

**Artificial Intelligence for Games**

Year 3 (2019/20), Semester 6

**SCHOOL OF INFOCOMM TECHNOLOGY**

Diploma in Information Technology

**ASSIGNMENT**

**Duration: 6th December 2019 – 22nd January 2020**

**Weightage: 40% of overall grade**

**Format: Individual – 60%**

**Group – 40%**

**Deadline: Report and files – 20th January 2020, 10am**

**Presentation – 20th-22nd January 2020**

**Competition – 28th January 2020**

|  |  |  |
| --- | --- | --- |
| **Team Name** |  | |
| **Student Name** | **Student No.** | **Character** |
| Tan Cheng Hian | S10179071A | Knight |
| Han Qin Ding | S10180179K | Archer |
| Lau Cheng En | S10179510J | Wizard |

Contents

[Overview of Team Strategy 3](#_Toc30511691)

[Knight Strategy 5](#_Toc30511692)

[Overall approach & State Diagram 5](#_Toc30511693)

[Approach to leveling up 7](#_Toc30511694)

[How to decide on a target to attack 8](#_Toc30511695)

[Use of healing 9](#_Toc30511696)

[Wizard Strategy 10](#_Toc30511697)

[Level-up Strategy 10](#_Toc30511698)

[State Diagram 12](#_Toc30511699)

[Meditating State 13](#_Toc30511700)

[Seeking State 13](#_Toc30511701)

[Attacking State 13](#_Toc30511702)

[Retreating State 17](#_Toc30511703)

[Defending State 17](#_Toc30511704)

[Archer Strategy 18](#_Toc30511705)

[State Diagram 19](#_Toc30511706)

[Seeking State 19](#_Toc30511707)

[Attacking State 20](#_Toc30511708)

[Kiting State 21](#_Toc30511709)

[Fleeing State 22](#_Toc30511710)

[Level up strategy 22](#_Toc30511711)

# Overview of Team Strategy

The general strategy our team adopted was to have the Wizard follow the Knight in the middle path while the Archer takes another path (bottom or top).

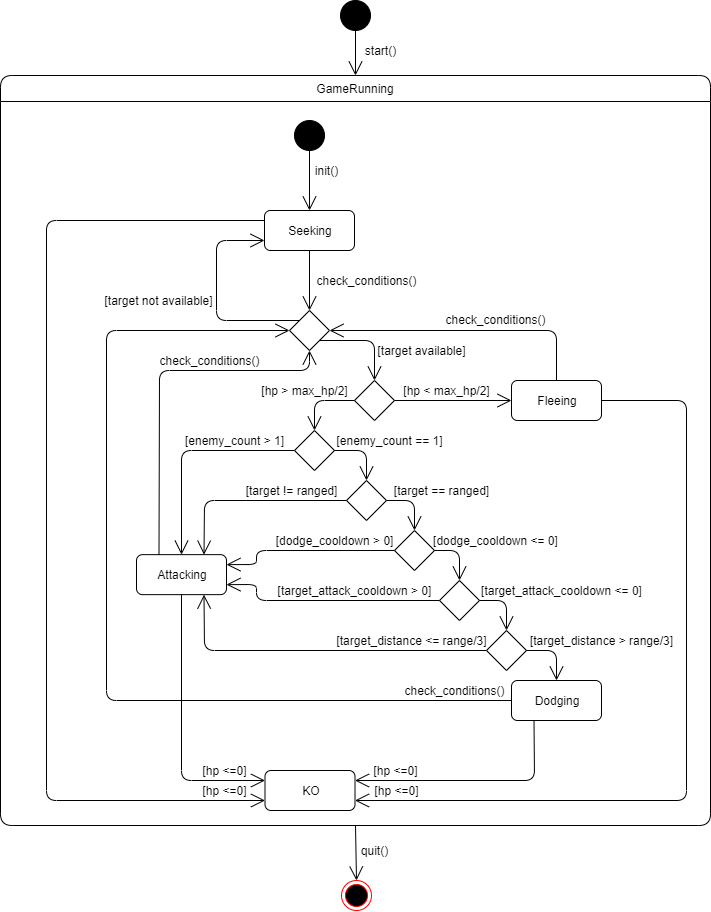
The reason behind this is that when testing different strategies between the 3 heroes, we found that the wizard cannot sustain on its own especially when up against a knight. This means that the wizard will need to have a “tank” in front of it for the best results. This allows for the wizard to clear the path while the knight keeps enemies away from the wizard, giving us a decent damage output and giving us a better chance at pushing towards the opponent’s base. Additionally, the archer has been coded to have it’s own dodging capability so it is well-enough going on its own path.

However, in the case where we are up against the hard mode (RED\_MULTIPLIER of 1.15). Our whole team will solely focus on going by the top path. This was found to be the most consistent in terms of wins as our normal strategy (wizard and knight together and archer alone) ends up losing to hard mode quite a few times, mainly due to the whole opponent team rushing down to our base on a different lane from us. There were even some times where we only won by a small margin. Hence, to ensure that we can beat the hard difficulty, we specifically changed our strategy.

**Note: Speed multiplier of 1.0 is required for our codes to work properly**

# Knight Strategy

### Overall approach & State Diagram



The knight uses a decision tree in the state diagram shown which will determine the state it will enter. There are a total of 4 states that the knight can go into. Seeking, fleeing, dodging, and attacking.

While in seeking state, the knight will travel from the node it is nearest to towards the target node which is the enemy base. Fleeing is similar to seeking except the target node is now the knight’s spawn node. A separate graph from the default orc path has been made for the knight so that it needs to travel a shorter distance to reach the nearest node and continue on its path, which minimizes the time needed to reach enemies and to flee from enemies. This path graph is similar to the default orc path graph provided, except with more nodes between every node.

The dodging state is only used for ranged enemies. It gets the vector from the knight’s position to its target position, then it will dodge 90 degrees from the vector, with a 50% chance of the rotated vector being multiplied by -1 to make its movements seem more realistic and spontaneous. The decision tree that the knight follows is shown above. The priority for the knight is firstly whether there were any enemies in its detection range. Then the second priority is whether the health is below half. The reason for this is to check whether the knight has to flee when it is low on health. This will help to preserve the health of the knight so that it is able to tank more damage for the wizard and archer.

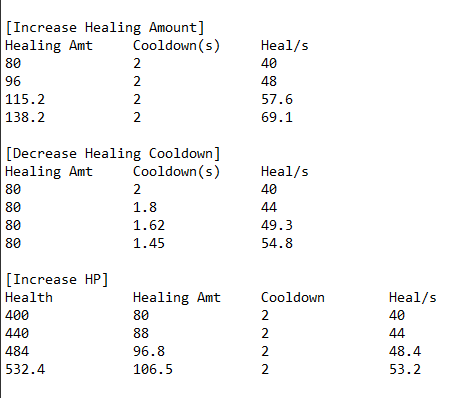
The rest of the decision tree after the transition where enemy == 1, is a decision tree to deal with ranged enemies. This is to check if the conditions are fulfilled for the knight to enter the dodging state. The knight will check if its target is a ranged enemy. If yes, it will then check for its attack cooldown. The attack cooldown can indicate whether the enemy is able to attack the knight, and hence it will dodge accordingly.

Before dodging, the knight will check for the distance between the target and itself. Since the knight’s movements are slow, it will get hit by the projectile regardless of whether it is dodging if it is too close. Therefore, the knight will stop attacking when it is a third of its range to the target. As the knight will be hit either way, there is no need to dodge and delay its transition to the attacking state, where it is able to deal damage to its target.

The attacking state gets a list of enemies that are near the knight, and targets them based on priority. This state has not been changed much, since the only way the knight can attack is by colliding into its targets. All of the states are constantly checking for when the knight is supposed to transition to another state based on the situation at hand to make sure that the knight is at the state most suited for dealing with its current situation.

### Approach to leveling up

The priority of the knight’s stats are decided based on the percentage chance they can level up. For healing cooldown and healing, these are the 2 most important aspects for the knight since it helps to keep it alive and absorb more damage. Hence, each get a 35% chance of being leveled up.



From the calculations shown, increasing healing amount will let the knight heal more per attack, while healing cooldown will let the knight attack faster. Hence, these will be the 2 most important attributes that will allow the knight to tank for longer periods of time. HP is slightly less important as the knight already has relatively high health, and leveling up hp will only give its healing a small boost. It is given 20% chance as it still allows the knight to tank attacks for longer.

Additionally, the healing cooldown will only be leveled up if it is higher than its melee cooldown. This is to minimize the time taken to attack and heal. Since the cooldown time taken for the knight to heal is 2 seconds at the start while the attack cooldown is 1.5 seconds, naturally healing would be done a lesser number of times. As the knight will only heal and attack when its healing cooldown is 0, it will try to level up the healing cooldown to be as close to the melee cooldown as possible. For the last 10%, it is split evenly between melee cooldown and melee damage. As attacking is not as important, there is no need to increase the damage or attack faster

### How to decide on a target to attack

The knight checks a list of available targets that it can attack, then checks what kind of enemy is available in that list. The priority of targeting from highest to lowest is base, towers, wizard, archer, knight, orcs. The reason is because the base is the most important to take down, hence when the knight is near the base, it will ignore all the other enemies and attack the base directly. The same reason is used for the towers.

Ranged enemies are of higher priority since they are able to attack the heroes that the knight is defending from a distance. To shield them from getting hit, the knight will have to get rid of the ranged attackers first. The wizard is more important than the archer due to its ranged attack, which can cause a lot of damage to multiple units and can wipe out a squad of orcs easily. The lowest of the priority are the orcs and knights. This is because they will be distracted by the knight at the front regardless, so they do not require any special attention. The mage behind the knight will attack the melee enemies attacking the knight.

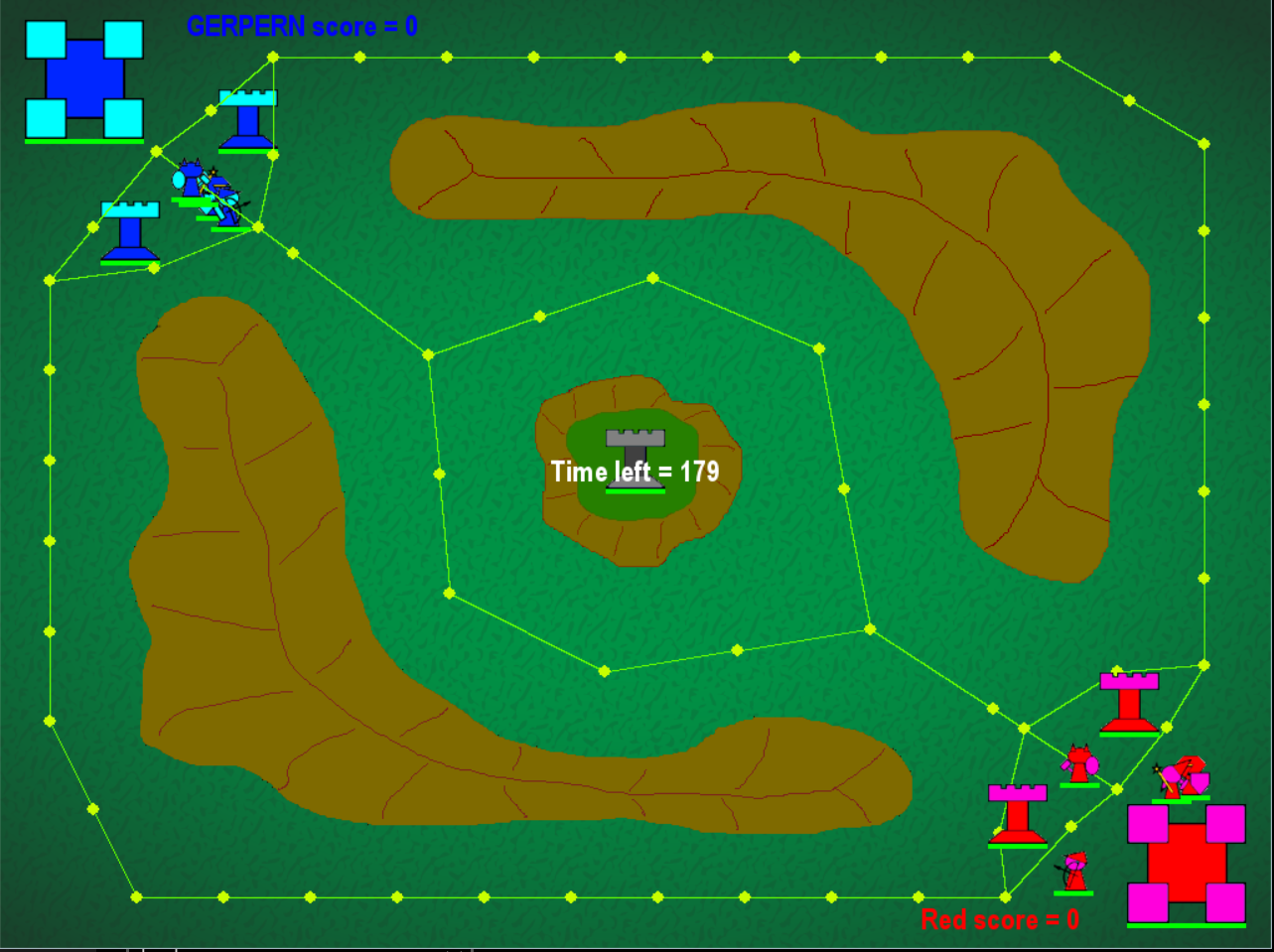
### Use of healing

Healing is only done in the fleeing state and the attacking state. The knight will heal immediately after it has done a melee attack, and it will only attack again when its healing cooldown is 0 or lower. This is because there will be a shorter time spent where the knight is unable to attack.

---------------------------------------- End of Knight Strategy ----------------------------------------

# Wizard Strategy

As mentioned above, the general strategy for the wizard is to follow on the same path as the ally Knight’s. Based on the state of its base compared to the opponent’s, the wizard will switch to a Defending state to walk around the base instead of going out *(more will be explained below)*.

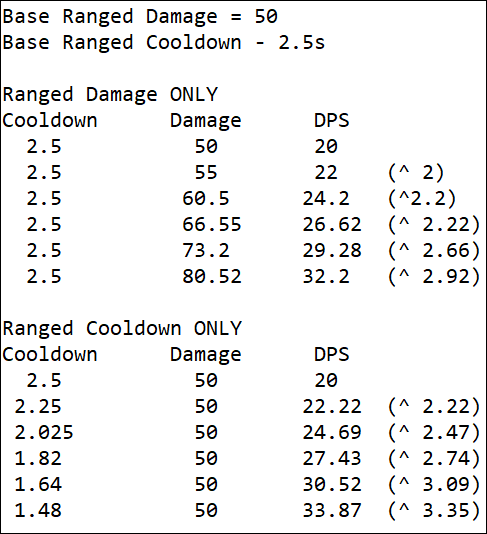


The graph and paths for the wizard were also modified to make the wizard’s movements and actions more efficient. I cut off a few of the corner nodes so that the wizard can travel faster (especially with its low speed) and added more nodes in between the existing ones so that it doesn’t keep moving back very far when finding the nearest node to go to.

Lastly, I added the paths on both bases for the wizard’s “Defending” state.

### Level-up Strategy

Based on calculations I made as shown below, we can see at a glance that upgrading “Ranged Cooldown” yields better results in terms of damage dealt per second (DPS). I believe that it should generally be better to think in terms of DPS as it is the total damage output that really matters at the end of the day. We will keep this in mind for now.



The default healing cooldown for all characters stand at 2 seconds. This means that for the wizard, as long as it maintains a ranged cooldown of at least 2 seconds, it can immediately heal after each attack without having to deal with the attack penalty as the healing would be available again by the next attack made. Therefore, by upgrading the “Ranged Cooldown” twice, I am able to maximise the healing potential of the wizard by attacking and healing every 2.025 seconds (refer to image).

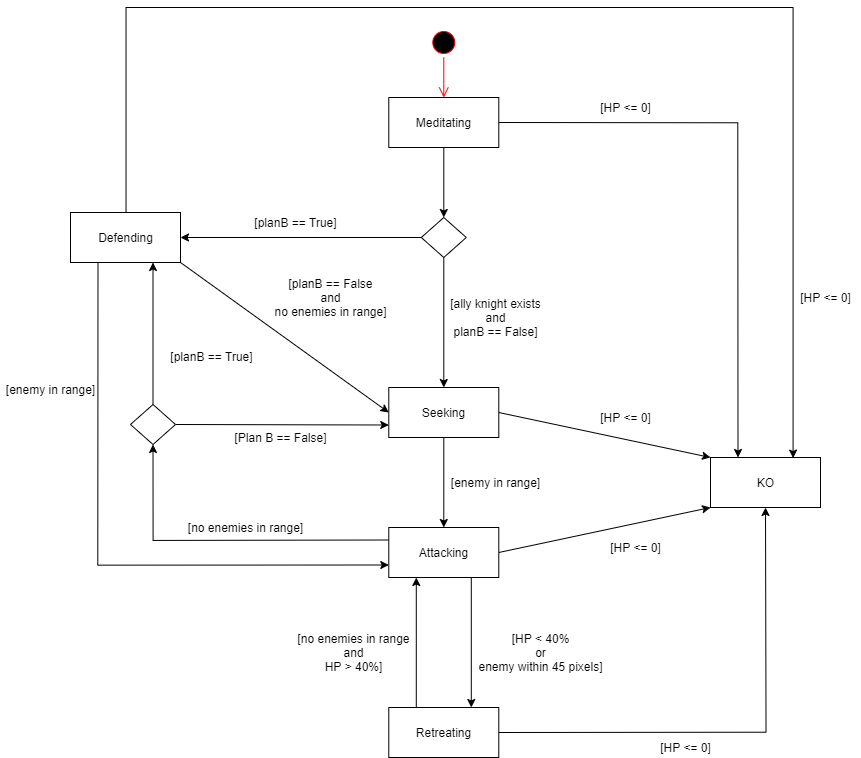
Whichever stat I upgrade after these 2, I want to make sure that I maintain this healing and attacking “combo”. My options are either to upgrade “Ranged Damage” throughout or to alternate between “Healing” and “Ranged” cooldown. My final decision was to go with “Ranged Damage” as I feel like I am compensating too much on my DPS for more healing and was better off dealing higher damage per shot. Hence, the level-up strategy for wizard is as follows:

Ranged Cooldown → Ranged Cooldown → Ranged Damage → Ranged Damage → ...

### State Diagram

Before going into how each of the states work, here is an overview of how the wizard operates and when it transitions to its various states.

*Note: “planB” is a variable that gets set to True based on a certain threshold of the base’s state (defined by number of towers and hitpoints). Otherwise, it will be set to False*

****

### Meditating State

This is the initial state of the wizard. This is sort of like a planning state where the wizard decides whether to enter “Seeking” state or “Defending” state. Upon entering the state, a check is done to see whether it’s team is losing based on the number of towers and the base’s health. If it is losing, it will go into “Defending” state. Otherwise, it will set its path to the same one as the ally Knight and enter “Seeking” state if the Knight exists.

### Seeking State

Based on the path set (same path as ally Knight), the wizard will advance towards the opponent’s base. If any enemies are in the wizard’s range, it will switch to “Attacking” state.

Additionally, it will heal, but only if its health is not full, and if it will not meet any enemies in the time span of its healing cooldown. In order words: I do not want to be healing if I am about to enter a fight because I will not be able to attack until my healing cooldown refreshes. How this was done was to calculate the wizard’s projected position in “healing cooldown”(e.g. 2) seconds multiplied by its current velocity. Afterwards, I check whether there are any enemies in its range at that position.

### Attacking State

As mentioned above, the wizard heals immediately after attacking as it won’t be affected by the attack penalty from healing.

The other change I made was to have the wizard stop moving when an enemy is 85% into its target range. I found this to be better because many times, it misses out on potential “good” spots to hit because it stops when it sees the first enemy, causing it to be unable to target the rest behind if there are.

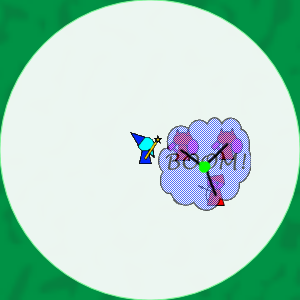
This is where the strength of the wizard comes in to play - its ability to deal damage in an area to multiple enemies. If done properly, the wizard can turn battles into its favor, being able to damage all enemies in front of it in a single attack. Hence, the strategy here is to target the spot that can hit the most enemies. My approach on the targeting for its attack was to use a recursive function as shown below:

findBestTarget(directionToCheck, bestSoFar, numTargets, totalTargetHP, self)

The function can be broken down into 2 parts: Checking of the midpoint of all enemies in range and then finding the 2nd best spot to target.

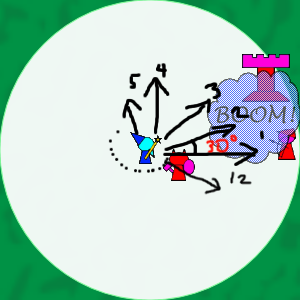
I will further explain the way it works through the scenarios below.

**Scenario 1:**

****

In this scenario, there are 3 enemies in the wizard’s range. The best spot to target would obviously be the midpoint of the 3 enemies. So when the function is first called, it will check for the midpoint of all enemies in range. I basically calculate the sum of their “x” and “y” coordinates separately and take the average of each to get the midpoint. If the distance from the midpoint to all the enemies are within the radius of the fireball, I return the midpoint as it means the fireball can hit everyone.

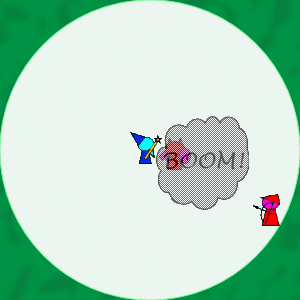
**Scenario 2:**

****

In this scenario, the midpoint is unable to target all enemies. The function will then call itself again, passing in its midpoint, the number of targets it can hit if the midpoint is targeted and the total hitpoints of targets hit.

The logic behind the recursive function is to start from an angle of 0 degrees and check from its position to the edge of its range (around 4 checks to reach the end of its range). If there is a spot where it can target more enemies then the current best coordinate, it will update the variable together with the other details. Afterwards, the function will call itself again but this time with an angle of + 30 degrees. The function repeats until it completes the whole circle, returning the best target it has found. In this case, it would be to hit both the tower and archer.

**Scenario 3:**

****

The last scenario is when the current best target and the one its checking has the same number of enemies hit, but they have differing hitpoints. The wizard would then target the group of enemies that has lower total hitpoints instead. The reason behind this is that if I eliminate the lower health enemy first, I would have fewer enemies to deal with later on, ultimately increasing my chance of surviving.

Again, this was done by passing in the total hitpoints of all targets in range when the current best target is updated. Similarly, if I were to choose between 2 orcs and a tower, I would focus on killing the orcs first.

If the wizard’s hitpoints happen to be below 40% during a battle or if an enemy is close very close (usually for melee characters), it will move to “Retreating” state. Note that this does not trigger when the wizard is “Defending”

When there are no longer enemies in range, the wizard will return to “Seeking” state. In the case where the variable “planB” is True, it will return to “Defending” state instead.

### Retreating State

In this state, the wizard’s priority is to stay alive. Since its healing cooldown would almost be the same as its ranged cooldown, it can afford to attack while running away from an enemy. Therefore, if there is an enemy in range, the wizard will continue attacking but at the same time move towards its own base. If there are no enemies, it will just heal itself.

When the wizard’s hitpoints are back to 60% and above and when there are no enemies in range, it will go back to “Seeking” state.

Likewise, if it's hitpoints are back to 60% and there are enemies in range, it will go to “Attacking” state.

If the wizard’s hitpoints happen to be below 40% during a battle or if

When there are no longer enemies in range, the wizard will return to “Seeking” state. In the case where the variable “planB” is True, it will return to “Defending” state instead.

### Defending State

The wizard goes into this state only when the following conditions are met:

* It’s base has no towers remaining and it’s opponent has at least 1 tower remaining
* It’s base has < 50% hitpoints and is lower than the opponent’s

In this state, the wizard circles around the front of the base to try and keep it’s base alive and clear of enemies.

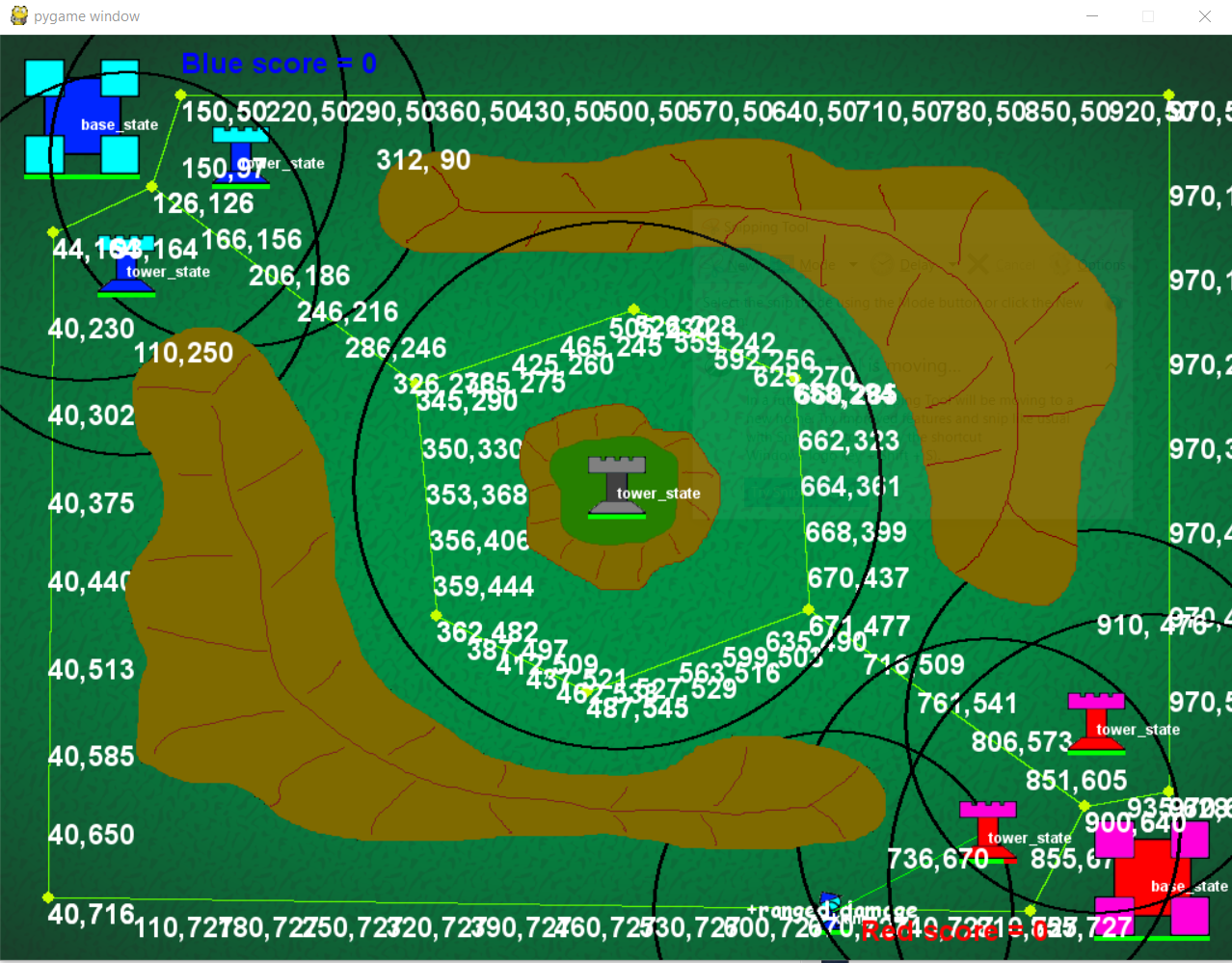
If an enemy is in range, it will move into “Attacking” state.

When the above mentioned conditions are no longer true, the wizard will go back to “Seeking” state and function normally.

---------------------------------------- End of Wizard Strategy ----------------------------------------

# Archer Strategy

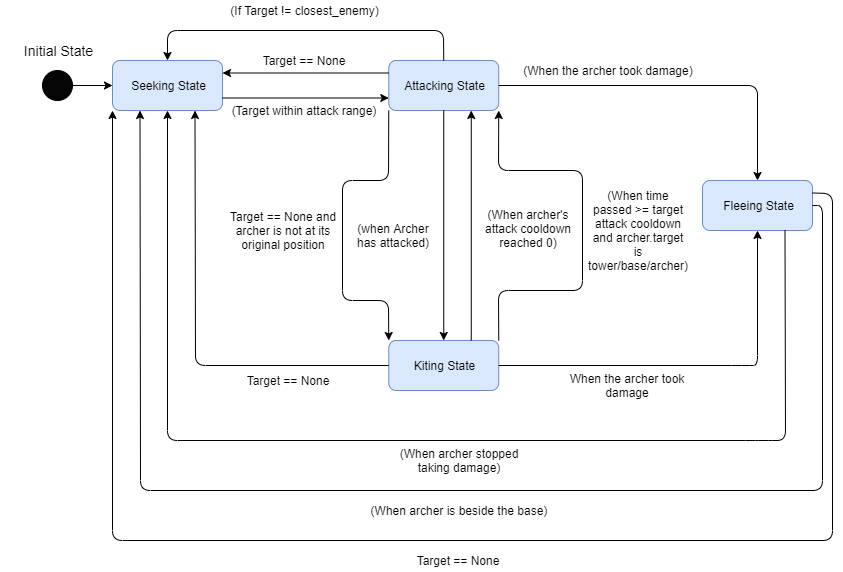
As mentioned above, the general strategy for the archer is to go top lane or bot lane, using the kiting mechanics to dodge the projectiles.



I’ve created a new pathfinding graph text file for my archer, it contains more nodes. The purpose of adding this new file is to fix my seeking state problem. Whenever the archer switches back to seeking state (from attacking/kiting/fleeing), it will move to the closest node. Most of the times, the closest node is far away and the archer has to move all the way back to the node, hence the archer will constantly be moving back and forth around the area, hardly making any progress towards the enemy base. Hence, by adding more nodes, the archer does not have to move all the back.

*(more about the states will be explained below)*.

### State Diagram



### Seeking State

* Seeking State is the initial state of the Archer. When the Archer enters the seeking state, it will generate its path using the AStar pathfinding algorithm. While it is in seeking state, it will constantly check the map for the nearest enemy that’s in the same lane as the Archer.
* By using a new get nearest enemy code for the archer, I will only compare the enemies that are in the same lane as the archer. The purpose of using this new “get\_nearest\_opponent \_Archer” function is to prevent the archer from getting stuck to a Mountain. Before this code was implemented, I’ve found out that the archer tends to get stuck to the mountains when its chasing an enemy from a different lane.
* After testing out the code, I’ve found out that the archer will ignore any enemies that are attacking our base from a different lane. This allowed the enemies to continue attacking our base, which caused us to lose. Hence in order to fix this problem, I edited the get nearest opponent function again. Now, if the archer is standing near its own base, the “get\_ nearest\_opponent \_Archer” code will compare all the enemy entities,regardless of their lane, and return the closest enemy, allowing it to set the enemy that’s attacking our base as the target
* I am able to determine the entities position using a getcharLane\_Position function. By comparing the entities’ and the 2 mountains’ top left and bottom right coordinates, I will determine the entities’ position. (e.g. if the entity’s top left x is higher than the top mountain’s bottom right corner x, it means that the entity is on the right of the top mountain/in the top lane) Initially i tried to use the rect of the mountains, but the coordinate provided was inaccurate, hence i had to hardcode the mountains' corner coordinates
* Once the target is set and it is within the archer’s attack range, the archer will switch to attacking state
* The archer will check if it’s health is below 100 and it has no target, if yes, the archer will heal. This code is to ensure that the archer will heal up when its low and the healing cooldown will not stop the archer from attacking

### Attacking State

* When the archer first enters attacking state its starting health and the starting time will be recorded. These two values will be used to determine whether the archer should flee. The state will constantly compare current time with the starting time, and if the difference went pass 0.5 secs, it will check whether the archer health decreased by comparing the starting health and current health. If yes, the archer will go to fleeing state.
* The state will check if target is within archer’s attack range, if yes it will check if the archer’s attack cooldown has reached 0. if yes again, the archer will attack and switch to kiting state. If the target is not in arhcer’s attack range it will constantly chase the target. If the target is dead, the archer will switch to seeking state to seek new target
* The archer will constantly check if its target is the closest enemy. If its not, the archer will return to seeking state to get new target. The purpose of this code is to ensure that archer does not chase the target when there are other enemies nearer to him. if he were to chase, he will be attacked while he's moving through the enemy units

### Kiting State

* When the archer enters kiting state, it will record the starting time and starting health of the archer. The state will constantly check if the archer’s health decreased by comparing current health and starting health. If yes, it will switch to fleeing state
* When the archer was in seeking state, the enemy type was determined and recorded using the target’s name. the enemy type will be used to determine which kiting function the archer is going to use (aggro, safe). if the enemy is an archer, tower or base, the archer will use the aggro\_kiting function. The function uses the vector of the path between the archer and the target to determine the vector of the projectiles. The state will calculate a kite\_position and record the current position. The archer will then move back and forth between the 2 spots to dodge the projectiles. if the target is an orc,knight or wizard, the archer will use the safe\_kiting function to move back to base. After manually moving the archer to dodge the wizard’s projectiles, I've found out that the only way for the archer to dodge the wizard's projectile and its explosion range is to move back.
* The state will check if the archer attack cooldown reached 0, if yes, it will switch to attacking state. This is to ensure that the archer is attacking at the maximum rate even while it is kiting
* The state also checks if the time passed (derived using the starting time and current time) is >= the target’s attack cooldown. If yes it will indicate the aggro\_kiting function to move in the opposite direction. This allows the archer to move to a new location whenever the target attacks/at the right time, allowing the Archer to dodge the projectiles. The archer will switch to attacking state once the aggro\_kiting function has been indicated.
* If the target is dead, the archer will switch to seeking state to seek new target

### Fleeing State

* Just like attacking state, when the archer enters kiting state, it will record the starting time and starting health of the archer. However, they are used to determine whether the archer should stay in fleeing state. After 1 seconds has passed it will check if the archer’s health is higher than the starting health after healing up, if yes, it will switch to seeking state.
* The archer will switch to seeking state when the target is None.
* The archer will immediately switch to seeking state when is beside the base. Once the archer is right beside the base, it will stop moving (end node), hence there is no longer for it to move back/flee and should proceed to kill the enemy
* The reason i chose seeking instead of attacking or kiting, is to ensure that archer will go for the nearest enemy instead of chasing the old target that is behind another enemy.
* While in fleeing state, the archer will heal if the healing cooldown is 0

### Level up strategy

* The strategy for archer is kill the enemy as fast as it can, while trying to minimize the damage its receiving.
* The archer will not often heal as it is relying on kiting\_state to dodge all the projectiles. Hence, there is no need to level up on healing.
* In order to fully utilise the kiting\_state, archer’s ranged attack and fast attack cooldown, I will only focus on increasing the amount of damage my archer deals per second. Hence stats like the archer’s hp, speed, projectile range(increasing range barely increases the damage dealt per second), melee damage/cooldown will not level.
* However, after running tests, I've found out that if the archer’s attack cooldown were to decrease, it would reduce the distance that the archer will travel in kiting state. The archer changes from kiting state to attacking state when his attack cooldown has reached 0, and when the archer changes to attacking state, it will stop moving. This means that the lower the archer’s attack cooldown, the earlier the archer will stop moving, which makes the kiting\_state irrelevant (archer will not be moving much, hence unable to dodge projectiles).

---------------------------------------- End of Archer Strategy ----------------------------------------