Enemy ATTACK Algorithm

Monday, October 15, 2018 9:27 PM

INPUTS :
enemies
isFirstAttack

OUTPUTS :
isPosition of column
CONSTRAINTS:
no one Attacking currently

After Timer = 0, [An enemy column is chosen] TO ATTACK. While ATTACK is in progress NO MORE ATTACKS will start.

Choosing Column: Initialized value
~~isFirstAttack = true~~
set Random Value 0 - 1
if 0:
set First Attack to left (-1)
Else
set to Right (1)
set ATTACK Direction to firstAttack
}
Else
{
attackDirection *= -1
}

if left:
choose col w/ minX
else
choose col w/ maxX
Return

Choose Enemies — V MAX enemy in column w/ same X to Attack
loop enemies
if enemy.X = AttackColumn
Set Attack Flag on enemy
increment numAttacking enemies

If 2 enemies in col:
top Enemy (Highest Y) will be lead w/ lower being wingman
The resultant position (Starting Position) needs to be stored so each enemy can find way back if shot down.
This Position needs a target group movement.

move AP Circle
The top enemy will go first, increasing Y until reaching radius of circle in next step.
The lead ship will then complete a half-circle in direction towards screen edge (clockwise if right (max) col, and vice versa)

TACK
The lead ship will then compute exit point to Attack player, this line will be course, lead will follow, ~~strictly to close on player~~

During the lead's move, Circle stores the wing man stays in position moving with group.
Once lead starts Attack mode, wing mate will start move, following lead's starts.

Return
If a ship Y goes negative (below screen) then it's X will match leader's position and Y will be screen height decreasing until reaching return position Y.
The ships X value will increment/decrement until return position X.
once returned, ship's isAttacking Flag is set to false.

~~If only 1 enemy in col:~~
As Above w/ our wingman

~~Shooting~~ Container of bullets - GameState / enemy static class.

During enemy logic - if attack state == ATTACK && isAttacking && isActive
for bullet bullet && bullet.position.y == enemy.y && bullet.direction == -1

GAMESTATE

AttackColumn
isAttacking
numAttackingEnemies
ATTACK Timer
timerMax
Vector<enemy*> attackingEnemies

ChooseAttackCol()
int GetRandDirection()
GetEnemyColExtremes()

ChooseEnemy(Enemy* enemy)

SetAttackLeader()

Enemy

leader - Flag
returnPosition
attackRadius
attackAngle
attackEndPoint
attackSpeed
attackInterrupt
attackState
attackDirection

dequeueToRetreat()
addInstantiating()

if (isAttacking)
if (attackTimer <= 0)
attackTimer -= deltaTime - TimerTick(deltaTime)
else
attackTimer = timerMax
isAttacking = true.

if (isAttacking)
loop enemies
if enemy.X > minCol.X
minCol.X = enemy.X
if enemy.X < maxCol.X
maxCol.X = enemy.X
end loop
if (maxCol.X == minCol.X) //one column
return maxCol.X
Ser := rand()%2 // 0-1
if (i == 0)
set Attack Direction = 1
else set Attack Direction 1
loop enemies
if ~~enemy.attackDirection == -1~~
if attackDirection == -1
set attackCol = maxX

GetColExtremes(maxX, maxX)

GetRandomDirection()

else set attackCol = maxX
set enemy.attackDir = attackDirection
attackingEnemies.push_back(enemy.y)

~~Sort Enemies Attack()~~
ChooseSetAttack()

if attackingEnemies.size() == 1
return
else
float maxY = 0.0f
~~if attackingEnemies.size() > 1 && attackingEnemies[1].y~~
~~Enemy* temp = attackingEnemies[0]~~
~~enemies[0] = enemies[1]~~
~~enemies[1] = temp~~
return

Not Needed ensure enemies created sorted.

SortEnemies

loop Enemies by size()
First enemy set isleader to true
All else set to false (default)

SetLeader()

update

if (isAttacking)
switch (attackState)
move:
while y < returnPosY + attack Radius
y += speed * time.
END while
set Attack State = Circle
break;
Circle:

attackingEnemies()

make Array of enemies to attack size of (n)
enemies[0] - state / attack state = move
enemies[1-n] - wait / attack state = wait
If (enemies[attackingEnemy].gameState = ATTACK
AND enemies.size() != attackingEnemy+1)
attackingEnemy ++
enemies[attackingEnemy].attackState = MOVE

AttackingEnemy = 0