EMERALD INTEGRATION

Remember to check out this video for the settings in the inspector, once that is done, you can follow the steps above:

https://www.youtube.com/watch?v=miCpkYpyZ_E

-Add this code in the script **EmeraldAIPlayerDamage**:

```
void DamageGameKitController(int DamageAmount, GameObject attacker)
{
        applyDamage.checkToDamageGKCCharacterExternally (DamageAmount,
        gameObject, attacker);
}
```

-And in that same script, add this code inside **SendPlayerDamage** function:

```
GameObject targetObject = null;
if (Target != null){
        targetObject = Target.gameObject;
}

DamageGameKitController(DamageAmount, targetObject);
```

-Then, there are two ways for this (follow ONLY the first way, which is much better and only make the second way instead if for any reason, you don't want to use inheritance):

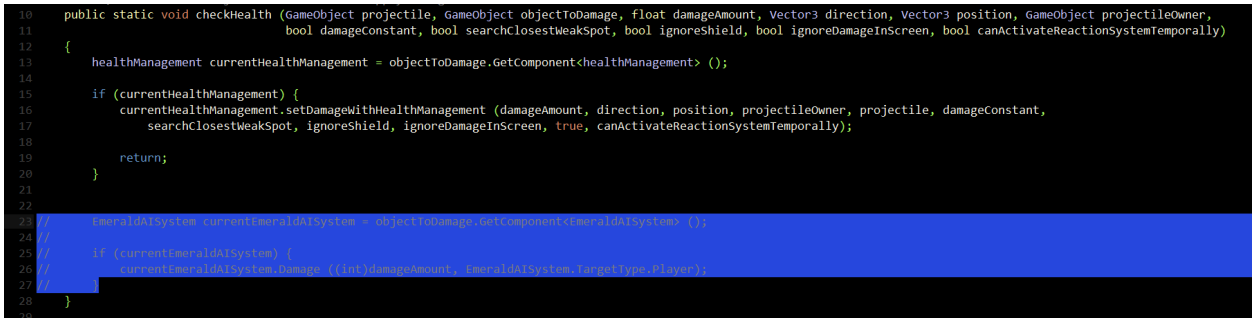
FIRST WAY: -Add the following code in **EmeraldAISystem**:

```
    public override void setDamageWithHealthManagement(float damageAmount,
Vector3 fromDirection, Vector3 damagePos, GameObject attacker, GameObject projectile,
bool damageConstant,
bool searchClosestWeakSpot, bool ignoreShield, bool
ignoreDamageInScreen, bool damageCanBeBlocked, bool
canActivateReactionSystemTemporally, int damageReactionID, int damageTypeID)
    {
        Damage((int)damageAmount, EmeraldAISystem.TargetType.Player);
    }
```

-And in that same script, replace **MonoBehaviour** word for **healthManagement** at the beginning of the script.

And that is all the additions needed, no need to use the second way unless you have a very specific reason for it.

SECOND WAY: -Open the script applyDamage.cs, and uncomment the code related to emerald in the last part of the first two functions of this component, called **checkHealth** and **checkCanBeDamaged**:

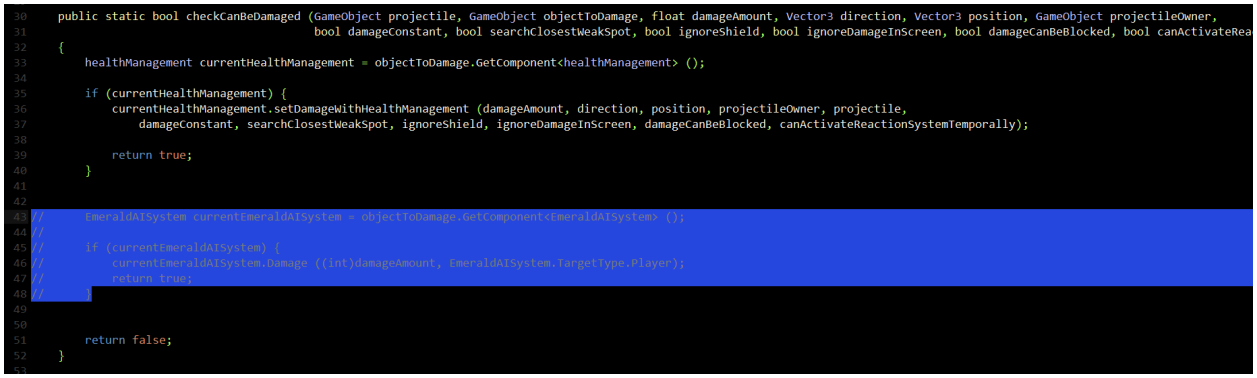


```
    public static void checkHealth (GameObject projectile, GameObject objectToDamage, float damageAmount, Vector3 direction, Vector3 position, GameObject projectileOwner,
                                    bool damageConstant, bool searchClosestWeakSpot, bool ignoreShield, bool ignoreDamageInScreen, bool canActivateReactionSystemTemporally)
    {
        healthManagement currentHealthManagement = objectToDamage.GetComponent<healthManagement> ();

        if (currentHealthManagement) {
            currentHealthManagement.setDamageWithHealthManagement (damageAmount, direction, position, projectileOwner, projectile, damageConstant,
                searchClosestWeakSpot, ignoreShield, ignoreDamageInScreen, true, canActivateReactionSystemTemporally);

            return;
        }


//      EmeraldAISystem currentEmeraldAISystem = objectToDamage.GetComponent<EmeraldAISystem> ();
//
//      if (currentEmeraldAISystem) {
//          currentEmeraldAISystem.Damage ((int)damageAmount, EmeraldAISystem.TargetType.Player);
//      }
    }
```



```
    public static bool checkCanBeDamaged (GameObject projectile, GameObject objectToDamage, float damageAmount, Vector3 direction, Vector3 position, GameObject projectileOwner,
                                          bool damageConstant, bool searchClosestWeakSpot, bool ignoreShield, bool ignoreDamageInScreen, bool damageCanBeBlocked, bool canActivateRea
    {
        healthManagement currentHealthManagement = objectToDamage.GetComponent<healthManagement> ();

        if (currentHealthManagement) {
            currentHealthManagement.setDamageWithHealthManagement (damageAmount, direction, position, projectileOwner, projectile,
                damageConstant, searchClosestWeakSpot, ignoreShield, ignoreDamageInScreen, damageCanBeBlocked, canActivateReactionSystemTemporally);

            return true;
        }


//      EmeraldAISystem currentEmeraldAISystem = objectToDamage.GetComponent<EmeraldAISystem> ();
//
//      if (currentEmeraldAISystem) {
//          currentEmeraldAISystem.Damage ((int)damageAmount, EmeraldAISystem.TargetType.Player);
//          return true;
//      }


        return false;
    }
```

-Add this code after all the Using lines in **applyDamage** script:

using EmeraldAI;