**Involved classes:**

* Weapon - This class uses most of the weapon settings contained in this class
* Projectile - Manages the projectiles used by the player's weaponry

**Player Weapon class**

create float baseFiringCD and set it to 0.35  
create float weaponSwapCD and set it to 0.1

create GameObject shortbow  
create GameObject crossbow  
create GameObject launcher

create GameObject projectile

create int weaponChoice and set it to 0  
create float fireRateCounter and set it to 0  
create GameObject array weaponObjectList  
create Weapon array weaponList  
create Vector3 mousePos

Start procedure:

set weaponObjectList to [shortbow, crossbow, launcher]  
set weaponList to [shortbow's Weapon component, crossbow's Weapon component, launcher's Weapon component]  
Execute SwitchWeapon(0)

Update procedure:  
set mousePos to the mouse cursor's screen position  
Rotate this gameObject to face mousePos  
Add Time.deltaTime to fireRateCounter

If left mouse button is pressed AND fireRateCounter >= weaponList[weaponChoice]'s fireMultiplier \* baseFiringCD:  
 set fireRateCounter to 0  
 Execute FireWeapon()

If right mouse button is pressed AND fireRateCounter >= weaponSwapCD:  
 set fireRateCounter to 0  
 Execute SwitchWeapon(weaponChoice+1)

SwitchWeapon procedure:  
Parameter: int newWeapon

hide the GameObject weaponObjectList[weaponChoice]  
set weaponChoice to the mod between newWeapon and the length of weaponObjectList  
show the GameObject weaponObjectList[weaponChoice]

FireWeapon procedure:

if (weaponList[weaponChoice]'s isMultishot = true):  
 create float projDivision and set it to weaponList[weaponChoice]'s weaponArc/(projectileCount-1)  
 create float currAngle and set it to (this gameObject's rotation - weaponList[weaponChoice]’s weaponArc/2)  
 for (int i = 0; i < weaponList[weaponChoice]’s projectileCount; i++):  
 Execute CreateProjectile(currAngle)  
 add projDivision to currAngle  
else:  
 Execute CreateProjectile(this gameObject's rotation)

CreateProjectile procedure:  
Parameter: Vector3 spawnAngle

create GameObject newProjectile with prefab projectile at this gameObject's origin, rotated to spawnAngle  
in newProjectile's Projectile component, execute ProjectileSettings(weaponList[weaponChoice])