are done, for example, to 1 each second.

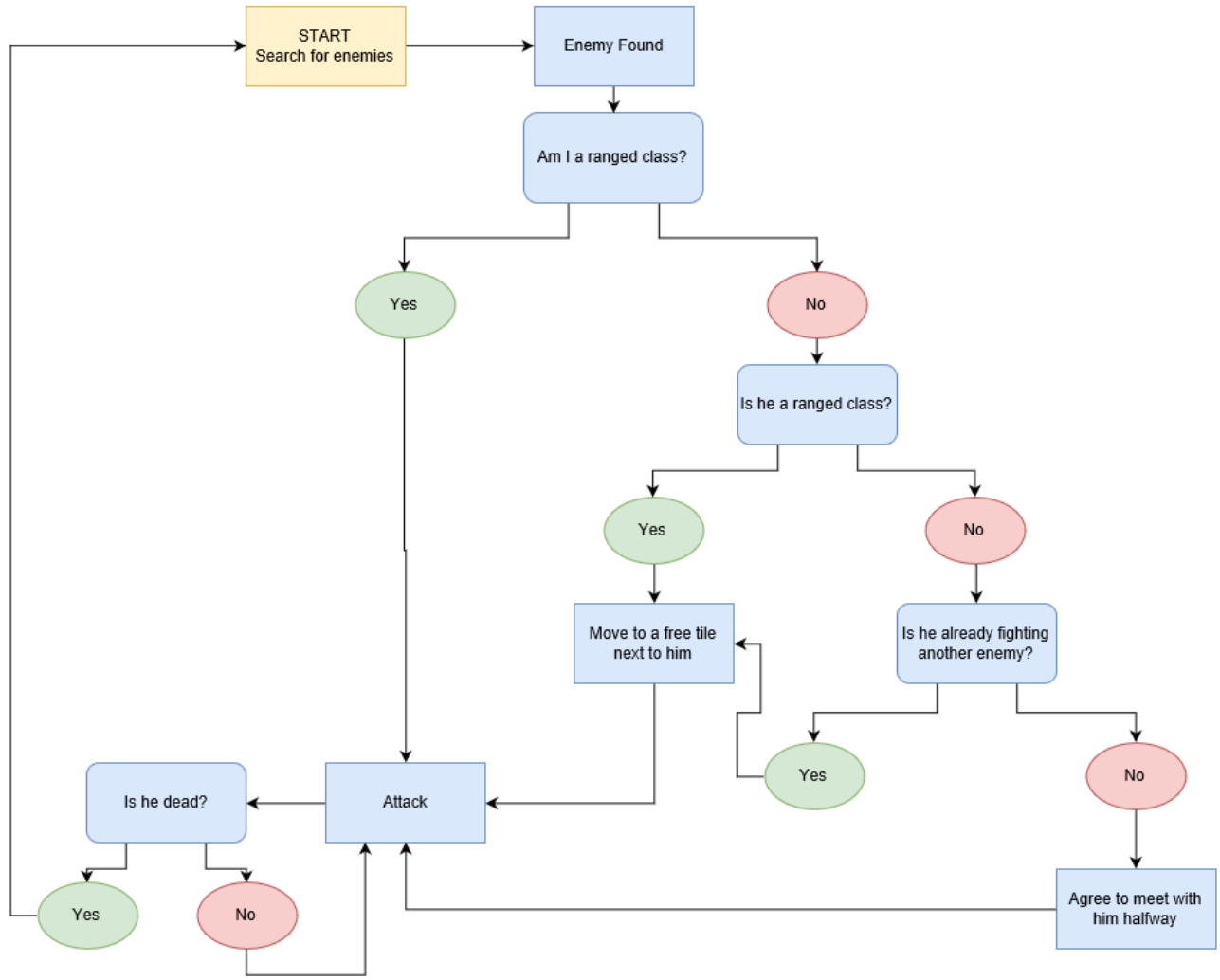
Identifying enemies - The Right Way

If you want to get better results, you should introduce a Quadtree system to your code. Given the radius of your unit, you could find how many nodes are inside it and just look for the nearest enemy in said nodes (if there's any enemy). This will speed things tremendously, as you'll just check for enemies in nodes instead of check every tile in a radius.
If you want to implement a Quadtree, follow this research by a fellow classmate (Xavier Olivenza):
Quadtree Research



We have identified an enemy... Now what?

Now things get complicated... and kind of messy. As said before, there's endless ways to program AI, but we will follow a basich scheme for now. Let's use a diagram to illustrate a basic fight between melee/melee, melee/ranged, ranged/ranged.



As we can see, there are multiple checks along the way we need to make before we begin attacking. Let's do some pseudocode to help you get the idea:

Things we should have:

- Pointer to the enemy unit
- STATE for the unit class

If the enemy isn't attacking anybody else
{
If we're both ranged

{

Change both STATEs to ATTACKING
}
Else if he's ranged and I'm not
{
His STATE changes to ATTACKING, we move to a tile next to him, STATE is MOVING_TO_ATTACK

}

Else if we're both melee
{
Find 2 tiles halfway between the units and generate a path to it (to your unit and to the enemy). Both STATEs are MOVING_TO_ATTACK
}

}
Else if he's attacking somebody else

{
If he's moving

{

Move to a tile next to his destination, STATE = MOVING_TO_ATTACK
}

Else if he's attacking

{

Move to a tile next to him, STATE = MOVING_TO_ATTACK

}

}
It should be noted that we use MOVING_TO_ATTACK so when our Move() function ends (we get to our destination), the unit changes its STATE to ATTACKING instead of just IDLE or NONE.

Attacking

This part's easy. You just gotta make sure that your enemy is still actually alive (his HP is above 0 and he's not a nullptr) and that he's still in your range (repath if he isn't) and then use a timer to control how much damage you do per second (dps). You could also apply damage at any other rate if you wish, just use a timer (although less interval of time between damage calculations means it'll be more resource intensive if there are several units fighting all the time)!

You're done!

When one of the units dies, the other one goes back to an IDLE state and starts searching again in his surroundings for other possible enemies. However, if for example it was a guard who defends an entrance, you could save his original position and just make him come back after defeating an enemy by generating a path to it. You may do as you wish with your AI.

Interesting Links

Game AI: The State of the Industry
Game AI: The State of the Industry Part II
Good examples of modern AI