Involved classes:

* Projectile: Mainly for utilising the piercing projectile's functions
* Player: Mainly for reduce the player’s health when the enemy hits them
* CartesianAndPolar: Converts between cartesian and polar coordinates to reorient after a knockback
* HitSystem: Increments the number of hits after enemy is damaged

**Enemy class**

// The following are set in Inspector:  
create int enemyID   
create float speed   
create float kbDuration   
create float kbMultiplier  
create float splitArcHalf   
create int splitEnemyCount   
create GameObject enemySplitup

create int health and set it to 1  
create float piercingCooldown and set it to 1  
create float pierceCounter and set it to 0  
create float speedStored

UpdateHealth procedure:  
parameters: int newHealth

set health to newHealth  
Look at cases for health:  
case 0:  
 change this sprite's colour to Clear  
case 1:  
 change this sprite's colour to Red  
case 2:  
 change this sprite's colour to Green  
case 3:  
 change this sprite's colour to Blue  
case 4:  
 change this sprite's colour to Black

CreateEnemySettings public procedure:  
parameters: int newHealth  
set speedStored to speed  
execute UpdateSkin(newHealth) function

Update procedure:

translate this gameObject by finalSpeed \* Time.deltaTime in the direction of the local y axis  
add Time.deltaTime to pierceCounter

OnTriggerEnter procedure:  
parameter: Collider other

if (other's tag == "Projectile):  
 in other’s Projectile component, execute DecreasePierce()  
 execute TakeDamage function  
else if (other's tag == "Piercing Proj" AND pierceCounter >= pierceCooldown):  
 in other’s Projectile component, execute DecreasePierce()  
 set pierceCounter to 0  
 execute TakeDamage function  
else if (other’s tag == “Player”)  
 in other’s Player component, execute ReduceHealth()  
 start Coroutine KillEnemy

TakeDamage procedure:

In Hit Display's HitSystem component, execute IncrementHit()  
if (enemyID > 0 && health > 1):  
 execute SplitEnemy()  
else if (enemyID = 0 && health > 1):  
 execute UpdateHealth (health - 1)  
 start Coroutine Knockback(kbDuration)  
else if (health <= 1):  
 start Coroutine KillEnemy

SplitEnemy procedure:

create float randomAngle and set it to 0  
create int splitArcSide and set it to 1

for (int n = 0; n < splitEnemyCount; n++):  
 if splitArcHalf = 1, set randomAngle to a random range between -90 and -splitArcHalf  
 if splitArcHalf = -1, set randomAngle to a random range between splitArcHalf and 90  
 multiply splitArcHalf by -1  
 create GameObject newEnemy and Instantiate it as enemySplitup at this gameObject's position  
 rotate newEnemy by randomAngle  
 in newEnemy's Enemy component, execute CreateEnemySettings(health-1)  
 in newEnemy's Enemy component, start the Coroutine Knockback(kbDuration)  
Start Coroutine KillEnemy

Knockback public coroutine:  
parameters: float duration, float speedMultiplier

create float array newPolarCoOrds

subtract speed by speedStored \* speedMultiplier  
wait ‘duration’ seconds  
add speedStored \* speedMultiplier to speed  
set newPolarCoOrds to the result of CartesianAndPolar’s function ConvertToPolar(this objecty's x position, this object's y position)  
set this object's rotation to newPolarCoOrds + 90 degrees

KillEnemy coroutine:  
set speed to 0  
execute UpdateHealth(0)  
Play child particle system  
Wait until particle system finishes playing  
Destroy this object