Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2233 Programación Avanzada (2022-1)

Tarea 2

Entrega

- **Tarea**
  - **Fecha y hora:** sábado 15 de octubre de 2022, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T2/`
- `README.md`
  - **Fecha y hora:** sábado 15 de octubre de 2022, 22:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T2/`

Objetivos

- Utilizar conceptos de interfaces y PyQt5 para implementar una aplicación gráfica e interactiva.
- Tomar decisiones de diseño y modelación en base a un documento de requisitos.
- Entender y aplicar los conceptos de *back-end* y *front-end*.
- Aplicar conocimientos de *threading* en interfaces.
- Aplicar conocimientos de señales.

Índice

1. *DCCruz vs Zombies*

En ~~capítulos~~ tareas anteriores, el malvado Hernan4444 viajó a un dimensión paralela para derrotar al tú de otro mundo en una batalla programón. Exitosamente lograste frustrar sus planes, ayudando al tú de dicha dimensión. Han pasado semanas y no has tenido noticias de este malvado personaje. Durante los últimos días tienes malos presentimientos y te preguntas que será de él ahora, por lo que, mediante tus habilidades de hackeo lograste doxearlo y obtener la dirección de su casa en esta otra dimensión. Un día decides infiltrarte e ir a la casa de este malvado ser, mientras avanzas hacia su ~~pieza~~ guarida secreta ves en el piso algo que parece un diario de vida, ante la curiosidad lo recoges y comienzas a revisar lo que tiene escrito. A medida que avanzas en la lectura te das cuenta que tiene varias anotaciones de infovis, criticas a las visualizaciones, algunas fotos de gráficos mal hechos, además encuentras un autógrafo de Tamara Munzner[1]. Cuando estas a punto de rendirte ves que detrás del autógrafo hay una última página que dice:

¡Lo logré! 🎉 🎉 *¡Descubrí el secreto de la clonación genética! lástima que al usarlo en personas, estos se convierten en zombies, pero es un detalle menor* 😅*. Ya la probé en mi amigable vecino Nico y todo salió en orden, ahora la usaré en mi y con el ejercito de clones-zombies míos y de Nicolas, iré a destruir a la única persona que ya le gané alguna vez en mi vida ... al gran maestro Cruz ... conocido en esta dimensión como Crazy Cruz.*

De repente comienzas a tener recuerdos de lo que ya ocurrió antes, al parecer tantos viajes te han afectado y recuerdas que en esta dimensión, el gran maestro Cruz no ha sido derrocado, por lo que harás todo lo posible para salvarlo esta vez de los *zombies*. Para esto, aprovecharás de tus extraordinarias habilidades en PyQt5, Threading y señales, para simular un juego donde utilizas plantas modificadas genéticamente para proteger a Crazy Cruz de oleadas de *zombies*, con esto esperas poder idear el mejor plan para mantener la casa del maestro a salvo.

2. Flujo del programa

DCCruz vs Zombies es un juego que consiste en defender la casa de Crazy Cruz de una invasión *zombie*, para ello tendrás que eliminar oleadas de *zombies* que intentarán devorar el cerebro del emblemático profesor. Para esto, tendrás a disposición un arsenal de **cuatro plantas**, especificadas en Plantas, que deberás colocar en el jardín de Crazy Cruz para defenderlo. Los *zombies* por su parte, serán de los **dos tipos** que aparecen más detallados en Zombies. El juego va por rondas, por lo tanto, a medida que se vaya avanzando de ronda, el juego aumentará su dificultad. En consecuencia, para asegurar el máximo puntaje, deberás pasar el máximo de rondas antes de eventualmente perder.

DCCruz vs Zombies contará con dos escenarios, el Jardín de la abuela y la Salida nocturna, en los que podrás jugar varias rondas hasta perder. Cada uno de estos escenarios tiene distintas dificultades asociadas a la aparición de soles y *zombies*. Luego de seleccionar un escenario, el objetivo será acabar, por cada ronda, con todos los *zombies* antes de que lleguen a la casa de Crazy Cruz. En caso de lograrlo, se calculará el Puntaje. Finalmente, tendrás la opción de avanzar a la siguiente ronda o salir.

Al iniciar el programa se mostrará la Ventana de inicio, la cual deberá tener un *input* de texto y un botón por cada una de las opciones: *Jugar*, *Ranking* y *Salir*. Al seleccionar la opción *Ranking*, deberá redirigir a la Ventana de *ranking* para visualizar los mejores puntajes registrados junto a un botón para volver a la ventana inicio. Al seleccionar la opción *Salir*, se deberá cerrar la ventana y terminar el programa. Por último, al escoger la opción *Jugar*, se deberá ingresar primero un nombre de usuario que sea alfanumérico, de caracteres acotados y que no sea vacío. En caso que **no** se cumpla algunas de estas características,

[1] Autora del libro guía utilizado en IIC2026 - Visualización de información: Visualization. Analysis and Design, 2026.

deberá aparecer un mensaje de error por medio de la interfaz dando la opción de ingresar un nombre nuevamente. En caso de que el nombre ingresado sea correcto, se deberá cerrar la ventana de inicio y abrir la Ventana principal. En esta, podrás seleccionar el **escenario** en el que deseas jugar. Una vez seleccionado el escenario y verificando lo pedido, se cerrará la ventana principal y se dará paso a la Ventana de juego, la cual mostrará el escenario escogido en la ventana principal junto a las estadísticas, opciones de juego y elementos mínimos que deberá contener el mapa. Al comenzar la ronda, los *zombies* comenzarán su invasión y deberás derrotarlos a todos. La ronda terminará una vez hayas derrotado a todos los *zombies*, cuando estos lleguen eventualmente a la casa, cuando se presione el botón avanzar o se presione el botón salir.

Una vez que haya finalizado la ronda, se abrirá la Ventana de post-ronda en la cual se mostrarán tanto las estadísticas acumuladas como de la ronda. En esta se deberá notificar al jugador si puede seguir jugando o no. En el primer caso, se dará la opción de continuar a la siguiente ronda. Por el contrario, en caso de no poder seguir, se deberá informar al jugador su puntaje total, registrándolo en `puntajes.txt` y se dará la opción de volver a la ventana de inicio.

3. Mecánicas del juego

DCCruz vs Zombies contiene ciertas mecánicas claves que deben implementarse para un correcto funcionamiento del programa. En esta sección se explica el funcionamiento de las principales mecánicas del juego.
Tanto las plantas como los *zombies* deben simular un movimiento fluido. Esto lo puede lograr usando los *sprites* proporcionados para las entidades de *DCCruz vs Zombies*. Los *zombies* deberán realizar el movimiento de su caminata, que se realizará de forma horizontal, en un sólo sentido y comenzando desde el lado derecho del mapa hacia la izquierda de este. Por otro lado, las plantas que disparan, deberán tener un movimiento fluido al disparar sus proyectiles. Además, los proyectiles deben salir disparados desde las plantas en dirección a los *zombies* para impactar en estos o llegar al borde del mapa, en ambos casos, luego de ocurrir, el proyectil deberá desaparecer. Por último, en caso de que la vida de una planta o un *zombie* llegue a `0`, **este deberá desaparecer del mapa**.

3.1. Plantas

Las plantas son una parte esencial para el funcionamiento del juego. Cada planta es única y puede poseer habilidades especiales que pueden ayudarte a formar una estrategia ganadora. Para poder plantar una planta en tu jardín debes pagar una cantidad de soles que será específico para cada planta y esta ocupará un espacio particular dentro del mapa. Esta cantidad de soles será descontada del total que hayas recolectado. Existen 2 tipos de plantas, las plantas que disparan y las plantas que no lo hacen.

3.1.1. Planta clásica

La primera planta que deberás implementar -y la más simple- es la planta clásica. Esta planta dispara proyectiles simples cada vez que pasa un tiempo `INTERVALO_DISPARO` y sus proyectiles no tienen ningún efecto adicional además del daño `DANO_PROYECTIL` que le provocan a los *zombies*. Esta planta tiene una vida igual a `VIDA_PLANTA`.

3.1.2. Planta azul

La planta azul, por su parte, provoca el mismo daño a los *zombies* que la anterior, con la diferencia de que sus proyectiles, además, disminuyen en un `RALENTIZAR_ZOMBIE`[2] por ciento la velocidad de movimiento de los *zombies* a los que impactan. Este efecto de ralentización solo debe ser implementado una única vez,

[2] Un valor entre `0` y `1`.

es decir, si un *zombie* es impactado más de una vez con uno de estos proyectiles, **su velocidad debe mantenerse constante luego del primer impacto**[3]. Al igual que la planta clásica, cada proyectil debe ser disparado con una diferencia de tiempo `INTERVALO_DISPARO` y tiene la misma vida `VIDA_PLANTA` que la planta clásica.

3.1.3. Girasol

Esta es una de las plantas más importantes para poder ganar las rondas. A pesar de no disparar ni causar daño a los *zombies*, esta planta genera soles, los cuales son necesarios para poder comprar otras plantas y ganar el juego. Los soles generados por esta planta deben aparecer periódicamente cada un intervalo de tiempo `INTERVALO_SOLES_GIRASOL` y deben aparecer en un lugar cercano al girasol que lo generó. Esta planta tiene la misma vida `VIDA_PLANTA` que la planta clásica. En ambos escenarios Jardín de la abuela y Salida nocturna, se deben generar `CANTIDAD_SOLES`[4] soles en cada intervalo mencionado anteriormente

3.1.4. Planta patata

Esta humilde planta se sacrifica por las demás para que tengas más tiempo para recolectar soles y plantar otras plantas. La planta patata no posee habilidades de ataque, pero tiene más vida que las demás y puede ser usada para obstaculizar a los *zombies* en uno de los carriles y darte más tiempo. La vida de esta planta es `2*VIDA_PLANTA`.

3.2. Zombies

Los *zombies* son los enemigos en este juego y, por lo tanto, una de las partes más esenciales para su funcionamiento. Los *zombies* aparecerán en el mapa y tratarán de llegar al extremo izquierdo del jardín, comiéndose las plantas que encuentren en su camino para lograr su objetivo. Al encontrarse un *zombie* con una planta, este deberá detenerse hasta que la vida de la planta llegue a `0` (en caso que el *zombie* aún tenga vida) y luego puede seguir su movimiento hacia el extremo izquierdo del mapa.

A medida que pasen las rondas, los *zombies* aparecerán con mayor frecuencia. Es por esta razón que, para modelar la tasa de aparición de los *zombies* en cada carril, deberás utilizar la función llamada `intervalo_aparición(ronda: int, ponderador: int) --> float` del archivo `aparicion_zombies.py` que retornará el tiempo entre la aparición de cada *zombie*, la cual recibe como parámetros, el número de la ronda actual y el ponderador de dificultad del escenario actual, `PONDERADOR_NOCTURNO` o `PONDERADOR_DIURNO` según corresponda.

Existen 2 tipos de *zombies*, el **Zombie clásico** y el **Zombie rápido**, los cuáles se diferenciarán en los *Sprites* como ***Walker*** y ***Runner***, respectivamente. El tipo de *zombie* que aparecerá, deberá ser escogido de forma aleatoria.

3.2.1. Zombie clásico

Este es el *zombie* más simple. Camina a una velocidad constante `VELOCIDAD_ZOMBIE` hasta que se encuentra con alguna planta, en cuyo caso empezará a alimentarse de ella, provocándole un daño igual a `DANO_MORDIDA` cada `INTERVALO_TIEMPO_MORDIDA` segundos. Este *zombie* posee una vida `VIDA_ZOMBIE`.

3.2.2. Zombie rápido

Este *zombie* ha dedicado gran parte de su vida a una rutina de alimentación y deporte superior a la de sus compañeros, por lo que ha desarrollado una velocidad superior a la de los otros *zombies*. La velocidad

[3] La ralentización sólo afectará la velocidad para moverse, no para alimentarse. Si `RALENTIZAR_ZOMBIE` es igual a `0.1` y la velocidad `20`, la velocidad cambiará a `18`.

[4] `CANTIDAD_SOLES` **debe ser un número entero**

de este es `1.5*VELOCIDAD_ZOMBIE` y tanto su vida como el daño de su mordida son las mismas que la del *zombie* clásico.

3.3. Soles

Los soles son necesarios para poder comprar las diferentes plantas y, así, lograr superar la invasión *zombie*. Además de ser generados por los girasoles, como fue descrito anteriormente, en el mapa **Jardín de la Abuela**, los soles deberán aparecer en **lugares aleatorios del jardín** cada un tiempo `INTERVALO_APARICION_SOLES`.

3.4. Escenarios y Dificultad

Antes de comenzar el juego podrás escoger uno de los siguientes escenarios: **Jardín de la abuela** y **Salida nocturna**. En ambos escenarios existirán dos carriles donde los *zombies* se desplazarán de derecha a izquierda. Además, en cada carril habrán 10 casillas, en las que podrás plantar solo una planta por casilla. La dificultad en cada escenario será distinta y dependerá del **ponderador de dificultad**.

3.4.1. Jardín de la abuela

Al ser de día, en este escenario se generarán soles en lugares aleatorios del jardín, los cuales te serán de gran ayuda a la hora de defenderte de los *zombies*. Este escenario tendrá asociado un ponderador de dificultad `PONDERADOR_DIURNO` con un valor de `0.9` que afectará la tasa de aparición de los *zombies* y la cantidad de puntos que se otorgarán al eliminar a todos los *zombies* de una ronda.

3.4.2. Salida nocturna

Al ser de noche, en este escenario estarás obligado a recolectar soles únicamente a partir de tus plantas Girasol. Tendrá un ponderador de dificultad `PONDERADOR_NOCTURNO` con un valor de `0.8` que afectará la tasa de aparición de los *zombies* y la cantidad de puntos que se otorgarán al eliminar a todos los *zombies* de una ronda.

3.5. Puntaje

Eliminar un *zombie* en el **Jardín de la abuela** te otorgará `PUNTAJE_ZOMBIE_DIURNO` puntos, mientras que en la **Salida Nocturna**, este evento te otorgará `PUNTAJE_ZOMBIE_NOCTURNO` puntos. Además, al eliminar a todos los *zombies* de una ronda recibirás **puntos extra** por tu buen desempeño:

$$\texttt{puntaje\_extra} = \texttt{puntaje\_ronda} \times \texttt{ponderador\_dificultad} \tag{1}$$

Donde, `puntaje_ronda` corresponde al puntaje obtenido en la ronda y `ponderador_dificultad` al ponderador de dificultad del escenario actual. Por lo tanto, en caso de eliminar a todos los *zombies*, el puntaje total será la suma del `puntaje_ronda` y del `puntaje_extra`. En otro caso, será solo el `puntaje_ronda`.

3.6. Comienzo de ronda

Al inicio de cada ronda el jardín estará vacío, comenzarás con un total de `SOLES_INICIALES` soles y podrás plantar las plantas que quieras sin que estas ejecuten su habilidad (disparar o generar soles), aparezcan *zombies* o *soles* en el escenario, **hasta que se presione el botón de comenzar ronda**. Una vez que se presione el botón, comenzarán a aparecer soles en el mapa y un total de `N_ZOMBIES` por carril con un intervalo entre cada aparición. Además, todas las plantas comenzarán a ejecutar sus habilidades (disparar o generar soles). Cabe mencionar que al avanzar de ronda, el mapa se deberá resetear al igual que los soles. Es decir, **ni los soles ni las plantas plantadas son acumulables**.

3.7. Fin de la ronda

Una vez que se eliminen a todos los *zombies* o los *zombies* invadan la casa, finalizará la ronda y se mostrará la ventana post-ronda con las estadísticas del juego. Para que la ronda se considere como ganada y pasar a la siguiente, debes eliminar a todos los *zombies*. Adicionalmente, dispones de un botón llamado **avanzar** que solo funcionará si tienes `COSTO_AVANZAR` soles. En caso de disponer dicha cantidad de soles y oprimir el botón, ganarás automáticamente la ronda **sin eliminar** a los *zombies* que faltaban, permitiendote avanzar a la siguiente ronda.

3.8. Fin del juego

El juego puede finalizar de dos formas. La primera ocurrirá si los *zombies* invaden la casa y la segunda ocurrirá si presionas el botón de salir. En ambos casos se detendrá el juego, se mostrará la Ventana de post-ronda y solo tendrás la opción de volver a la Ventana principal. Además, en estos dos casos deberás guardar el puntaje de tu partida en el archivo `puntajes.txt`. Si la ventana se cierra inesperadamente, no es necesario guardar las estadisticas de la partida que estaba en curso.

4. Interfaz gráfica

4.1. Modelación del programa

Se evaluará, entre otros, los siguientes aspectos:

- Correcta **modularización** del programa, lo que incluye una adecuada separación entre ***back-end*** y ***front-end***, con un **diseño cohesivo** y de **bajo acoplamiento**
- Correcto uso de **señales** y ***threading*** para modelar todas las interacciones en la interfaz.
- Presentación de la información y funcionalidades pedidas (puntajes o soles, por ejemplo) a través de la **interfaz gráfica**. Es decir, **no se evaluarán** *items* que solo puedan ser comprobados mediante la terminal o por código, a menos que el enunciado lo explicite.

4.2. Ventanas

Dado que *DCCruz vs Zombies* se compondrá de diversas etapas, se espera que estas se distribuyan en múltiples ventanas. Por lo tanto, tu programa deberá contener como mínimo las ventanas que serán mencionadas a continuación, junto con los elementos pedidos en cada una. Los ejemplos de ventanas expuestos en esta sección son simplemente para que te hagas una idea de cómo se deberían ver y no es necesario que tu tarea se vea exactamente igual.

4.2.1. Ventana de inicio

Es la primera ventana que se debe mostrar al jugador cuando ejecute programa. Debe incluir una línea de texto editable para ingresar el **nombre de usuario**, un botón para comenzar una nueva partida, un botón para ver el *ranking* de jugadores y un botón para salir. Si el usuario aprieta el botón de **comenzar una nueva partida**, se deberá verificar que el nombre de usuario cumple la restricción **alfanumérica**, esté en el rango de `MIN_CARACTERES` y `MAX_CARACTERES`, y que **no sea vacío**. En el caso que no se cumpla alguna restricción, se deberá notificar mediante un **mensaje** de error o *pop-up*, en la **interfaz**, indicando el motivo del error. Si cumple con todo lo antes mencionado, se debe cerrar la ventana y mostrar la Ventana principal. De otro modo, si el jugador selecciona la opción de **ver el *ranking*** de puntajes, se deberá mostrar la Ventana de *ranking*. Por último, si se selecciona la opción de **salir**, se cierra la ventana y se termina el programa.

4.2.2. Ventana de *ranking*

Debido a la gran dificultad de *DCCruz vs Zombies*, es importante dar reconocimiento a sus mejores jugadores. Por esta razón, en esta ventana se deberá mostrar los 5 mejores jugadores que han sido registrados en `puntajes.txt`, desplegando su nombre de usuario y el puntaje obtenido. Los nombres y los puntajes de los jugadores deberán ser ordenados de forma descendente, es decir, el primer puntaje será el más alto y el último puntaje será el más bajo. Además, se debe poder volver a la Ventana de inicio mediante un botón.

4.2.3. Ventana principal

En esta ventana, el jugador podrá seleccionar el escenario según la **dificultad** del juego. También, debe tener un **botón** que de inicio al juego. Se tendrá que verificar la elección de **un sólo escenario**, y avisar en el caso que esto no se cumpla. Si cumple con lo pedido, se deberá cerrar la ventana y dar paso a la **ventana de juego** para comenzar la ronda.

4.2.4. Ventana de juego

En esta ventana se desarrollan las mecánicas del *DCCruz vs Zombies*. Esta deberá contener el mapa del juego, las plantas disponibles, las estadísticas y las opciones de **comenzar la ronda**, **avanzar de ronda**, **pausar** y **salir** del juego. Las estadísticas del juego deberán actualizarse a medida que avance la ronda. Deben contener como mínimo:

- Los 4 tipos de plantas disponibles, con su respectivo costo.
- Cantidad de Soles disponibles para utilizar en la ronda.
- *Zombies* destruidos en la ronda.
- *Zombies* restantes para completar la ronda.
- Puntaje acumulado sin contar la ronda.
- Ronda actual.

Además, tu ventana debe mostrar el escenario elegido en la **ventana principal** y que la presencia de *Crazy Cruz* se mantenga durante **toda** la partida. La ronda sólo comenzará cuando se apriete el botón de comenzar ronda, **sólo en ese momento** comenzarán a aparecer los soles tanto del girasol como los aleatorios, y *zombies* junto con las plantas comenzarán su animación. La Planta clásica y Planta azul tienen que disparar su proyectil en línea recta hacia los *zombies* y también, los soles tienen que aparecer cerca del girasol que lo genera o en un lugar aleatorio si no es producido por el girasol, según corresponda. Por último, se podrá colocar en cualquier momento plantas en el escenario de juego, se especifica más en Tienda (`click izquierdo`).

La ronda será completado cuando destruyas a todos los *zombies* de la ronda, en donde debe mostrar *Crazy Cruz* anunciando que has ganado.

En el caso de perder la ronda por la llegada de un *zombie* a la casa, se debe anunciar por medio del *sprite* con el texto **"The Zombies ate your brains!"**

Si el jugador completa la ronda o pierde, esta ventana se cerrará ¿ocultará? en otras partes del enunciado dice que los elementos deben ser limpiados entre las rondas y se dará paso a la **ventana de post-ronda**. Además, si el jugador posee los soles suficientes y presiona el botón avanzar, se debe cerrar esta ventana y abrir la **ventana de post-ronda**. Finalmente, en caso que el jugador decida salir, se deberá volver a la **ventana de inicio**.

4.2.5. Ventana de post-ronda

Cada vez que se termine una ronda, se deberá mostrar esta ventana. Esta deberá contener un resumen sobre la ronda anteriormente jugada, mostrando:

- Ronda actual
- *Zombies* destruidos
- *Zombies* restantes
- Soles restantes
- Puntaje obtenido en la ronda
- Puntaje total acumulado
- Botones para continuar partida y salir del juego

Si el jugador superó la ronda, se deberá mostrar en la ventana un mensaje con esta información y se deberá mostrar habilitada la opción siguiente ronda la siguiente ronda. Si se selecciona la opción de siguiente ronda, se deberá abrir la Ventana de juego considerando la información dada en Comienzo de ronda. Por otro lado, si no se supera la ronda, se deberá mostrar un mensaje con la información de que no se puede seguir a la siguiente ronda y se tendrá deshabilitado el botón mencionado anteriormente.

5. Interacción con el usuario

5.1. *Crazy Cruz*

Crazy Cruz estará en una esquina del mapa, al terminar una ronda dirá una frase mediante un cuadro de dialogo dependiendo si el jugador gana o pierde la ronda (queda a tu criterio que es lo que dirá).

5.2. Movimiento de las plantas

En cuanto se presione el botón de comenzar partida[5] las ~~lanza guisantes~~ plantas clásicas y las ~~hielaguisantes~~ plantas azules, comenzarán a disparar independientemente si en su línea haya o no *zombies*. Por otro lado, en cuanto comience la ronda los girasoles comenzaran producir soles.

Para poder animar las plantas se entregan distintos sprites por planta (ver sección de Archivos) los cuales, al momento de realizar la animación, se deben intercalar entre si para simular un movimiento fluido.

En el caso de los guisantes, en cuanto la planta los dispare estos seguirán una linea recta comenzando desde la posición de la planta y con dirección a la zona de aparición de los *zombies*. Cuando este colisione con un *zombie*, se debe aplicar el daño y desaparecer.

5.3. Movimiento de los zombies

De manera similar a la animación de las plantas, se deberá simular el movimiento fluido de los *zombies* mientras ellos caminan, intercalando los sprites al igual que las plantas.

5.4. *Click*

Cada vez que quieras interactuar con algún botón deberá ser haciendo `click` en él. Además de los botones, podrás interactuar con los siguientes objetos dentro del la ventana de juego:

[5] Ver sección Comienzo de ronda en mecánicas de juego

5.4.1. Soles (`click derecho`)

Los soles que caerán dependiendo del mapa se podrán recolectar haciendo `click derecho` sobre ellos. Del mismo modo se podrán recolectar los soles producidos por los girasoles. Al recolectar los soles se debe agregar una cantidad `2 * SOLES_POR_RECOLECCION` a la cuenta. En caso de que se este jugando en el escenario Salida nocturna, al recolectar los soles se deberá agregar una cantidad `SOLES_POR_RECOLECCION` a la cuenta

5.4.2. Tienda (`click izquierdo`)

Durante la partida, el jugador podrá utilizar el `click izquierdo` para comprar plantas en la tienda y plantarlas en el terreno habilitado. Para realizar esta compra, deberá utilizar los soles recolectados. Cuando se entre a la ventana de juego, la tienda estará habilitada para comprar y también se podrán plantar las diferentes plantas en los lugares habilitados. **La compra y plantación sólo estará prohibida durante la pausa y el fin de la ronda**.

Para poder plantar, se debe seguir el siguiente proceso:

1. Hacer `click izquierdo` en la planta deseada de la tienda (o pala si se implementa el bonus)
2. Hacer `click izquierdo` en el lugar en que se quiera plantar (o donde se quiera quitar una planta para el caso de la pala)
3. Internamente revisar si se posee la cantidad de soles necesaria para comprar una planta Si no se posee la cantidad de soles para comprarla, se debe indicar al usuario mediante un mensaje en la interfaz, que la acción es inválida, en el caso contrario se puede continuar con el siguiente paso.
4. Internamente revisar si la posición es válida, es decir, se revisa que no exista una planta en la posición deseada y que se este plantando dentro del terreno habilitado para plantar. (en el caso de la pala se debe revisar que exista un planta en el lugar donde se hizo `click izquierdo`). Si la posición es válida se procede al último paso, de lo contrario se notificará al usuario en la misma ventana.
5. Se coloca la planta en la posición donde se hizo `click izquierdo` y se descuenta el costo de la planta seleccionada a sus soles totales. (en el caso de la pala, se quita la planta).

5.5. *Cheatcodes*

Con la única finalidad de ~~facilitar la corrección~~ mejorar la experiencia en cada ronda de una partida del juego, se deberá poder presionar ciertas combinaciones de teclas de forma continuas o simultáneas (según tu preferencia[6]) durante la partida:

- **S + U + N**: Esta combinación agrega una cantidad `SOLES_EXTRA` de soles a tu cuenta.
- **K + I + L**: Esta combinación mata a todos los *zombies* de la ronda terminando inmediatamente esta.

5.6. Pausa

En la ventana de juego, debe existir un botón de pausa que al ser presionado pause o reanude el juego, el cual también debe ser activable con la tecla **P**. El juego debe estar visualmente pausado, sin ocultar sus elementos. Los *zombies* se quedaran en quietos, las plantas dejaran de disparar, los guisantes que estén en el aire se quedaran en la posición en que estaban al momento de pausar el juego, y en general todo el programa deberá parar. Una vez que se vuelva a presionar la tecla **P** o se haga *click* sobre el botón de pausa, todos los elementos reanudarán lo que estaban haciendo, sin reiniciarse ni perderse.

[6] Se debe referenciar en el readme el método usado, ya sea pulsar teclas en orden o de forma simultánea.

6. Archivos

Para el correcto funcionamiento de la tarea, deberás hacer uso de los siguientes archivos entregados:

- *Sprites*: corresponden a los elementos visuales que se encuentran en la carpeta `sprites`.
- Sonidos: corresponde a la música del juego y las 6 frases que puede decir *Crazy Cruz* (explicada en *Bonus*) incluidos en la carpeta `sonido`.
- `aparicion_zombies.py`: este archivo cuenta con la función `intervalo_aparicion()` de *zombies* .

Adicionalmente, tu programa debe ser capaz de leer/modificar los siguientes archivos:

- `puntajes.txt`: Este archivo almacenará todos los resultados de partidas terminadas, guardando el nombre de cada usuario junto a su puntaje obtenido en el juego.
- `parametros.py`: En esta tarea, deberás crear este archivo y agregarle todos los parámetros en `ESTE_FORMATO` señalados a lo largo del enunciado, además de cualquier otro que veas pertinente añadir, como los `PATHS` o rutas de archivos.

6.1. *Sprites*

Esta carpeta contiene todas las subcarpetas de las diferentes imágenes en formato `.png` que se ocuparán para tu tarea, entre ellos se encuentra:

- **`Plantas`**: esta carpeta consta con los cuatro tipos de de plantas y 3 estados para cada una (menos para el girasol que tiene 2). Los nombres de cada uno son del estilo `{lanzaguisantes_X.png}`, `{lanzaguisantesHielo_X.png}`, `{papa_X.png}` y `{girasol_X.png}`. En los dos tipos de *Lanzaguisantes* y en el *Girasol* la `X` es el orden de los `sprites` que se **debe** utilizar para representar la sensación de movimiento, mientras que en la *Papa* la `X` es el orden de como se va deteriorando a medida se le acaba la vida, también **debe** implementarse.
- **`Zombies`**: esta carpeta contiene otras dos carpetas con las acciones que pueden tener ambos tipos de *zombie*: `Caminando` y `Comiendo`. En la primera hay dos `sprites` para simular que cada tipo de *zombie* avanza y son del estilo `{zombieHernanRunner_X.png}` y `{zombieNicoWalker_X.png}` en ellos la `X` representa el orden de los `sprites` para representar el movimiento, esto se **tiene** que implementar en el juego. Por otro lado la carpeta `Comiendo` contiene tres `sprites` para cada tipo de *zombie* que se **deben** ocupar para simular que el *zombie* está comiendo una *planta*; sus nombres son del tipo `{zombieHernanComiendo_X.png}` y `{zombieNicoComiendo_X.png}` y la `X` es el orden de los `sprites`.
- **`CrazyCruz`**: contiene un `sprite` de *Crazy Cruz* llamado como `{crazyCruz.png}`.
- **`Fondos`**: en ella se encuentran los 2 fondos posibles que determinan la ronda del juego. Esta carpeta contiene `{jardinAbuela.png}` y `{salidaNocturna.png}`
- **`Elementos juego`**: contiene una imagen de los soles liberados por el girasol y que aparecerán aleatoriamente en la pantalla del escenario de día, llamado `{sol.png}`. También se encuentran los guisantes disparados por el *Lanzaguisantes* y el *Lanzaguisantes de Hielo* con el nombre de `{guisante_X.png}` y `{guisanteHielo_X.png}`, la `X` representa el que **debe** tener el guisante: el número `1` es para cuando es disparado de la planta, mientras que el `2` y `3` es para cuando impacta al *zombie*. También se encuentra el texto que debe aparecer en la pantalla en caso de que se pierda la ronda: `{textoPerder.png}` y el logo del juego llamado, `{logo.png}`.
- **`Bonus`**: esta carpeta contiene el `sprite` utilizado en los bonus (ver la sección de *Bonus*). Este es `{pala.png}` .

6.2. Sonidos

Esta carpeta contendrá los sonidos que pueden implementarse en el juego en caso de que se realicen los bonus relacionados a la música y a *Crazy Cruz*, todo detallado en la sección de *Bonus*. Se encuentran `musica.wav` y seis sonidos del estilo `crazyCruz_X.wav`, en el que la `X` representa una de las seis opciones diferentes de diálogo.

6.3. `puntajes.txt`

Este archivo se encargará de mantener un registro de **todos los jugadores** del juego *DCCruz vs Zombies*. Cuando se finaliza una partida, se debe guardar el puntaje final del jugador en el archivo `puntaje.txt`, en el formato **usuario, puntaje**. Esta nueva fila debe posicionarse debajo de las ya existentes. Posteriormente, esta información debe ser mostrada en la **ventana de ranking**, señalando los 5 mejores puntajes de todo el archivo. Cabe recalcar que este archivo se entregará con datos existentes de registros de jugadores, por lo que debes considerar estos datos al buscar los mayores puntajes. Un ejemplo de cómo debería verse es presentado a continuación:

```
iespinazac,1250
PedroDP99,1000
jlizamajara,865
Hellonston,680
drcid98,400
```

6.4. `parametros.py`

En este archivo se encontrarán los parámetros mencionados anteriormente en el enunciado en `ESTE_FORMATO`, en donde cada línea almacena una constante con su respectivo valor. En tu tarea deberás **importar**[7] correctamente este archivo y utilizar los parámetros almacenados.

Además de incluir los parámetros descritos anteriormente, es importante incluir todo tipo de valor que será constante en tu programa, como los *paths* de archivos que se utilizarán y todo valor constante creado por ti. Es importante que los nombres de los parámetros sean declarativos, de lo contrario se caerá en una mala práctica.

Este archivo debe ser **importado como módulo** y así usar sus valores almacenados. En caso de que el enunciado no especifique un valor de algunos de los parámetros, deberás asignarlo a tu criterio, procurando que no dificulte la interacción con el juego y considerando que los ayudantes podrán modificarlo para poder corroborar que el juego sestá implementado de forma correcta.

Por otro lado, parámetros predeterminados como la ruta de los archivos o las dimensiones de la ventana y sus elementos nunca serán modificados, de tal manera que no interfiera con el funcionamiento del juego.

6.5. `aparicion_zombies.py`

En este archivo estará la función `intervalo_aparicion(ronda: int, ponderador: int) --> float`, la cual debe ser utilizada para alterar el intervalo de aparición de los *zombies* a medida que pasan las rondas. La función recibe como parámetros, el número de la ronda actual y el ponderador de dificultad del escenario actual. Para retornar un `float` que representa el tiempo entre *zombies*. **No puedes editar esta función**.

[7] Para mas información revisar el material de modularización de la semana 0.

7. *Bonus*

En esta tarea habrá una serie de *bonus* que podrás obtener. Cabe recalcar que necesitas cumplir los siguientes requerimientos para poder obtener *bonus*:

1. La nota en tu tarea (sin bonus) debe ser **igual o superior a 4.0**[8].
2. El bonus debe estar implementado **en su totalidad**, es decir, **no se dará puntaje intermedio**.

Finalmente, la cantidad máxima de décimas de *bonus* que se podrá obtener serán 9 décimas. Deberás indicar en tu `README` si implementaste alguno de los bonus, y cuáles fueron implementados.

7.1. Bonus Crazy Cruz Dinámico (3 décimas)

Para darle más protagonismo a *Crazy Cruz*, puedes implementarle un sonido característico de este personaje. Su risa se encuentra dentro de la carpeta de Sonidos y se utiliza de la siguiente manera:

- `crazyCruz_X.wav`: este sonido debe reproducirse al momento de eliminar un *zombie*. Puedes ir variando los sonidos con el valor de X en el nombre del archivo.

7.2. Pala (2 décimas)

La pala es un artefacto que te permitirá quitar una planta del jardín para poder plantar otra. Deberá añadirse a las opciones de la tienda para ser utilizado. Al momento de ser utilizado sobre una planta, esta última deberá ser eliminada del mapa.

7.3. Bonus Drag and Drop Tienda (3 décimas)

Al seleccionar una planta de la tienda, en vez de plantar mediante el uso del `click izquierdo`, deberás usar drag and drop para arrastrar la planta a su posición deseada en el mapa de juego. Al soltar la planta, este debe quedar en una posición válida, es decir, no puede quedar por encima de otro elemento, o fuera del área permitida. En caso de que se intente lo anterior, no se debe permitir colocar la planta. Una vez se suelte la planta en una posición válida, comenzará a ejecutar su función.

7.4. Bonus Música (1 décima)

Para animar un poco tu programa, decides incluir música en este. Para esto, deberás utilizar como música el archivo WAV entregado en la carpeta Sonidos llamado `musica.wav`. La música debe ejecutarse durante todo momento en el programa, reiniciándose cada vez que se cambie de ventana. Además, cuando se pause el programa, la música debe detenerse y reanudarse una vez el programa se reanude. La reproducción de esta canción se deberá hacer mediante la librería PyQt5[9] , por lo que el uso de cualquier otra librería de Python para reproducirlas será considerado como inválido y no obtendrá el puntaje

8. Propuesta de avance

Para esta tarea no habrá una entrega de avances, sin embargo se propone uno con el fin de que comiencen implementando una de las partes importantes de la tarea. El avance corresponderá a manejar apropiadamente **dos tipos diferentes de colisiones**, realizar el **movimiento fluido** de un *zombie* y el de los guisantes.

[8] Esta nota es sin considerar posibles descuentos.
[9] Las clases QSound o QMediaPlayer podrían serte útiles.

Para ello, puedes crear una ventana en donde se encuentre una planta y un *zombie*. La planta deberá disparar, según lo indicado en el enunciado, logrando que el guisante colisione con el *zombie*. Además, puedes implementar el movimiento del *zombie* y lograr que este colisione con la misma planta ya mencionada, a medida que se desplaza.

Para este avance podrás pedir un *feedback* general de lo implementado, durante la sala de ayuda para esta tarea.

9. `.gitignore`

Para esta tarea **deberás utilizar un** `.gitignore` para ignorar los archivos indicados, este deberá estar dentro de tu carpeta `Tareas/T2/`. Puedes encontrar un ejemplo de `.gitignore` en el siguiente ***link***.

Los archivos a ignorar para esta tarea son:

- Enunciado.pdf
- Carpeta de *Sprites* entregada junto al enunciado.
- Carpeta de *Sonidos* entregada junto al enunciado.
- Archivo `puntajes.txt`
- Archivo `aparicion_zombies.py`

Recuerda **no ignorar tus archivos de parámetros y archivos.ui, o tu tarea no podrá ser revisada**.

Se espera que no se suban archivos autogenerados por las interfaces de desarrollo o los entornos virtuales de Python, como por ejemplo: la carpeta `__pycache__`.

Para este punto es importante que hagan un correcto uso del archivo `.gitignore`, es decir, los archivos no **deben** subirse al repositorio debido al archivo `.gitignore` y no debido a otros medios.

10. Entregas atrasadas

Posterior a la fecha de entrega de la tarea se abrirá un formulario de Google Form, en caso de que desees que se corrija un *commit* posterior al recolectado, deberás señalar el nuevo *commit* en el *form*.

El plazo para rellenar el *form* será de 24 horas, en caso de que no lo contestes en dicho plazo, se procederá a corregir el *commit* recolectado.

11. Importante: Corrección de la tarea

Para esta tarea, el carácter funcional del programa será el pilar de la corrección, es decir, **sólo se corrigen tareas que se puedan ejecutar**. Por lo tanto, se recomienda hacer periódicamente pruebas de ejecución de su tarea y *push* en sus repositorios.

En la **distribución de puntajes**, se señalará con color:

- **Amarillo**: cada ítem que será evaluado a nivel de código, todo aquel que no esté pintado de amarillo significa que será evaluado si y sólo si se puede probar con la ejecución de su tarea.
- **Azul**: cada ítem en el que se evaluará el correcto uso de señales. Si el ítem está implementado, pero no utiliza señales, no se evaluará con el puntaje completo.

En tu archivo `README.md` deberás señalar el archivo y la línea donde se encuentran definidas las funciones o clases relacionados a esos ítems.

Finalmente, si durante la realización de tu tarea se te presenta algún problema o situación que pueda afectar tu rendimiento, no dudes en contactar a la ayudante jefa de Bienestar al siguiente correo: **bienestar.iic2233@ing.puc.cl**.

12. Restricciones y alcances

- Esta tarea es **estrictamente individual**, y está regida por el **Código de honor de Ingeniería**.
- Tu programa debe ser desarrollado en Python 3.10.
- Tu programa debe estar compuesto por uno o más archivos de extensión `.py`.
- Si no se encuentra especificado en el enunciado, supón que el uso de cualquier librería Python está prohibido. Pregunta en la *issue* especial del **foro** si es que es posible utilizar alguna librería en particular.
- Debes adjuntar un archivo `README.md` **conciso y claro**, donde describas los alcances de tu programa, cómo correrlo, las librerías usadas, los supuestos hechos, y las referencias a código externo. **Tendrás hasta 2 horas después del plazo de entrega** de la tarea para subir el `README` a tu repositorio.
- Tu tarea podría sufrir los descuentos descritos en la **guía de descuentos**.
- Entregas con atraso de más de 24 horas tendrán calificación mínima (1,0).
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

Las tareas que no cumplan con las restricciones del enunciado obtendrán la calificación mínima (1,0).