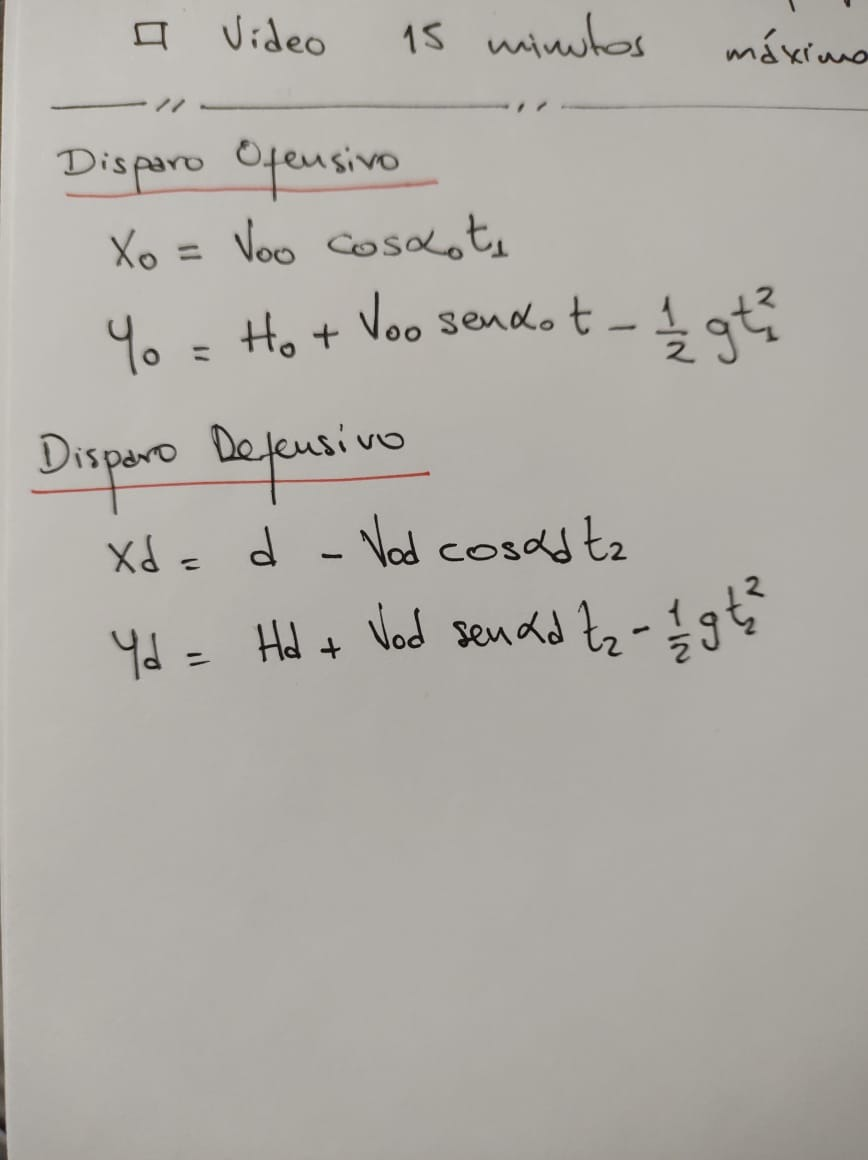


Siempre se sabe en el frente ofensivo cuando disparo la defensa y con qué parámetros, esta información llega 1 segundo después.



**Definición de clases**

|  |
| --- |
| CAÑON |
| posicionX  posicionY |
| disparar(tipoBala, Vo, angulo)  analizarInfo() |

|  |
| --- |
| BALA |
| velocidaInicial  angulo  posicionX  posicionY  rangoDetonacion |
| sensarCercania() |

|  |
| --- |
| ESCENARIO |
| cañonOfensivo  cañonDefensivo  distancia  tiempo |
| simularDisparo()  simularDefensivo(bool daño)  simularOfensivoEfectivo() |

Se utilizará una clase escenario que contenga los cañones.

Cada cañón tendrá una posición en X y una posición en Y; y podrá tener las acciones de disparar y de analizar información.

Las balas o disparos tendrán una velocidad Inicial, ángulo inicial, posición en X y posición en Y, además un atributo de rango de detonación. Las balas tendrán la acción de sensar cercanía con un objeto.

Para saber si un disparo ofensivo hará daño, se simulan las formulas del disparo parabólico variando el tiempo y en cada iteración midiendo la distancia de la bala con el cañón defensivo. Las iteraciones se deben suspender cuando: la distancia recorrida en X sea mayor a (Distancia entre cañones + rango de daño), o la altura de la bala sea menor o igual a cero.

Lo mismo se haría para el disparo defensivo, pero cuanto tiene en cuenta el disparo ofensivo el tiempo tiene un retraso de 2 segundos. En cada iteración en el tiempo se analizará la distancia a la bala ofensiva. Y si logra evitar el impacto, la iteración debe continuar para analizar si hiciera daño al cañón ofensivo.

Para evitar que la bala ofensiva sea interrumpida, se utiliza otro tipo de bala que tendrá otro rango de efectividad y es importante saber acá que el tiempo tendría un retraso de 3 segundos después del primer disparo ofensivo. Para así garantizar que todos los análisis sean en el mismo instante de tiempo.