

FITXA D'ACTIVITAT		
CURS: 2022-23	DATA: 12/2022	GRUP: 1DAW/1DAM/1ASIX
PROFESSOR/A:	Javier Salvador Carmen Quintás	
MÒDUL/UF:	MP05/UF1-M16/UF2	
TÍTOL	Pr4: Refactorización de código	

## Descripción y Objetivos

El objetivo principal de esta práctica es aprender a refactorizar nuestro código, aplicando las nuevas técnicas y habilidades adquiridas, como los parámetros por referencia.

## Refactorización del Código del Videojuego

1. Usar las variables/parámetros por referencia para crear funciones globales donde creáis oportuno de vuestro código. **(5p)**

Para empezar con las variables como podemos observar he introducido una variable por referencia llamada eneHP y que es de tipo int.

```
void attackOption2(int& eneHP) {  
    cout << "[a] espada [b] magia [c] cuchillo\n";  
    cin >> attackOption;  
  
    if (attackOption == "a") {  
        cout << "Has hecho " << espadaDmg << " de danyo\n";  
        eneHP = eneHP - espadaDmg;  
    }  
  
    if (attackOption == "b") {  
        cout << "Has hecho " << magiaDmg << " de danyo\n";  
        eneHP = eneHP - magiaDmg;  
    }  
  
    if (attackOption == "c") {  
        cout << "Has hecho " << cuchilloDmg << " de danyo\n";  
        eneHP = eneHP - cuchilloDmg;  
    }  
}
```

Oriol de Caldas

En el siguiente void podemos ver como hemos asignado la variable anterior en su void correspondiente. Lo hemos introducido dos veces ya que hay dos enemigos por lo tanto dos opciones de ataque.

```
void heroAttack() {  
    attackOption = "z";  
    if (enemyOption == "1") {  
        while (attackOption != "a" && attackOption != "b" && attackOption != "c") {  
            cout << "Como quieres atacar a Sauron\n";  
            attackOption2(& eneHP: enemyHP );  
        }  
    }  
  
    if (enemyOption == "2") {  
        while (attackOption != "a" && attackOption != "b" && attackOption != "c") {  
            cout << "Como quieres atacar a Voldemort\n";  
            attackOption2(& eneHP: enemyHP2);  
        }  
    }  
}
```

Oriol de Caldas

FITXA D'ACTIVITAT		
CURS: 2022-23	DATA: 12/2022	GRUP: 1DAW/1DAM/1ASIX
PROFESSOR/A:	Javier Salvador Carmen Quintás	
MÒDUL/UF:	MP05/UF1-M16/UF2	
TÍTOL		
ACTIVITAT:	Pr4: Refactorización de código	

A continuación, he introducido las variables eName de tipo string, eneHP de tipo int y eneAl de tipo bool.

```
void checkEnemyStatus(string& eName, int& eneHP, bool& eneAl) {
    if (eneHP > 0) {
        cout << "La vida de " << eName << " es " << eneHP << "\n";
    }

    else if (eneHP <= 0) {
        cout << "Esta muerto\n";
        eneAl = false;
    }
}
```

Oriol de Caldas

En este caso hemos usado variable por referencia en un bool y hemos añadido las variables eneDmg de tipo int y eName de tipo string.

```
bool enemydmg(int& eneDmg, string& eName) {
    eneDmg = 1 + (rand() % 25);
    heroHP = heroHP - eneDmg;
    cout << "El enemigo " << eName << " te ha atacado con " << eneDmg << "\n";
}
```

Oriol de Caldas

En el siguiente bool podemos ver como hemos asignado la variable anterior en el enemydmg. Lo hemos introducido dos veces ya que hay dos enemigos por lo tanto dos opciones de ataque.

```
bool enemyAttack() {
    if (enemyIsAlive && heroIsAlive) {
        enemydmg(& eneDmg: enemyDmg, & eName: enemyName);
        enemydmg(& eneDmg: enemyDmg2, & eName: enemyName2);
    }

    if (heroHP > 0) {
        cout << "Te quedan " << heroHP << " puntos de vida\n";
        return true;
    }

    if (heroHP <= 0) {
        cout << "Has muerto\n";
        return false;
    }
}
```

Oriol de Caldas

Por último, como podemos observar hemos asignado las variables por referencia que hemos creado en el void de checkEnemyStatus.

```
int main() {
    inicio();

    while ((enemyIsAlive || enemyIsAlive2) && heroIsAlive) {
        enemySelected = enemySelect();
        heroAttack();

        if (enemySelected == 1) {
            checkEnemyStatus(& eName: enemyName, & eneHP: enemyHP, & eneAl: enemyIsAlive);
        }

        if (enemySelected == 2) {
            checkEnemyStatus(& eName: enemyName2, & eneHP: enemyHP2, & eneAl: enemyIsAlive2);
        }

        heroIsAlive = enemyAttack();
    }
}
```

Oriol de Caldas

FITXA D'ACTIVITAT		
CURS: 2022-23	DATA: 12/2022	GRUP: 1DAW/1DAM/1ASIX
PROFESSOR/A:	Javier Salvador Carmen Quintás	
MÒDUL/UF:	MP05/UF1-M16/UF2	
TÍTOL		
ACTIVITAT:	Pr4: Refactorización de código	

2. Modificar el código para mejorarlo, usando lo aprendido en M3 y M5/M16 desde el desarrollo de la PR3 (3p)

He modificado el código metiendo este void para que las variables referenciales me pudieran funcionar correctamente.

```
void attackOption2(int& eneHP) {
    cout << "[a] espada [b] magia [c] cuchillo\n";
    cin >> attackOption;

    if (attackOption == "a") {
        cout << "Has hecho " << espadaDmg << " de danyo\n";
        eneHP = eneHP - espadaDmg;
    }

    if (attackOption == "b") {
        cout << "Has hecho " << magiaDmg << " de danyo\n";
        eneHP = eneHP - magiaDmg;
    }

    if (attackOption == "c") {
        cout << "Has hecho " << cuchilloDmg << " de danyo\n";
        eneHP = eneHP - cuchilloDmg;
    }
}
```

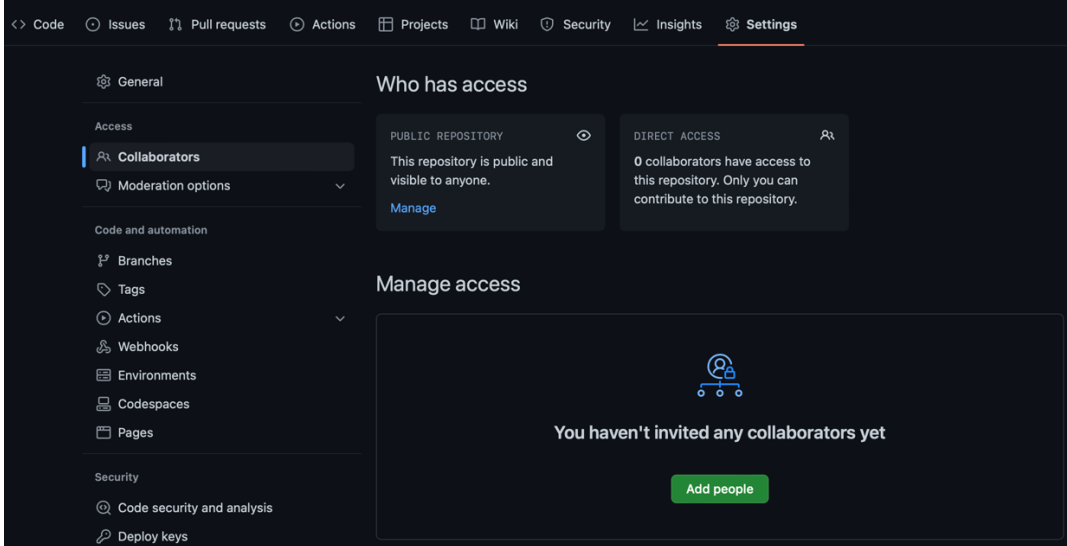
Oriol de Caldas

3. Subir a GIT el juego, teniendo en cuenta los parámetros del apartado de **Entrega de la Actividad PR4. (2p)**

## Entrega de la Actividad Pr4:

- Al Moodle se subirá un fichero con únicamente una URL del proyecto de GIT
- La evaluación se completará con una **comprobación in situ**.
- En el proyecto de GIT, dentro de una carpeta que se llame PR4, tendrá que haber:
  - El código del juego (Sin los archivos de DEBUG)
  - Un documento PDF explicando todos los cambios realizados a nivel de código/funciones
- En caso de que el proyecto GIT sea privado, ir a SETTINGS→COLLABORATORS y añadir al usuario @JAVIZAWA

FITXA D'ACTIVITAT		
CURS: 2022-23	DATA: 12/2022	GRUP: 1DAW/1DAM/1ASIX
PROFESSOR/A:	Javier Salvador Carmen Quintás	
MÒDUL/UF:	MP05/UF1-M16/UF2	
TÍTOL ACTIVITAT:	Pr4: Refactorización de código	



The screenshot shows the GitHub repository settings page for a repository. The left sidebar contains navigation links: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings (highlighted). Under the 'Settings' tab, the 'General' section is active, showing 'Access' options: Collaborators (selected), Moderation options, and Code and automation (Branches, Tags, Actions, Webhooks, Environments, Codespaces, Pages). The 'Who has access' section shows 'PUBLIC REPOSITORY' (This repository is public and visible to anyone. Manage) and 'DIRECT ACCESS' (0 collaborators have access to this repository. Only you can contribute to this repository.). The 'Manage access' section shows a message: 'You haven't invited any collaborators yet' with an 'Add people' button.