

**INFORME DEL TRABAJO FINAL**

**(ABET)**

CURSO DE ALGORITMOS – CC215

Carrera de Ciencias de la Computación

Sección: SW21

Alumnos:

Daniella Alexandra Crysti Vargas Saldaña U202219211

Gianpaul Alonso Bernal Saravia U20221A634

Joaquín Sebastian Ruiz Ramírez U20201F678

Noviembre 2022

**CONTENIDO**

1. Introducción
2. Objetivo del estudiante
3. Diseño del diagrama de clases
4. Plan de actividades
5. Diseño del producto y funcionalidad adicional
6. Aporte
7. Conclusiones
8. Anexos
9. Bibliografía
10. **INTRODUCCIÓN**

El propósito del informe presente es demostrar la aplicación de los conocimientos que hemos adquirido en torno a Programación Orientada a Objetos (POO): Conceptos de cIases, objetos, reIaciones de agregación, reIaciones de herencia/poIimorfismo y eI uso de componentes visuales. Para ello, se ha seguido el enunciado asignado en clase durante el curso “Algoritmos”. En síntesis, se ha propuesto a los estudiantes crear un videojuego cuya temática central se vea desarrollada en torno a uno de los 11 objetivos de la Agenda 2030 para el Desarrollo Sostenible, propuesta por la Organización Panamericana de Salud (OPS). En el caso de nuestro grupo, el objetivo y la meta de dicho objetivo que se han escogido fueron:

**Objetivo 11:** *Desigualdades e inequidades en torno a la salud*

Reducir las desigualdades e inequidades en torno a la salud por medio de enfoques intersectoriales, multisectoriales, regionales y subregionales de los determinantes sociales y ambientales de la salud.

**Meta N°3:**

Reducir significativamente las desigualdades relacionadas con la calidad del agua y el saneamiento al avanzar con los sectores responsables en el acceso a los servicios de agua y saneamiento para su gestión segura (metas 6.1 y 6.2 de los ODS).

Sobre el objetivo, se puede decir que su propósito principal es asegurar la llegada de los servicios de salud esenciales a toda la población, reduciendo en su totalidad la marginación que sufren muchos sectores olvidados. Para esto, el objetivo busca recalcar ciertas capacidades que deben de ser desarrolladas por los gobiernos involucrados para el cumplimiento de este objetivo. Entre estas, por ejemplo, están el fortalecimiento de la capacidad de los países para medir, monitorear y analizar sistemáticamente las desigualdades de salud.

A través de la meta establecida, se ha podido enfocar la problemática que presenta el objetivo de manera más ordenada. Para dicho caso, lo que se busca es facilitar y extender el acceso a agua potable para los sectores de la población que más lo requieran. Según datos de la INEI (2020), en el 2019, en el Perú, alrededor de un 9,2% de la población carecía de acceso a agua a través de una red pública; al mismo tiempo, un 25,1% carecía de acceso a una red pública de alcantarillado. Además, un dato a recalcar que estas cifras no muestran por completo, es el hecho de que el sector de la población peruana que más afectado se ve por esta problemática, es el rural, pues es donde se encuentran los porcentajes más altos de falta de acceso a agua.

En vista de ello, se seleccionaron el objetivo y la meta previamente mencionados.

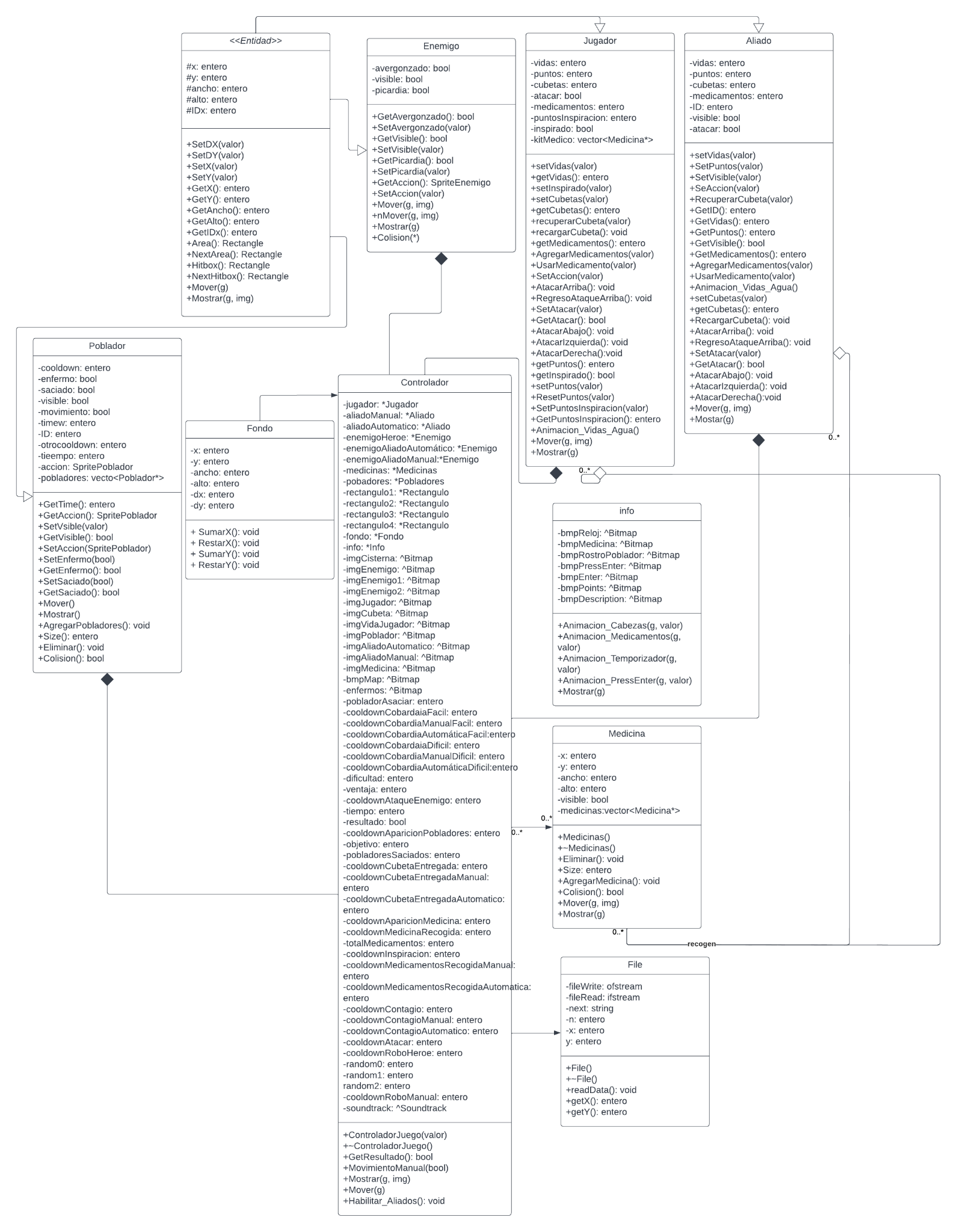
1. **OBJETIVO DEL ESTUDIANTE**

Describa cómo ha logrado el objetivo de acuerdo al *student outcome* de su carrera.

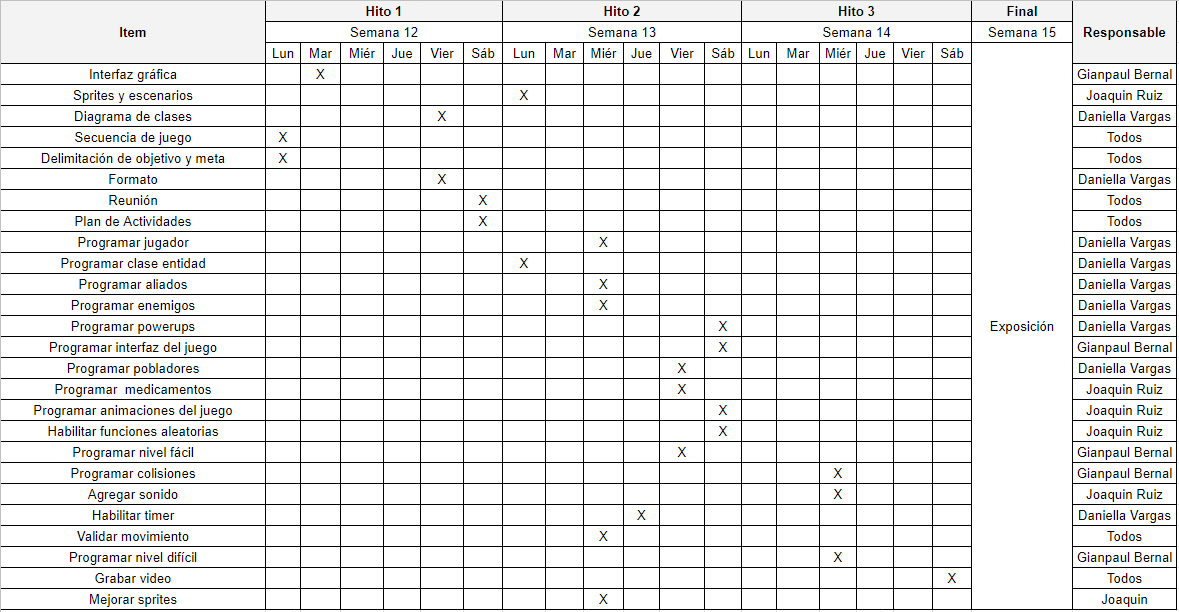
*“La capacidad de diseñar, implementar y evaluar una solución basada en computación para cumplir con el conjunto de requerimientos en el contexto de sistemas de información”*

A lo largo del curso de “Algoritmos”, se espera que los estudiantes logren adquirir habilidades específicas al finalizar el curso. Estas habilidades se basan en criterios que nuestro grupo ha logrado cumplir de la siguiente manera.

* ***Diseña una solución basada en computación para cumplir con el conjunto de requerimientos en el contexto de sistemas de información:***
  + Se elaboró un desempeño de calidad satisfactoria. Para la elaboración de este desempeño se diseñaron componentes detallados que, al ser criticados dentro de nuestras profesiones, cumplían con nuestros estándares y principios.
* ***Implementa una solución basada en computación para cumplir con el conjunto de requerimientos en el contexto de sistemas de información:***
  + Los estudiantes desarrollaron un programa que cumplía con las restricciones y requerimientos establecidos en el enunciado.
* ***Evalúa una solución basada en computación para cumplir con el conjunto de requerimientos en el contexto de sistemas de información:***
  + El plan del proyecto refleja por completo los procesos y metodologías aplicadas en el desempeño final. Además, se adoptaron procesos necesarios para el cumplimiento de requisitos y características propias del proyecto.

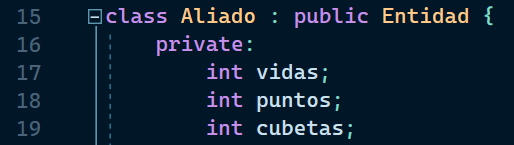
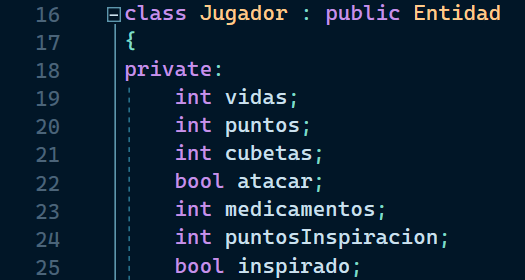
1. **DISEÑO DEL DIAGRAMA DE CLASES**
2. **PLAN DE ACTIVIDADES**

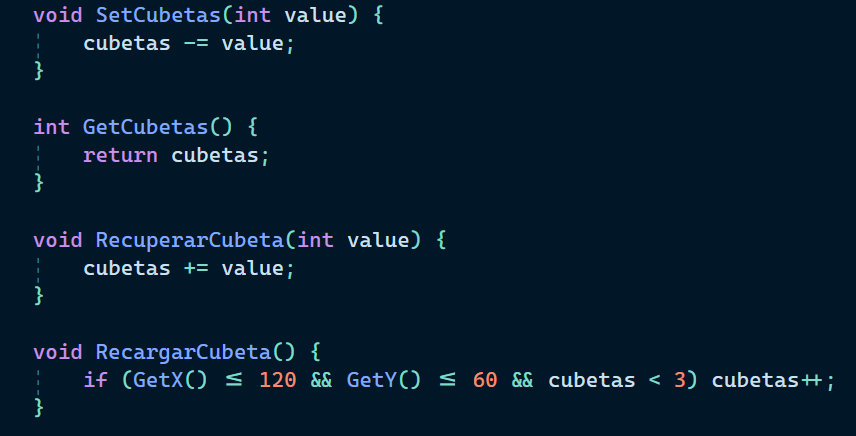
Descripción detallada de las actividades a realizar para la construcción del videojuego.



1. **DISEÑO DEL PRODUCTO**
2. Sistema de entrega de agua (Encapsulación, Abstracción, Herencia, Polimorfismo)

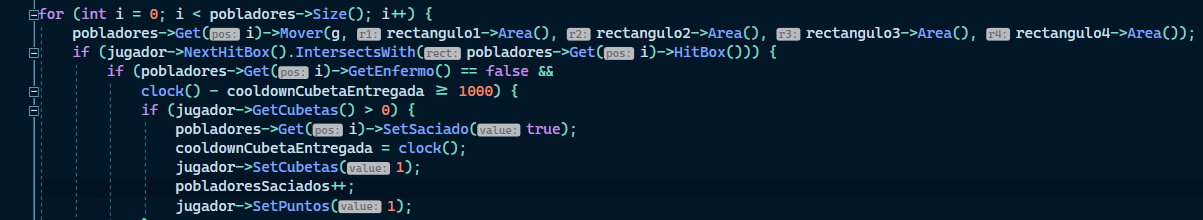
Frente al objetivo de cubrir el objetivo de saneamiento y acceso al agua potable, el sistema de entrega de agua es uno de los más importantes de nuestro juego. Aquel comienza con los atributos privados de las clases Aliado y Jugador, las cuales pueden ser accedidas sólo mediante métodos de set y get (Encapsulación).

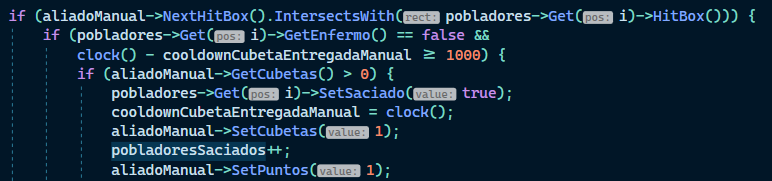


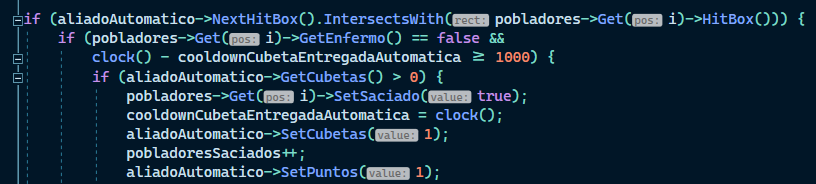


Tenemos a las cubetas, las cuales incorporamos aplicando el principio de Abstracción siendo un atributo fundamental. Este es un dato de tipo entero que nos da a conocer cuántas cubetas está cargando cada personaje. El rango de cubetas que pueden cargar se sitúa entre 0<=N<=3. El constructor de cada de estas clases instancia los valores de cubetas en 0 por predeterminado.

Los aliados y el jugador principal entregan estas cubetas a los pobladores al colisionar con estos. Ellos solo pueden entregarles cubetas si el poblador a entregar no se encuentra enfermo (un poblador se enferma pasados los 20 segundos sin haber accedido a agua potable) Las estructuras de selección simple utilizadas evitan que se entreguen estas cubetas a las enfermos si es que no se cuentan con medicamentos para sanarlos.



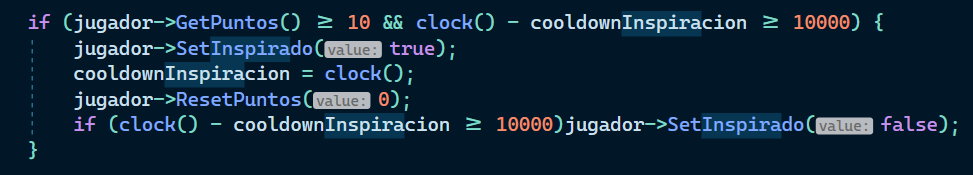




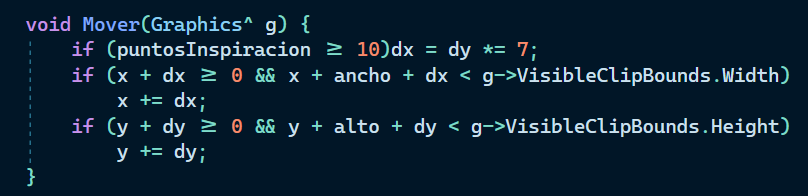
De tal modo que no se entreguen múltiples cubetas al instante, se implementó un enfriamiento (*cooldown*) de 1 segundo. Esto significa que tiene que pasar más de 1 segundo para que el jugador o aliado entreguen otra cubeta. Lo mencionado, es tiempo suficiente para que el poblador pueda dirigirse a casa tras haber saciado y no tomar agua de más.

El *cooldown* emplea una metodología simple: clock() de la librería time.h, recoge los milisegundos pasados desde la ejecución del programa, este timer es restado - 1 segundo en milisegundos, el enfriamiento de la entrega de cubetas. Cada aliado y jugador requieren de un *cooldown* individual, pues no entregan cubetas al mismo tiempo, son independientes.

Por otro lado, el sistema de entrega de cubetas está conectado con el sistema de obtención de puntos. Cada vez que un aliado o jugador entrega una cubeta a un poblador no saciado, se actualiza el contador de puntos. Cabe resaltar, que cada vez que se alcanza un hito de 10 puntos, el jugador principal obtiene el *power up* de *inspiración* por 10 segundos, lo cual lo hace ir 7 veces más rápido de su velocidad normal.

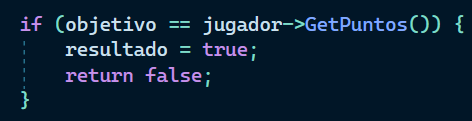


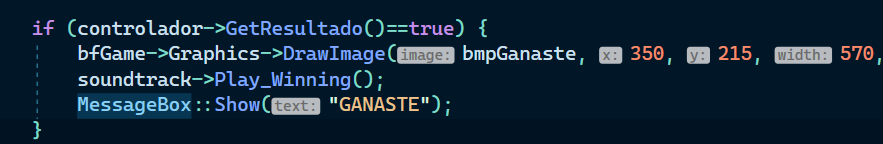
Este comportamiento distinto frente al de los aliados o pobladores, quienes como el jugador, heredan sus métodos de movimiento desde la clase base o Entidad, implica la presencia de Polimorfismo.



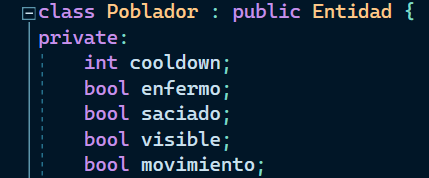
Adicionalmente, cuando el equipo llegue a los 20 puntos o pobladores saciados, automáticamente vencen el reto de saneamiento y ganan la partida.



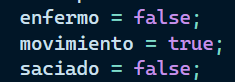




Acerca de la conexión con los pobladores. Estos, también presentan atributos fundamentales, como el estado de enfermedad, saciedad o visibilidad. Todos estos son datos de tipo booleano, pues solo pueden albergar 2 valores: verdadero o falso.



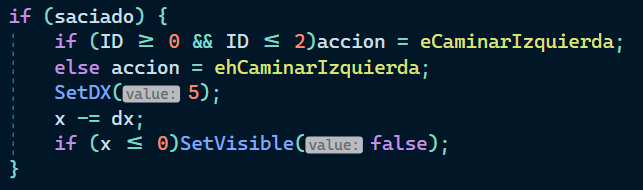
Por predeterminado, el constructor instancia lo valores mencionados (a excepción de visibilidad) en estado inactivo:



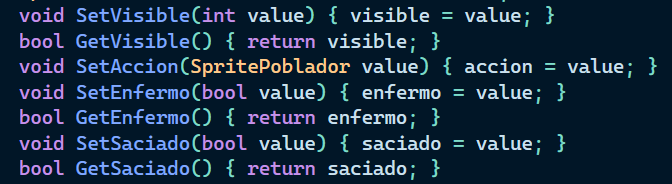
Es crucial que visibilidad no sea iniciada con un valor negativo, pues cuando un poblador carece de visibilidad, es eliminado desde la clase controladora:



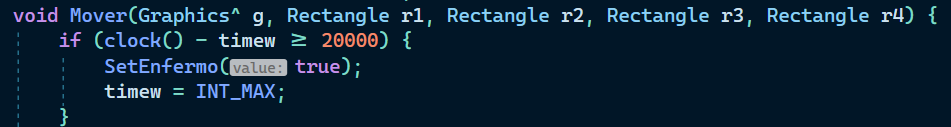
Un poblador solo puede dejar de ser visible si es que ha sido saciado, pues al suceder esto su movimiento cambia y se desplaza hacia el extremo izquierdo del mapa, para irse a casa, contento de haber recibido agua.



Los atributos esenciales encapsulados privadamente de los pobladores que se dieron a conocer, se comunican con otros métodos gracias a los setters y getters:

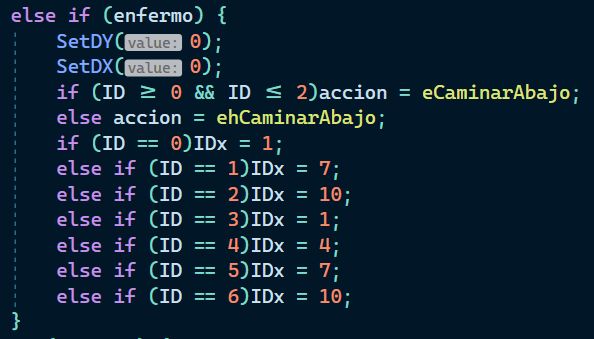


Los pobladores se desconectan del sistema de entrega de agua al enfermarse, tras pasar 20 segundos sin haber accedido a agua potable de calidad:

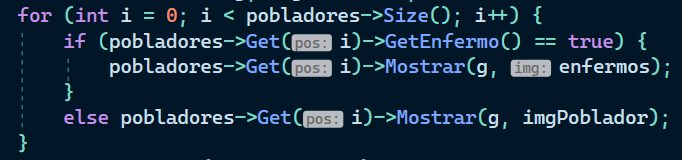


clock() recoge el tiempo pasado tras la ejecución del programa en milisegundos, mientras timew es una variable que utilizamos para controlar el suceso del evento. Una vez enfermos, esta variable almacena un valor máximo, para evitar que los pobladores se vuelvan a enfermar por ningún motivo, minimizando la ocurrencia de bugs.

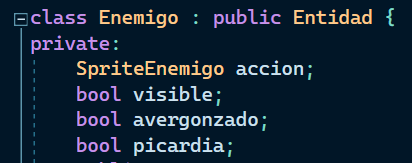
Es simple para los jugadores conocer si un poblador se encuentra enfermo y evitar ser contagiados y perder salud por esto. En primer lugar, un poblador enfermo se dejará de mover y en segundo, su sprite cambiará por uno con la piel pálida dada su debilidad.



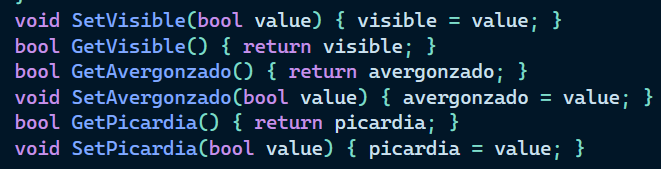
Los pobladores sanos y los pobladores enfermos, muestran sprites diferentes.



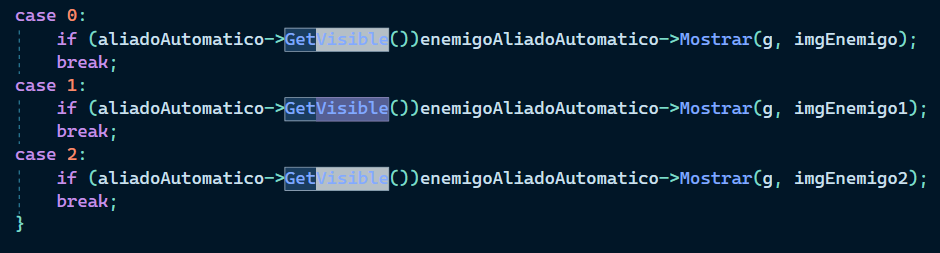
Ahora, podemos hablar de los enemigos, quiénes impiden que el equipo cumpla con su misión. Ellos también se encuentran íntimamente vinculados con el sistema de entrega de agua. Enemigo también es hijo de la clase Padre Entidad y contiene 3 características que se manejan de manera notoriamente distinta a otros personajes del juego: vergüenza, picardía y visibilidad.



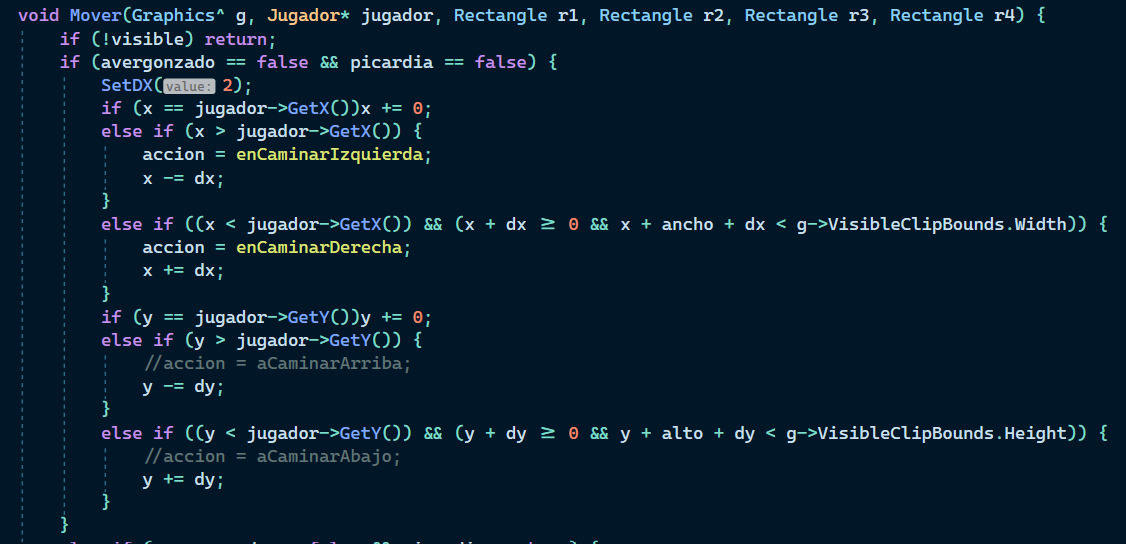
De manera similar, se comunican con otras funcionalidades del juego a través de los métodos de acceso:

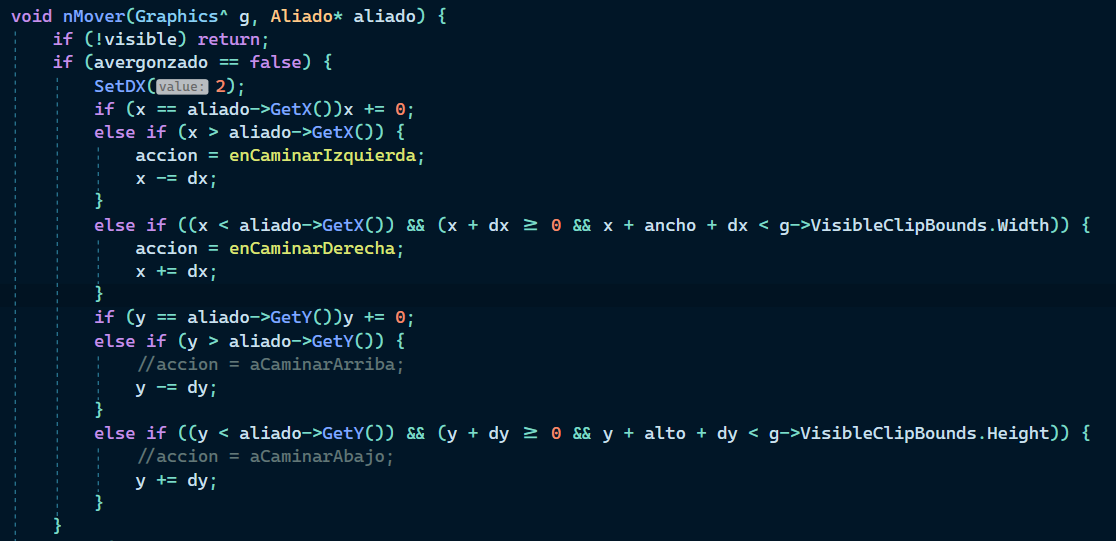


Hay 1 enemigo por cada aliado o jugador:



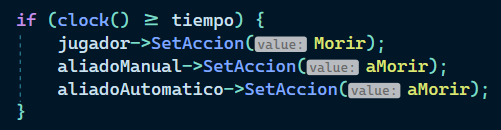
Los enemigos, tienen una movilidad peculiar que los caracteriza por perseguir a los miembros del equipo:

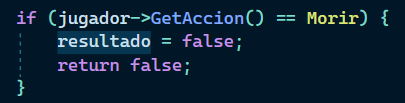


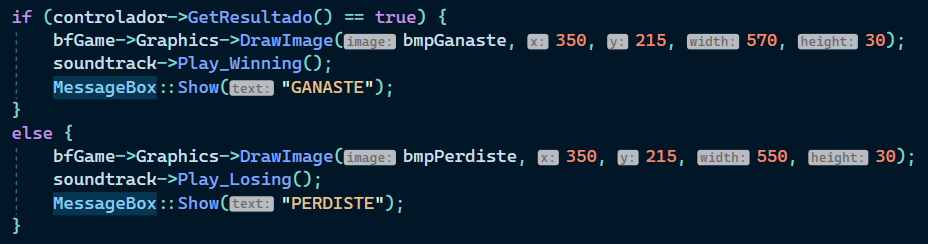


Ellos, reciben el parámetro tipo puntero de los objetos tipo Aliado o Jugador, para conseguir sus coordenadas en el formulario y poder manipular su propio movimiento hasta alcanzarlos. Se acercarán tanto como puedan para robarles las cubetas a los jugadores y hacerles perder tiempo. Si es que pasa 1 minuto y medio y el equipo todavía no ha cumplido con su objetivo, automáticamente pierden.

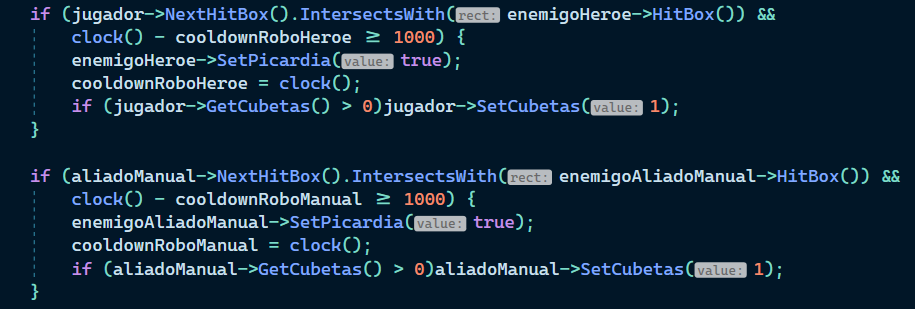








Este robo de cubetas se da a través de colisiones, manejamos las colisiones con Hitboxes más exactas para hacer el juego más justo.



Así como se aplica un enfriamiento para la entrega de cubetas, también se aplica uno para el robo de estas, para que no se roben múltiples cubetas al instante. Además, se valida que el número de cubetas sea mayor a 0 antes de que el enemigo escape, pues sino obtendremos un número irracional (negativo) de cubetas.

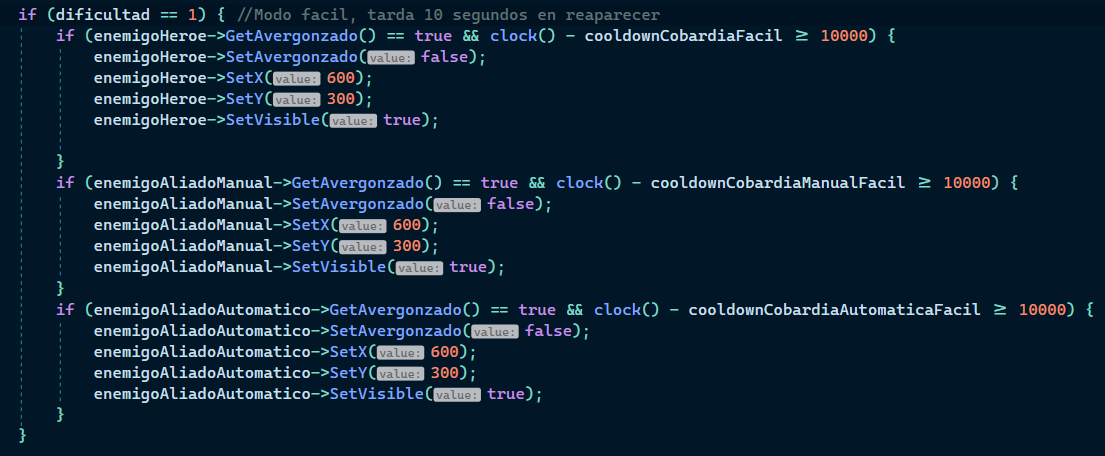
Picardía, es el estado en el que entran los enemigos que recientemente han robado una cubeta, su movimiento cambia a uno relajado, que se mueve libremente por el mapa sin limitaciones

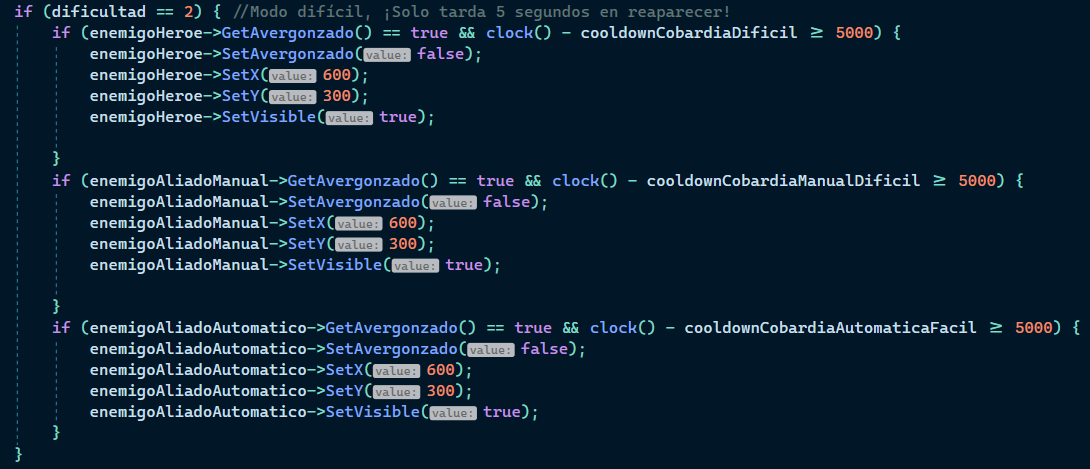


El enemigo sale de este estado cuando un aliado o jugador ejerce defensa propia sobre él para recuperar la cubeta. Esto resulta en la activación del estado de vergüenza del enemigo y su posterior huída, para aparecer nuevamente dentro de 10 segundos en el modo fácil y 5 en el modo difícil. Al huir de los ojos de los usuarios, se vuelve invisible.







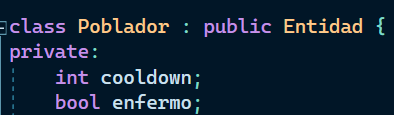


Como se puede observar, el aliado con movimiento manual a través del teclado, como el jugador, tienen estados de ataque que se activan con teclas adicionales (P y Q respectivamente). Sin embargo, el aliado con movimiento automático no requiere de estas al responder al instante.

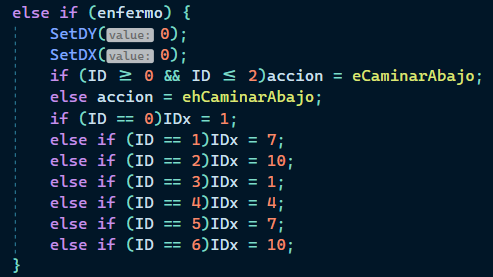
En esta luz, los enemigos impiden que el sistema de entrega de agua funcione correctamente, los pobladores son parte de este al ser los receptores y beneficiarios de aquel, mientras que el jugador es el responsable principal de su cumplimiento y eficiencia, con el apoyo de sus aliados. Se observó cómo el sistema de entrega aplica los principios de Herencia, Polimorfismo, Abstracción y Encapsulación de la POO.

1. **Curación de los enfermos**

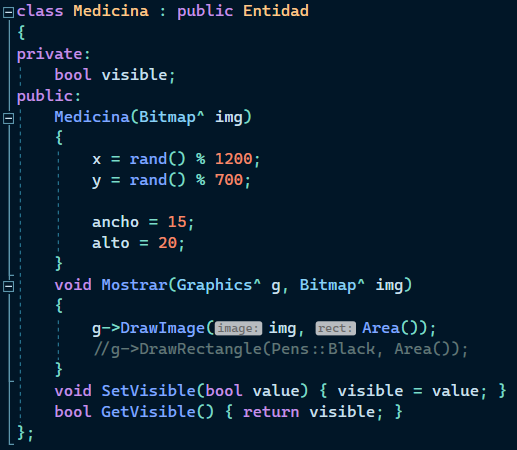
La curación de enfermos representa otra mecánica fundamental en el videojuego, dado que permite que el equipo jugador pueda llegar a más pobladores, sin perder por ausencia de salud (perder las 3 vidas otorgadas de manera automática al inicio del juego). En la funcionalidad anterior, hablamos sobre el estado de enfermedad booleano en la clase Poblador:



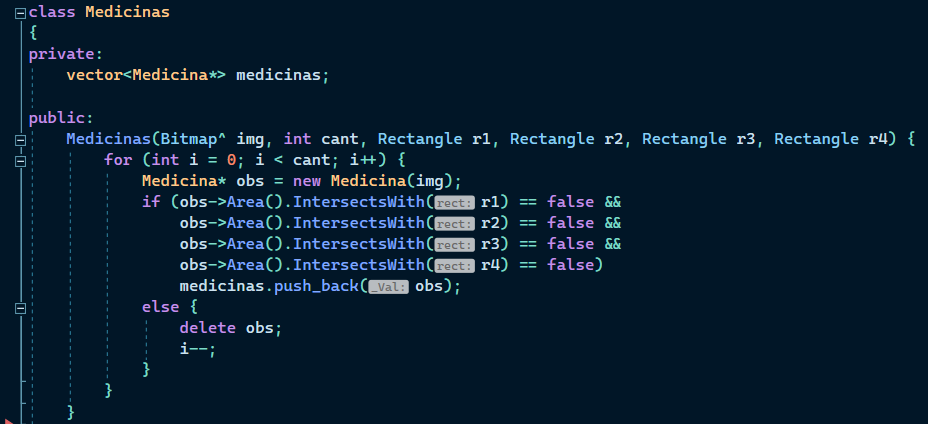
También, se mencionó acerca de las repercusiones que tiene esta condición en la movilidad de los pobladores e implementa el polimorfismo, que hace notorio su estado.



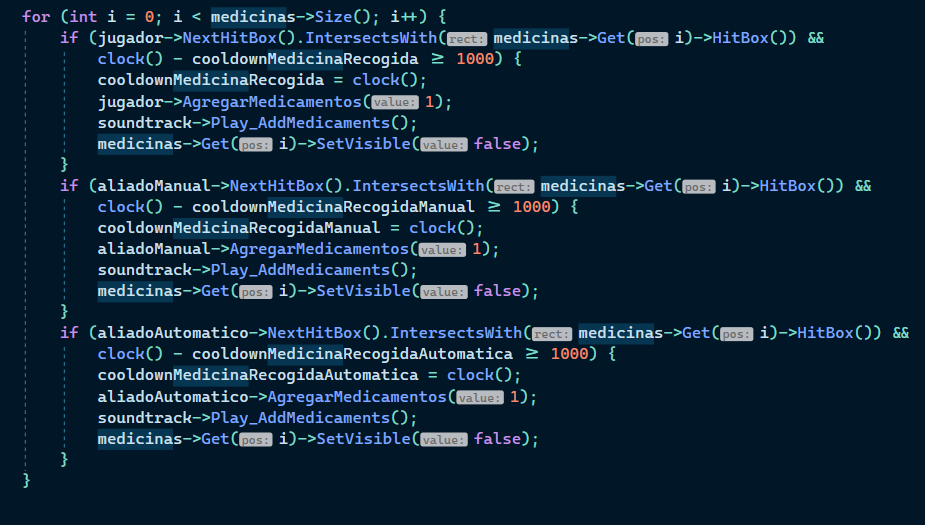
Para combatir esta enfermedad, se creó una nueva clase, la clase Medicina, que hereda de Entidad y contiene métodos y atributos encapsulados tanto de manera privada como pública.



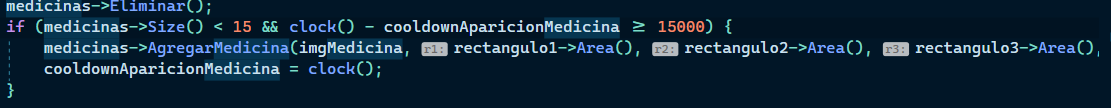
La medicina está representada por un sprite auto explicativo en el juego. Además, tiene su propia clase controladora que se asegura de que esta no haga *spawn* en una zona no accesible o no válida del escenario del formulario.



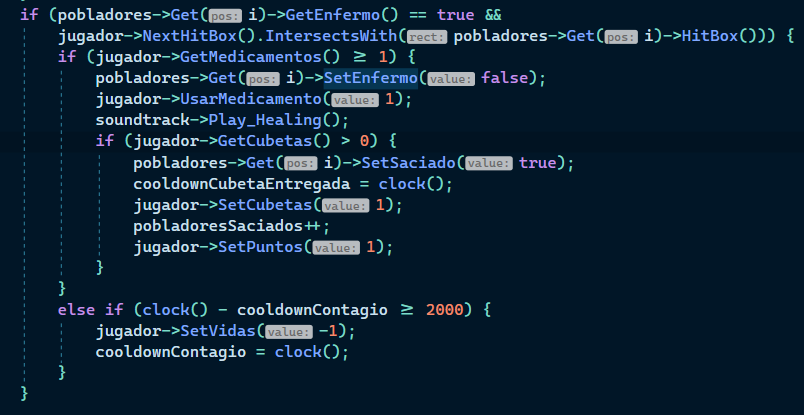
Asimismo, comprenden su propio sistema de recojo en la Superclase, a modo de colisión con el jugador o aliados, para que estos puedan guardar los medicamentos en sus kits médicos y curar a los pobladores donde quiera que vayan, evitando ser contagiados.

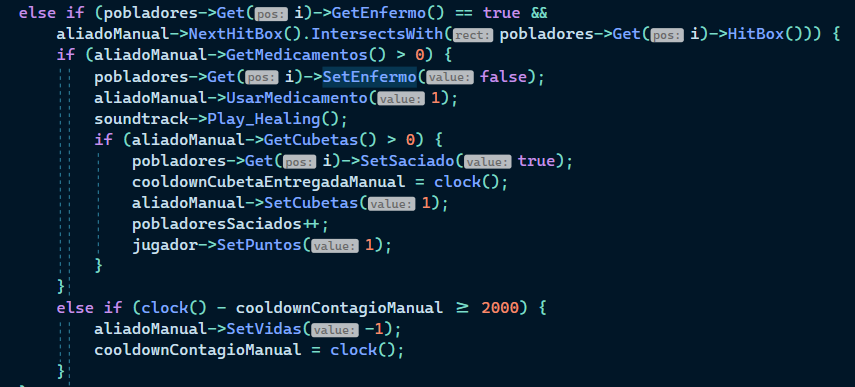


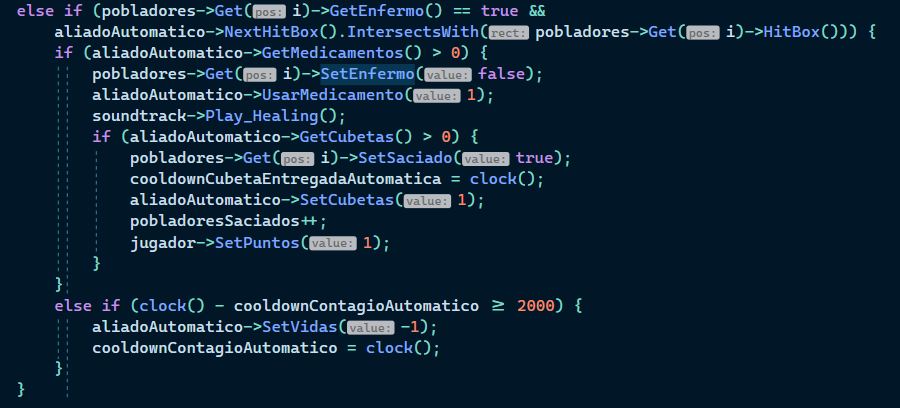
Además, la medicina también tiene un enfriamiento de aparición de 15 segundos, y su método para ser agregada recibe como parámetros los rectángulos validados como zonas no accesibles del mapa:



Finalmente, se aprecia cómo funciona el cooldown de contagio para perder salud y la entrega de medicina para curar a los pobladores:



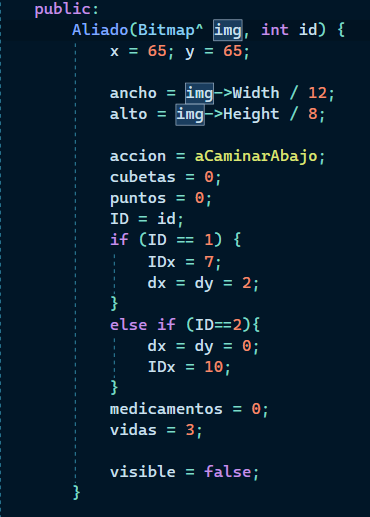




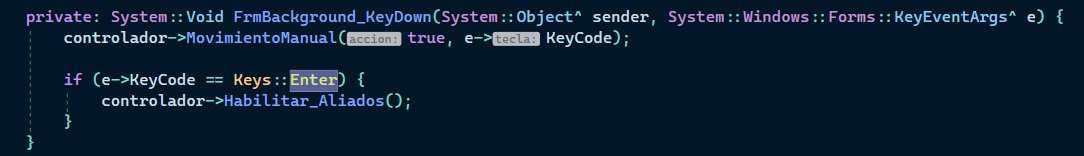
1. Movimiento del aliado automático

Como última funcionalidad principal, tenemos al movimiento del aliado automático, lo cual lo hace casi ver como un personaje inteligente independiente de los controles del usuario jugador.

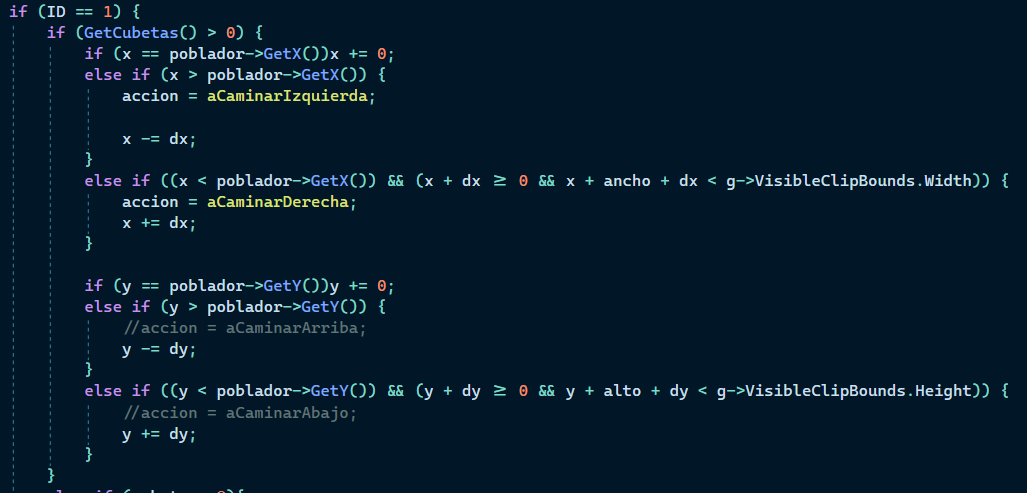
El aliado manual, se identifica con el ID N°1



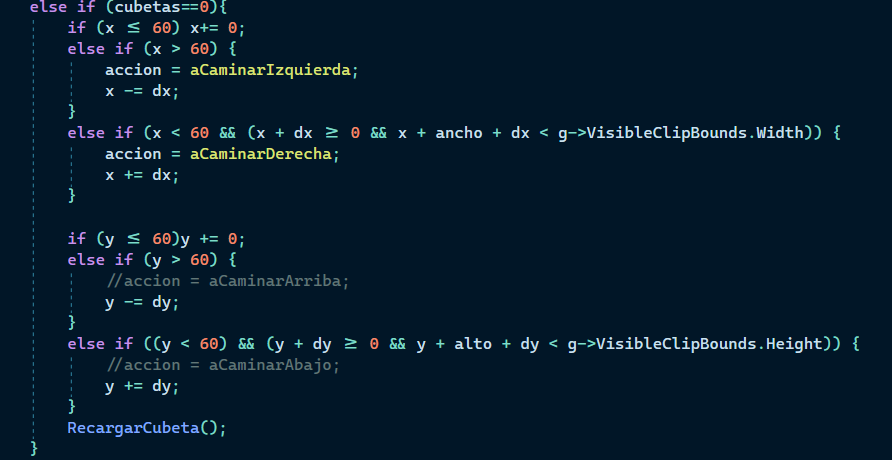
Además, por predeterminado no es visible, dado que los aliados sólo son llamados a la acción cuando el usuario principal presiona Enter en el teclado.



El movimiento de este aliado es una clara manifestación del polimorfismo, dada su diversidad de comportamientos frente a sus clases hermanas:



Si es que el aliado cuenta con más de una cubeta, buscará a un poblador en el vector de pobladores y obtendrá sus coordenadas para alcanzarlo.



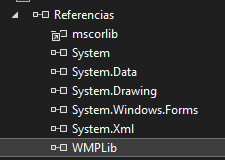
Sin embargo, si es que no cuenta con las cubetas necesarias para cumplir la actividad, regresará por su cuenta a recargar las cubetas.

1. **APORTE**

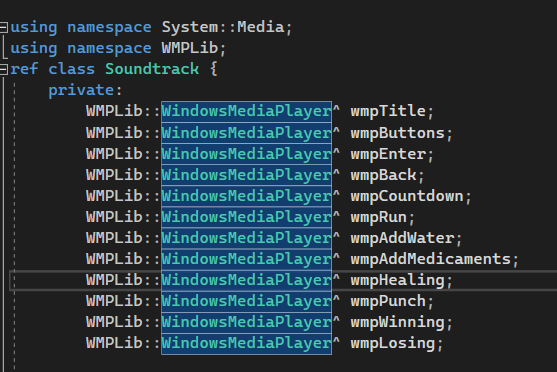
El desarrollo del videojuego como trabajo final, incentivó nuestra capacidad de pensamiento crítico, ya que la construcción de algoritmos nos motivó a explorar e investigar de manera exhaustiva una solución hacia el problema. También, el trabajo en grupo permitió delegar actividades entre nosotros para tener una mayor eficacia al momento de progresar con el desarrollo. Un claro ejemplo de ello, es nuestra responsabilidad con el cumplimiento de entrega en hitos anteriores a este trabajo. Tanto el diseño del juego como la programación, fue desarrollado a raíz de una organización exhaustiva y ordenada. También, el proyecto terminado en su totalidad, representa nuestra capacidad de aplicar procesos de ingeniería que puedan satisfacer este tipo de necesidades. Por lo tanto, el desarrollo final de este proyecto logra cumplir con el propósito y las competencias esperadas por el curso.

Asimismo, desarrollar estas competencias nos permitió construir nuevas funcionalidades en el videojuego con el objetivo de enriquecer más la experiencia hacia el usuario. Aquellas son las siguientes:

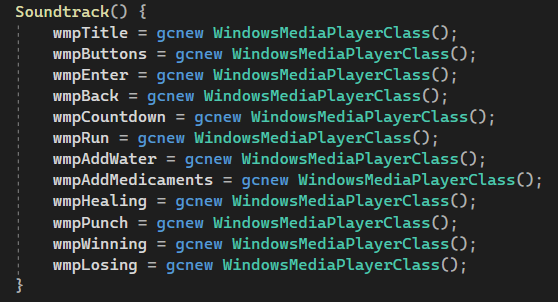
* **Música:** Para el desarrollo de la música en el videojuego, tuvimos que hacer uso de una librería adicional llamada **WMPLib.dll.** Este es un fichero, desarrollado por Microsoft, que nos permitía añadir funciones de reproducción de música a partir de la conocida **Windows Media Player**, un reproductor de antaño que actualmente no se encuentra disponible en versiones posteriores de Windows. La inclusión al proyecto fue a través de una página externa que nos proporcionaba esta librería y posteriormente añadido a las referencias de la solución del proyecto para su correcto funcionamiento.



Creamos una clase **Soundtrack** con el objetivo de centralizar los sonidos del videojuego en un solo archivo. Declaramos como atributos a **WindowsMediaPlayer** con un nombre en particular que simbolizaba las distintas utilidades que iban a tener a lo largo del videojuego.



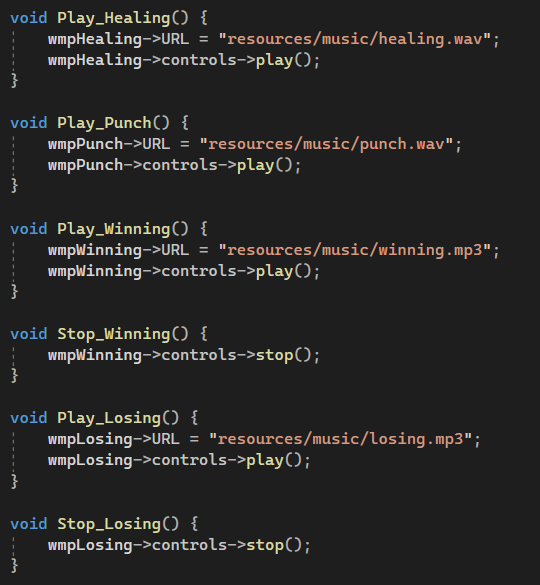
Construimos nuestro constructor, inicializando estos atributos a instancias de **WindowsMediaPlayerClass(),** con el objetivo de utilizar aquellos atributos en métodos posteriores.



Aquellos métodos son los siguientes:



****

****

Como se podrá observar, los métodos tienen una funcionalidad similar. La diferencia reside en la ruta del archivo hacia los recursos de música en cada definición y que, además, existen 2 métodos que realizan diferentes funciones. Si tomamos como ejemplo a los métodos **Play\_TitleScreen()** y **Stop\_TitleScreen()**, ambos realizan instrucciones específicas. Por un lado, el primer método reproduce la canción cuando esta sea llamada y la siguiente, detiene la reproducción de la canción.

1. **CONCLUSIONES**

Enumere las conclusiones del trabajo final enfatizando su punto de vista en base al objetivo.

1. El desarrollo del trabajo final, nos permitió afrontar un nuevo desafío y un enfoque relacionado a la programación orientada a objetos. Siendo la POO un paradigma totalmente nuevo para nosotros, nos permitió relacionar el mundo real con el código.
2. Este trabajo nos ayudará a recomendar este tipo de proyectos a nuevos estudiantes que busquen expandir más sus conocimientos en la programación. Siendo un nuevo enfoque la POO en C + +, nos sirvió para seguir investigando sobre nuevos paradigmas y a analizar sobre que podemos crear a partir de aquellas.
3. El trabajo final, desde un contexto de programación estructurada y modular, nos permitió aplicar los conceptos del mundo real a la lógica, y así poder tener una visión clara de lo que podemos lograr en la programación desde nuestras carreras.
4. **ANEXOS**

**Video:**

[**https://drive.google.com/drive/folders/1izCZauPLqJdKD6gdRFR6misRszEiBenw?usp=sharing**](https://drive.google.com/drive/folders/1izCZauPLqJdKD6gdRFR6misRszEiBenw?usp=sharing)

1. **BIBLIOGRAFÍA**

[**https://www.inei.gob.pe/media/MenuRecursivo/boletines/boletin\_agua\_junio2020.pdf**](https://www.inei.gob.pe/media/MenuRecursivo/boletines/boletin_agua_junio2020.pdf)