Involved classes:

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

**Enemy class**

create float baseSpeed and set it to 2  
create float typeSpeedReduction and set it to 0.2

create float piercingCooldown and set it to 0.7  
create float kbDuration and set it to 0.5  
create float splitArcHalf and set it to 30

create public gameObject enemyTriangle  
create public gameObject enemySquare

create int enemyID and set it to 1  
create int health and set it to 1  
create float finalSpeed and set it to 2  
create float finalSpeedStorage

create float currPiercingCooldown and set it to 0

UpdateSkin procedure:

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 newID, int newHealth

set enemyID to newID  
set finalSpeed to baseSpeed - typeSpeedReduction \* (newID - 1)  
set finalSpeedStorage to finalSpeed  
set health to newHealth  
execute UpdateSkin() function

Update procedure:

translate this gameObject by finalSpeed \* Time.deltaTime in the direction of the local y axis  
if (currPiercingCooldown < piercingCooldown):  
 add Time.deltaTime to currPiercingCooldown

OnCollisionEnter procedure:  
parameter: Collision other

if (other's tag == "Projectile):  
 in other’s Projectile component, execute DecreasePierce()  
 execute TakeDamage function  
else if (other's tag == "Piercing Proj" AND currPiercingCooldown >= piercingCooldown):  
 in other’s Projectile component, execute DecreasePierce()  
 set currPiercingCooldown to 0  
 execute TakeDamage function

TakeDamage procedure:

In Hit Display's HitSystem component, execute IncrementHit()  
if (enemyID > 1 && health > 1):  
 execute SplitEnemy()  
else if (enemyID = 1 && health > 1):  
 subtract health by 1  
 execute UpdateSkin()  
 start Coroutine Knockback  
else if (health <= 1):  
 start Coroutine KillEnemy

SplitEnemy procedure:

Create float randomAngle and set it to 0

Repeat twice:  
 Look at cases for enemyID - 1:  
 case 1:  
 create GameObject newEnemy and Instantiate it as enemyTriangle at this gameObject's position  
 case 2:  
 create GameObject newEnemy and Instantiate it as enemySquare at this gameObject's position  
 In newEnemy's Enemy component, execute CreateEnemySettings(enemyID -1, health-1)  
 while the absolute value of randomAngle is less than splitArcHalf/2:  
 set randomAngle to random range between -90 and +90  
 Rotate newEnemy by randomAngle  
 In newEnemy's Enemy component, start the Coroutine Knockback  
Start Coroutine KillEnemy

Knockback public coroutine:  
create float k and set it to 0

Create float kbSpeedMultiplier and set it to (1 + 1 / (kbDuration/2))  
create float array newPolarCoOrds

subtract finalSpeed by finalSpeedStorage \* kbSpeedMultiplier  
while (k <= kbDuration):  
 translate this gameObject by finalSpeed \* Time.deltaTime \* kbSpeedMultiplier in the direction of the local y axis  
 Add Time.deltaTime \* kbSpeedMultiplier to k  
 wait Time.deltaTime seconds  
add finalSpeed by finalSpeedStorage \* kbSpeedMultiplier  
set newPolarCoOrds to the result of CartesianAndPolar.ConvertToPolar(this objecty's x position, this object's y position)  
set this object's rotation to newPolarCoOrds + 90 degrees

KillEnemy coroutine:  
set health to 0  
set finalSpeed to 0  
execute UpdateSkin()  
Play child particle system  
Wait 0.9 seconds  
Destroy this object