SE 354 AI TANK PROJECT

Prepared for

Assistant Professor Kaya Oğuz

Prepared by

Ege Şenkul

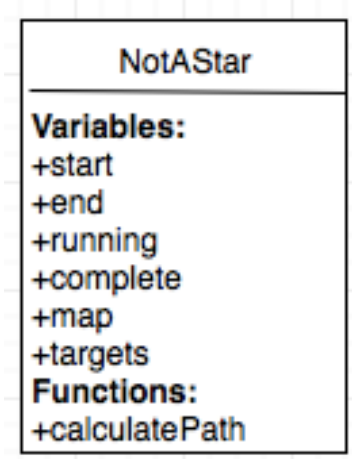
May 15, 2018

AITankScript

In that script, we have some classes, variables and functions which related to our tank.

NotAStar

It is a class that which related out tank movement. I don't use that class in my project.



Variables:

Start and **end** points which are Vector3 type variables. They are our start and end points of our route.
Running: That variable is a bool type value and it is true if our tank moving or not.
Complete: That variable is a bool type value and it is true if our tank completes its movement.
Map: It is a two-dimensional integer variable which is keeping map.
Targets: It is an ArrayList which is store targets of tank.

Function:

calculatePath: It is a function that which is calculate the path but, in that case, it doesn't do it.

AITankScript:

It is a MonoBehaviour type class. It has a bunch of variables and functions about basic information about our tank.

Variables:

Health: It is an integer type variable. Its initial value is 100. It stores our tank's health. Since it is a private variable we can get it with getHealth function.
Armour: It is an integer type variable. Its initial value is 50. It stores our tank's armour value. Since it is a private variable we can get it with getArmour function.
Puan: It is an integer type variable. Its initial value is 0. It stores our tank's points. Since it is a private variable we can get it with getPuan function.
invulTime and invulnerable: If our tank gets invurable game object. Invulnerable bool value becomes true and invulTime set to Time.Time which is current time.
Died: It is a bool type variable. If our tank is died it becomes true.
hexTime: It is a float type value if our tank get damage it is updated with Time.time.
damageMult: It is an integer type variable. It is a multiplier of our getting damage. Its initial value is 1. If our tank hit a damage object, it is updated.
astar: It is a variable which type of NotAStar. It is an object of NotAStar. We use it for movement.
myThread: It is a variable which type of a Thread. We use it for some calculations.
playername: It is a variable which type of a string. It stores name of the tank / player.
weapons: It is a private list which has Weapon type objects.
currentWeapon: It is a private integer variable which is holding current weapon index. Its initial value is 0.
Functions:
ClearVaues: It is a public function which is resetting all values. Clears weapon list and add machine gun. Set currentWeapon to 0. Assign invulnerable to false, damageMult to 1, health to 100 and armour to 50.
Update: That function runs in every frame. In that function, we set invulnerable to false if invulnerable true and difference between time and invulTime is greater than 10. Set damageMult to 1 if damageMult equals to true and difference between time and hexTime is greater than 10. If we press to keyboard button P, we make an A* calculation.

pickupItem: That function takes an item type variable or weapon type variable for input. If that function gets item type variable to input. We add item's health and armour value to our health and armour value. After that addition action if health or armour value get higher than 100 we set them to 100. If that value has an invulnerability we assign our invulnerability to true and invulTime to Time.time. If that function gets a weapon type variable for input. We assume that we don't have that weapon and then check if we have that weapon we update it's ammoCount, if we don't have that weapon we add that weapon to our weapon list.

getWeapons: That function returns our weapons list.

setCurrentWeapon: That function gets an integer value for input. In that function, we check if that value is not greater than our weapon list size and then we set currentWeapon to that index.

kill: In that function, we kill our tank object. We assign it's setActive property to false and assign dead bool value to true. Also, we write console to ¨killed¨ log file.

takeAHit: In that function if our invulnerable is true we do nothing. If it is false, that function decrease our health to damageCaused and decrease armour by damage value. If health becomes less than 0 kill function called and multiplier becomes 2. Also, that function returns damageCaused*multiplier value.

hitObstacle: That function calls when our tank hit an obstacle. In that function, we kill our tank and update our score to score minus half of our score.

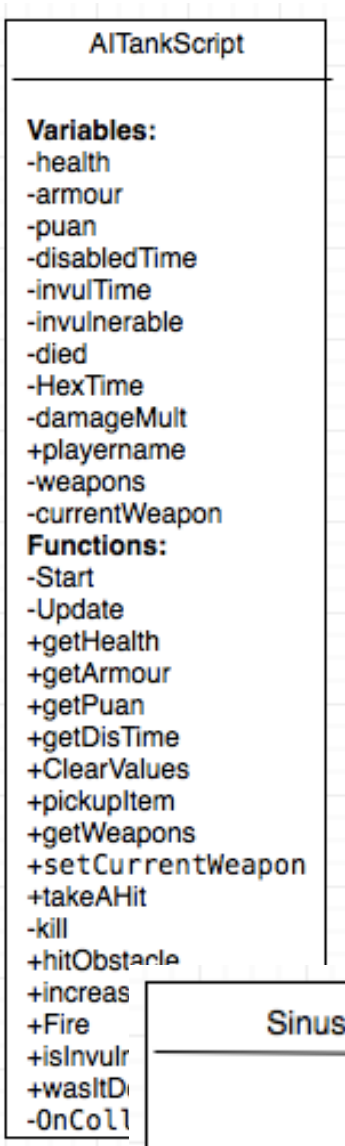
increasePoints: That function gets an integer value for input parameter. That function increase our point value to that integer much.

Fire: If our current weapons ammo count is not equal to zero or the time between that time and last fired not less than our current weapon's ammo per second value we fired. We get direction of our tank's forward. Create a new game object which is an instance of ¨bullet¨. We give some values to that game object's properties like damage, parent and direction. We give that object to a position. We update our current weapons last fired time and our current weapon is not Machine Gun we decrease that weapon's ammo count.

is Invulnerable: That function returns our invulnerable bool value.

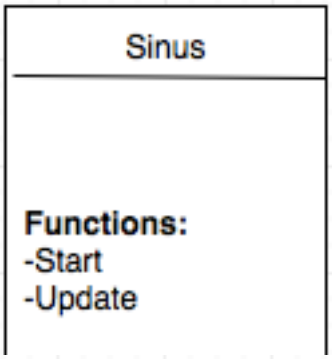
wasItDead: That function if our tank dead true and assign died value to false otherwise return false.

OnCollisionEnter: If we touch a trigger our tank will be died, and our point becomes half of it.



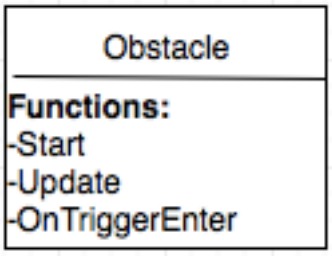
Sinus

With that script, our items can make a sinusoidal smooth movement. For that movement, we use rotate and position functions. Rotate function for rotating that objects and position function for making an up-down movement. That movement becomes smoother with using Mathf.Sin function.



Obstacle

We use that script in our trigger objects like wall trigger. With that script, if we enter a trigger we call our tank's hitObstacle function.



Tank Script

I

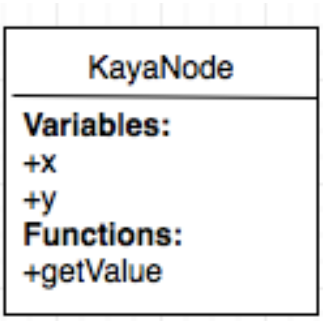
KayaNode:

This is our node class. We use that class for representing game data to nodes and vertices.

KayaNode constructor: That constructor gets a variable which type is Vector3.
Variables:
x: That variable's data type is integer. That variable represents node's x value.
z: That variable's data type is integer. That variable represents node's z value.

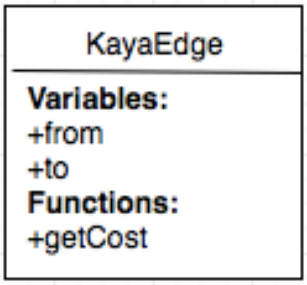
Functions:

getValue: That function returns a Vector3 type value. Which's y value equal to 0.



KayaEdge:

This is our edge class. We use that class for calculating cost to transfer one place to another place.



Variables:

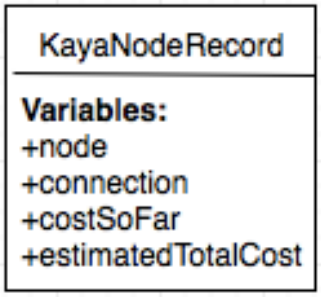
from: It is a variable which data type is KayaNode. It stores us from node.
to: It is a variable which data type is KayaNode. It stores or to node.

Functions:

getCost: That function returns difference between from and to node's getValue values.

KayaNodeRecord:

We use that class to keep record calculation of while doing A* algorithm.

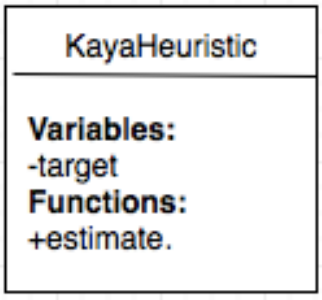


Variables:

node: It is a variable which store the node.
connection: It is a variable which store our connection.
costSoFar: It is a variable which keeps our cost so far.
estimatedTotalCost: That float variable storing our estimated total cost calculation.

KayaHeuristic:

That class is using for calculating the heuristic value for A* algorithm.



Constructor: It gets a Vector3 value for input parameter and set that value to its private target variable.

Variables:

target: It is a variable which data type Vector3. It keeps our target position.

Functions:

estimate: That function return distance between target and it's input vector's magnitude. Or it can be return distance between target and it's input node's get value magnitude.

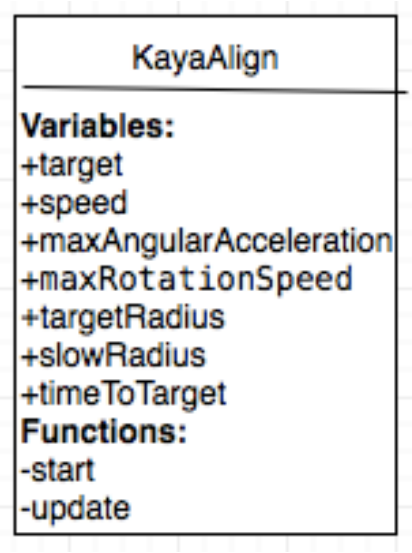
KayaStar:

This class is our A* algorithm class.

Variables:

alan: That variable is two-dimensional integer array and it store our map data in it.
Functions:

getConnections: That function returns us to a list which has our KayaEdge type connections. That list created based on all map. And that list created based on input node.
findSmallest: That function gets KayaNodeRecord list for input. Returns index of the smallest KayaNodeRecord of that list.
FindRecordInList: That function gets KayaNodeRecord list and a node for input parameter. If that node in that list return KayaNodeRecord which related to that node otherwise returns null.
aStar: That function gets start and end points for input and returns A* path with using KayaHeuristic, KayaNode, KayaEdge and KayaNodeRecord classes.



KayaAlign:

That class is a MonoBehaviour class. That class is using for alignment.

Variables:

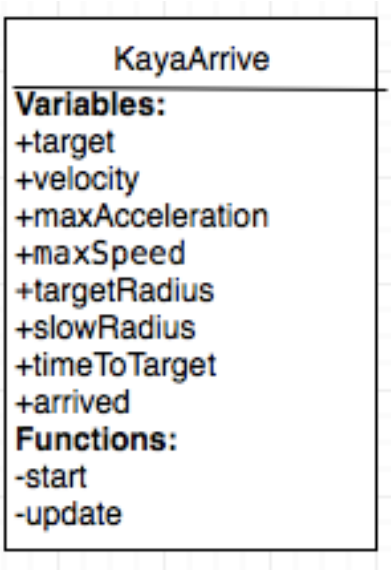
target: That float variable stores our target.
speed: That float variable stores our speed value. We use it with time.deltaTime for make smoot rotation.
maxAngularAcceleration: That integer value stores our maximum angular acceleration value.
maxRotationSpeed: That integer value stores our maximum rotation speed. We do not want to rotate too fast.
targetRadius: With that float value, we check our rotation status.
slowRadius: That float value is our radius which we start to be slowing down our rotation.
timeToTarget: We use that value to divide our angular value to it.

Functions:

Start: In that function, we assign our target.
Update: In that function, we do our rotation action.

KayaArrive:

We use that class for basic movement and checking of arrived to target or not.



Variables:

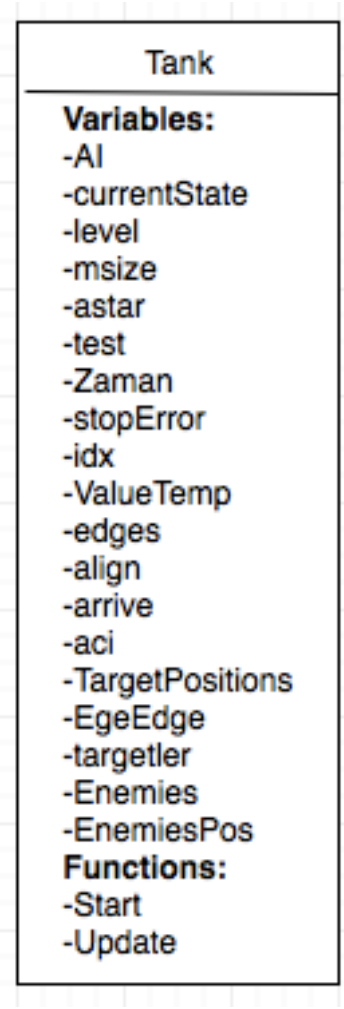
target: That Vector3 variable stores our target position.
velocity: That Vector3 variable stores our velocity.
maxAcceleration: That float value stores our maximum acceleration.
maxSpeed: That float value stores our maximum speed.
targetRadius: With that variable, we check if we arrive or not.
slowRadius: With that variable, we start our slowing action.
timeToTarget: We use that value to divide our linear value to it.

Functions:

Start: In that function, we assign our target value.
Update: In that function, we make basic movement and do checking of arrived or not.

Tank:

That class is the most important class.



Variables:

AI: We are using AI variable to call some functions and variables in AITankScript.
currentState: We are using this variable to store the current state.
Level: This variable is calling to get level component of level game object.
msize: This variable is the size of the map.
astar: We are using this variable to generate the path of the start and end positions. Start will be the tanks position and end is the targets position.
test: This variable is being used to check the arrived of the tank.
stopError: We are using this variable to fire 5 seconds.
idx: We are using this variable to find the paths index.
ValueTemp: This variable is being used to find the index of list and assign the next Tanks position.
edges: Is a list of KayaEge. This variable is calling the A* function with start and end positions to generate the path.
align: Is a list of KayaAlign. This variable is being used to alignment of the tank.
arrive: Is a list of kayaArrive. This variable is being used to make the movement of the tank.
aci: This variable is being used to get the rotation angle of the tank before makes the look at the enemy tank.
targetler: This variable is being used to assign the “Spawner” tagged objects in the list.
Enemies: This variable is being used to assign the enemies in the list.
EnemiesPos: This variable is being used to get the enemies objects which is Enemies gameobject array.

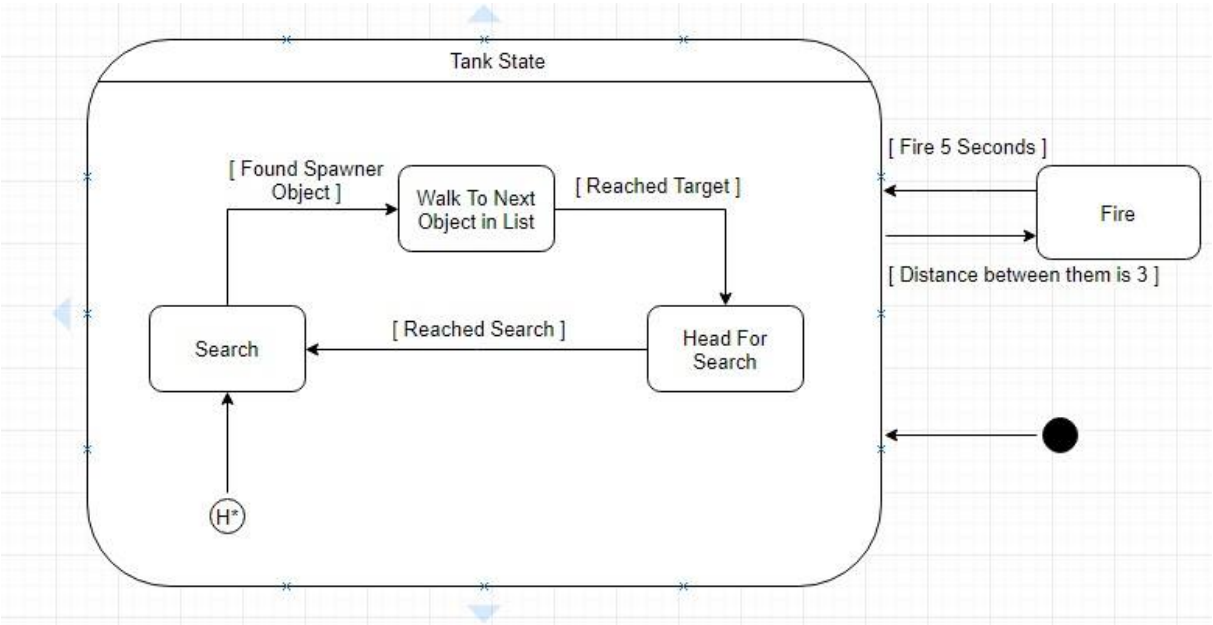
Functions:

start: In this function, we initialize our variables. And change our state to Search.
Update: In this function, if current state is not null we execute the current state.
ChangeState: This function is being used to change the current state. If current state is not null with that function our ai will exit current state and enter new state.

States:

Interface: We declared Enter, Execute and Exit functions in this interface.
Search: In this state, we are initializing our test value to true. Get positions of all “Spawner” tagged objects in a list. Then change our current state to move state with position of a “Spawner” tagged object.
MoveToTaget: In this state, first we declare the constructor for getting target position from the search state. After getting the target position we are creating the A* for declaring the paths. While moving the target position we are checking is there any enemy nearby us with distance between 3. If it is yes, we will fire 5 seconds and moving to our target position. After arrived to target position we change our state to search state and make a loop.

State Machine



Decision Tree