



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 Programación Avanzada (2020-1)

Tarea 02

Entrega

- **Avance de tarea**
 - **Fecha y hora:** miércoles 20 de mayo de 2020, 20:00
 - **Lugar:** GitHub — Carpeta: Tareas/T02/
- **Tarea (y Actividad Formativa 5)**
 - **Fecha y hora:** sábado 6 de junio de 2020, 20:00
 - **Lugar:** Repositorio personal de GitHub — Carpeta: Tareas/T02/
- **README.md**
 - **Fecha y hora:** lunes 8 de junio de 2020, 20:00
 - **Lugar:** Repositorio personal de GitHub — Carpeta: Tareas/T02/

Objetivos

- Utilizar conceptos de interfaces y PyQt5 para implementar una aplicación gráfica e interactiva.
- Entender y aplicar los conceptos de *back-end* y *front-end*.
- Implementar una separación entre el *back-end* y el *front-end*.
- Aplicar conocimientos de *threading* en interfaces.
- Aplicar conocimientos de señales.

Índice

1. <i>DCCafé</i>	4
2. Flujo del juego	4
3. Entidades	5
3.1. Jugador	5
3.2. Chef	5
3.3. Bocadillos	6
3.4. Clientes	7
3.5. Mesas	7
3.6. <i>DCCafé</i>	7
3.6.1. Reinicio del <i>DCCafé</i>	9
4. Interfaz	9
4.1. Modelación del programa	9
4.2. Ventanas	9
4.2.1. Ventana de inicio	9
4.2.2. Ventana de juego	10
5. Interacción del usuario con <i>DCCafé</i>	13
5.1. <i>Drag and Drop</i>	13
5.2. <i>Click</i>	13
5.3. Movimiento	13
5.3.1. Colisiones	14
5.3.2. Colisiones especiales	14
5.4. Pausa	15
6. Archivos	15
6.1. <code>mapa.csv</code>	15
6.2. <code>datos.csv</code>	16
6.3. <code>sprites</code>	16
6.4. <code>parametros.py</code>	16
7. Otras funcionalidades	16
8. <i>Sprites</i>	17
9. <i>Bonus</i>	18
9.1. Presidente (3 décimas)	18
9.2. Multijugador (5 décimas)	18
9.3. Configuración de parámetros (3 décimas)	19
9.4. Ratones en el café (3 décimas)	20
10. Avance de tarea	21
11. Actividad Formativa 5	21
12..gitignore	21
13. Entregas atrasadas	22

14.Importante: Corrección de la tarea	22
15.Restricciones y alcances	22

1. *DCCafé*

MiuEnzo despertó. Tuvo ese sueño de nuevo: atendía en el café más exitoso de todo el mundo, *DCCafé*. Sin embargo, hasta ahora, él y su amigo **Mr. Bearmartínez** solo tienen una mesa y no logran coordinarse al servir bocadillos. De pronto recordó el éxito de DCCriaturas Fantásticas gracias a ti, y decidió contactarte para pedirte ayuda. Tu misión es crear un programa con PyQt5 que ayude a MiuEnzo y Mr. Bearmartínez a entregar el mejor servicio de café de la historia.



Figura 1: Logo de *DCCafé*.

Para lograr esto, tendrás que usar tus conocimientos de interfaces gráficas y *threading* para modelar el comportamiento de una cafetería. Esto incluye la **preparación** de bocadillos y la **entrega** de estos desde la cocina hasta los clientes antes de que se acabe su **tiempo de espera**. Los clientes serán atendidos por **rondas**. En cada ronda, un número definido de clientes solicitará un bocadillo que le deberá ser entregado dentro de un plazo de tiempo. Además de esto, MiuEnzo te contó que cuando cierra las puertas del *DCCafé*, puede **comprar** mesas y **contratar** más chefs para mejorar su servicio. Sin embargo, debes tener cuidado cuando tengas demasiados clientes, pues tienes una **reputación** que cuidar y éstos se enojan cuando no reciben sus bocadillos a tiempo.

2. Flujo del juego

DCCafé es un juego que simula la administración y atención de una cafetería. El jugador tomará el rol del mesero y eventualmente podrá realizar cambios administrativos en el café, tales como contratar más chefs para agilizar la preparación de los bocadillos o comprar más mesas para poder atender más clientes.

El objetivo del juego es superar la mayor cantidad de rondas antes que la cafetería sea clausurada por mala reputación. Cada ronda tendrá un número específico de clientes que deberá atender. Al término de cada ronda se podrá conocer la reputación del *DCCafé* determinado por la cantidad de clientes que lograste atender y los que se fueron sin su bocadillo.

Al iniciar el programa se mostrará la **ventana de inicio**, en donde se dará la opción de continuar una partida existente o reiniciar y comenzar de nuevo. Luego de seleccionar una opción, la **ventana de inicio** se debe cerrar y se debe abrir la **ventana de juego**.

El juego cuenta con tres fases distintas, la primera fase es la **pre-ronda**, donde el jugador tendrá la posibilidad de poder administrar las características del *DCCafé*. Se podrá únicamente eliminar o comprar mesas y contratar chefs. Los objetos se deben comprar en una tienda, y situar en el mapa del juego mediante *drag and drop*. Luego de realizar todos los cambios que el jugador considere necesario debe presionar el botón comenzar, el cual le dará inicio a la ronda del juego.

En la segunda fase llamada **ronda** del juego, el jugador tomará el rol del mesero y deberá atender a los clientes que llegarán al *DCCafé*. Los clientes aparecerán en las mesas que están desocupadas y el mesero

deberá llevarles un bocadillo. Existen dos tipos de clientes que deben ser atendidos, los cuales tienen un tiempo de espera determinado. El chef será el encargado de preparar los bocadillos, los cuales también demorarán un tiempo determinado. Si el pedido no se entrega en el tiempo estipulado, el cliente se irá y el *DCCafé* perderá reputación. En cambio, si se logra llegar con el pedido a tiempo se obtendrá dinero, el cual servirá para realizar mejoras en el *DCCafé*. La ronda terminará cuando todos los clientes se hayan ido, ya sea porque recibieron su bocadillo o por haber esperado mucho tiempo.

Por último, en la tercera fase de **post-ronda** se dará un resumen de todo lo que ha ocurrido en la última ronda. Dentro de las estadísticas que se deberán mostrar deben estar la cantidad de clientes atendidos y perdidos, el dinero recaudado hasta el momento y la reputación que tienen el *DCCafé*. En caso que la reputación sea nula, el juego se termina y se deberá avisar al jugador que se ha clausurado el *DCCafé*, por otro lado si la reputación es mayor a cero el jugador tendrá la opción de guardar, cerrar el programa o continuar a la siguiente ronda.

3. Entidades

3.1. Jugador

El personaje principal, **MiuEnzo**, es el mesero del *DCCafé* y es controlado por el jugador. La tarea principal de MiuEnzo es la de ordenar la comida a los chefs y llevársela a los comensales.



Figura 2: MiuEnzo.

El jugador podrá interactuar con los distintos elementos del juego, ya sea por medio de colisiones, *click* o *drag and drop*. Además, deberá poder moverse libremente por todo el *DCCafé*, respetando los márgenes de la ventana y los objetos que tenga por delante, es decir, no se puede caminar por encima de los objetos sino que se debe colisionar con ellos.

3.2. Chef

Un amigo muy querido por MiuEnzo es el chef **Mr. Bearmartínez**, quien además, es su compañero de trabajo y el encargado de la cocina. Este vendrá acompañado de sus hermanos idénticos, por lo que en el café podrás tener más de un chef.



Figura 3: Mr. Bearmartínez.



Figura 4: El chef cocinando en su mesón.

Los chefs se podrán conseguir en la **tienda** y preparará los platos sobre su mesón. Una vez que estén listos los dejará junto a él para poder retirarlos, esto se debe visualizar correctamente a través de cambios de *sprites* del chef.

Por otro lado, como se dijo anteriormente en la tienda se podrá comprar cuantos chef deseas, pero además, con el tiempo podrá ir mejorando su técnica para preparar cada vez bocadillos de mejor calidad, aumentando su nivel desde **Principiante** hasta **Experto** dependiendo de la **cantidad de platos preparados**:

- **Principiante:** Es el **estado inicial** de Mr. Bearmartínez. La calidad de sus bocadillos es baja y la gente pagará una menor cantidad por ellos. Por ser el nivel más bajo, su nivel de experiencia es **1**.
- **Intermedio:** Necesita de **PLATOS_INTERMEDIO¹** platos preparados para conseguir este nivel. Sus bocadillos presentan una calidad mejorada, pero sigue siendo bastante estándar. Su nivel corresponde a **2**.
- **Experto:** Se necesita un total de **PLATOS_EXPERTO** platos para lograr el título. Preparan los mejores bocadillos de todo el DCC, son tan deliciosos que tenerlos en el *DCCafé* lo hace muy atractivo al público. Es el nivel más alto de destreza, por eso su nivel de experiencia es **3**.

Al conseguir un chef en la tienda, este recién estará comenzando a dedicarse a la cocina, es decir, será un **Principiante** por lo que es probable que se equivoque mientras cocina. Si esto ocurre, entonces la comida se perderá y MiuEnzo deberá ordenarle al chef que comience nuevamente. La probabilidad de fallar disminuirá con el aumento del nivel del chef según la fórmula:

$$\frac{0.3}{nivel_chef + 1}$$

donde **nivel_chef** corresponde al nivel de experiencia del chef.

3.3. Bocadillos

Son el corazón y la clave del éxito de *DCCafé*, por lo tanto debes preocuparte de que todos los clientes estén satisfechos con sus ordenes. El menú ofrece todo tipo de **bocadillos**, así que no te preocupes, tienes para todos los gustos (los clientes se contentarán con lo que les lleves). Todo bocadillo debe ser primero retirado en el mesón del chef, para luego ser llevado a su mesa correspondiente, de no ser así, el cliente nunca podrá gozar de tus deliciosas preparaciones y se irá. Además todos los bocadillos cuestan un total de **PRECIO_BOCADILLO** dinero.



Figura 5: Ejemplos de bocadillos del *DCCafé*.

Para poder satisfacer a tus clientes, debes estar al tanto de los **tiempos de preparación** y la **calidad** de tu bocadillo. Estos atributos son muy importantes para medir la calidad de tu atención.

- **Tiempo de preparación:** Es el tiempo que demora el chef en preparar un bocadillo para poder ser retirado por el mesero, está dado por la siguiente fórmula:

$$max(0, 15 - reputación - nivel_chef \times 2)$$

donde **reputación** y **nivel_chef** corresponden a la reputación actual de *DCCafé* y el nivel de experiencia del Mr. Bearmartínez encargado de preparar el pedido, respectivamente.

¹Las palabras escritas en **ESTE FORMATO** son parámetros que tendrás que escribir e importar desde el archivo **parametros.py**.

- **Calidad del pedido:** Mide qué tan bien preparado y delicioso es tu pedido, lo cual se traduce en una probabilidad que determinará si el cliente te entregará **propina** o no. Está dado por la siguiente fórmula:

$$\max \left(0, \left(\frac{\text{nivel_chef} \times (1 - \text{tiempo_espera} \times 0,05)}{3} \right) \right)$$

donde **tiempo_espera** está dado según el tiempo que demora el mesero en entregarle el bocadillo al cliente, desde que toman el pedido del mesón; y **nivel_chef** corresponde al nivel de experiencia del Mr. Bearmartínez encargado del bocadillo.

3.4. Clientes

Los clientes adoran visitar *DCCafé*, pero siempre y cuando sepan que serán bien atendidos, ya que de no ser así pueden afectar gravemente la **reputación del local**. Un cliente puede aparecer cada **LLEGADA_CLIENTES**, al ingresar al local, se sientan rápidamente en las mesas desocupadas (puedes hacer que aparezcan instantáneamente en ellas), hacen su pedido (siempre es un pedido único) y se marchan luego de terminar. No creas que se van sin pagar, todos los clientes pagan el precio de los bocadillos (**si lo reciben**) y además pueden dejarte una propina equivalente a **PROPIA** dinero, si los atiendes bien. Pero además, tienes que fijarte muy bien en el **tiempo que demoras** en entregarles sus pedidos (el tiempo de espera empieza a correr desde que se sientan), para ello puedes revisar los **dos diferentes tipos** de clientes y sus características:

- **Relajado:** Este cliente tiene la característica de llevarse las cosas con calma, no le preocupa esperar un par de segundos más con tal de saborear un delicioso bocadillo. Puede llegar a esperar un total de **TIEMPO_ESPERA_RELAJADO** su plato. La probabilidad de aparición de este cliente está dado por **PROB_RELAJADO**².
- **Apurado:** Hay que tener mucho cuidado con este tipo de cliente, ya que es el que más puede perjudicar tu reputación si es que no le prestas mucha atención. Siempre anda con prisa, por lo que tiempo de espera es de tan solo **TIEMPO_ESPERA_APURADO**. La probabilidad de que aparezca este cliente está dada por **PROB_APURADO**².

Todos los clientes comienzan con estado de ánimo **alegre** al llegar al *DCCafé*. Sin embargo, pasan al estado **enojado** pasado la **mitad de su tiempo de espera**. Esto conllevará un cambio en la *sprite* del cliente, lo que te permitirá visualizar los distintos estados del cliente.

3.5. Mesas

Las mesas son una parte muy importante del *DCCafé*, ya que en estas se ubicarán los diferentes clientes que lleguen por un bocadillo. Estas se podrán conseguir en la tienda y no habrá un límite de obtención aparte del espacio en la que se encuentren. Cada cliente podrá utilizar solo una mesa.

Por último, si te equivocas al ubicar una mesa no te preocupes, esto se puede arreglar. Durante la pre-ronda debes poder eliminar las mesas haciendo *click* sobre ellas. Pero cuidado, ya que no se recuperará el dinero ni la mesa.

3.6. *DCCafé*

El local de más renombre de la ciudad, el *DCCafé*. Aquí se ganan la vida Mr. Bearmartínez y MiuEnzo, con sus exquisitos bocadillos y excelente atención. Aún así, todos los locales como *DCCafé* deben cumplir con ciertas características y/o requisitos establecidos por la institución más importante: el DCC. Es por eso que como dueño del local, debes asegurarte de que todos los requisitos solicitados se cumplan. El

²Debes considerar que la suma de estas probabilidades debe ser igual a 1.

sistema financiero del DCC funciona en base a **dinero** (como te habrás dado cuenta), por lo cual se te pide como mínimo que tu local cuente con un registro de las **ganancias** por ronda, lo cual contempla los **pagos** y **propinas** por parte de los clientes. También deberás contabilizar la cantidad de pedidos **exitosos** y el **total** de pedidos por ronda. Y por supuesto, no te puedes olvidar de tus **empleados**, **clientes** y **mesas**, todo debe quedar guardado y registrado en las bases de datos de *DCCafé*³.

Además, *DCCafé* es evaluado un sistema de puntos de reputación implementado por el DCC. Este determina la reputación de tu local y mejora/empeora el desempeño de tus trabajadores. El máximo de puntos que puedes tener son **5**, pero si llegas a **perder todos tus puntos**, el DCC se verá obligado a cerrar tus puertas para siempre. Es por eso que debes tener mucho cuidado con el servicio que das a tus clientes, ya que al término de cada ronda evaluarán tu desempeño.

En resumen, tu local debe tener:

- **Dinero total:** Es un valor de tipo **int** que contempla todos el dinero generado por tu local. La **propina** (dado por calidad del pedido, explicado en Bocadillos) también debe ser incluida aquí.
- **Pedidos exitosos:** Es un valor de tipo **int** que contabiliza la cantidad de pedidos entregados con éxito durante la ronda.
- **Pedidos totales:** Es un valor de tipo **int** que contabiliza la cantidad de pedidos totales de la ronda.
- **Puntos de reputación:** Es un valor del tipo **int** que cuenta la cantidad de puntos obtenidos al final de la última ronda.
- **Ronda:** Es un valor del tipo **int** que indica el número de ronda en que te encuentras actualmente.
- **Disponibilidad:** Es un valor del tipo **bool** que determina si *DCCafé* está abierto o no. En caso de estar abierto deberá ser **True**, y en caso contrario será **False**. Este valor debe actualizarse dependiendo de si el *DCCafé* se encuentra en la pre-ronda, ronda o la post-ronda.
- **Calcular reputación:** Este método calcula la reputación del local una vez que se ha terminada la ronda, ya que es aquí donde se determina si se pierde la partida o no. La formula es la siguiente⁴:

$$\max \left(0, \min \left(5, \left(\text{reputación_actual} + \text{floor} \left(4 \times \frac{\text{pedidos_exitosos}}{\text{pedidos_totales}} - 2 \right) \right) \right) \right)$$

donde **reputación_actual** corresponde a los puntos de reputación que tenía el café al iniciar la ronda; **pedidos_totales** corresponde a la cantidad de pedidos recibidos, es decir, la cantidad de clientes que fueron al *DCCafé*; y **pedidos_exitosos** corresponde a la cantidad de clientes que fueron atendidos correctamente, es decir, aquellos que se les entregó su pedido dentro del tiempo de espera.

El resultado de esta función procederá a reemplazar el valor almacenado en **Puntos de reputación**.

- **Clientes Ronda:** Método que se encarga de calcular el número de clientes que llegarán en la ronda. Al alcanzar dicha cantidad de clientes, no deberán seguir apareciendo. Una vez se hayan marchado todos, se finaliza la ronda. Está dado por la fórmula:

$$5 \times (1 + \text{ronda_actual})$$

³Las entidades señaladas en **negrita** deberán quedar guardadas como atributos del *DCCafé*.

⁴Las funciones matemáticas **floor** la puedes encontrar en la librería **math**.

3.6.1. Reinicio del *DCCafé*

Al **crear una nueva partida** siempre comenzarás en la etapa de **ronda**, además los parámetros iniciales serán los siguientes:

- Una cantidad de dinero inicial definida por el parámetro **DINERO_INICIAL**.
- Una reputación inicial definida por el parámetro **REPUTACION_INICIAL**.
- Una cantidad de chefs iniciales definida por el parámetro **CHEFS_INICIALES**.
- Una cantidad de mesas iniciales definida por **MESAS_INICIALES**.
- Una cantidad de clientes iniciales definido por **CLIENTES_INICIALES**
- La disponibilidad comenzará en **True**.

Las mesas y mesones con los que se inicia una nueva partida deberán comenzar en posiciones aleatorias dentro del mapa, considerando que ninguna entidad puede quedar encima de otra.

Además de lo anterior, la partida guardada previamente **deberá sobrescribirse** con la nueva.

4. Interfaz

4.1. Modelación del programa

Para la correcta modelación del programa se deben tomar en cuenta distintos aspectos, dentro de los cuales se evaluarán los siguientes:

- Correcta **modularización** del programa, para poder cumplir con una adecuada separación entre en **back-end** y **front-end**, presentando un **diseño cohesivo** y con **bajo acoplamiento**⁵.
- Para el **front-end** del programa debes implementar una interfaz gráfica de usuario usando la biblioteca **PyQt5**.
- Correcto uso de **señales** y **threading**⁶ para modelar todas las interacciones en la interfaz.

Para el flujo del programa debes considerar la implementación de un **reloj interno**, por el cual se regularán todos los subprocessos del juego (tiempo de preparación de la comida, tiempo de espera de los clientes, etc). La rapidez con la que avanza el reloj es variable y debes definirlo como un parámetro.

4.2. Ventanas

Dado que el programa se compondrá de diversas etapas, se espera que éstas se distribuyan en múltiples ventanas. Tu programa deberá contener como mínimo las ventanas que mencionaremos a continuación, con el fin de que el flujo del *DCCafé* se pueda seguir correctamente. Cabe destacar que eres libre de usar los **sprites** que más te gusten, los ejemplos de ventanas expuestos en esta sección son simplemente para que te hagas una idea de cómo se deberían ver, por lo que no esperamos que tu tarea sea exactamente igual.

4.2.1. Ventana de inicio

Esta ventana es la primera que se debe mostrar al ejecutar tu programa, donde aparecerán dos opciones en forma de botones. La primera opción te permitirá continuar la partida que tienes guardada previamente. Luego de seleccionar una opción, esta ventana se **cierra** y se abre la **Ventana de juego**.

⁵Para más información puedes revisar el material [Diseño de software: Front-end y Back-end](#).

⁶Considera el uso de **Qthreads**, **Qtimer**, etc.



Figura 6: Ejemplo ventana de inicio del *DCCafé*.

4.2.2. Ventana de juego

El juego contará con tres escenarios distintos. En cada escenario deben estar presentes tres áreas que juntas darán el flujo requerido para el programa. La primera área representa el **mapa del juego**, en la cual se deberán mostrar todos los movimientos que tienen los personajes junto con la interacción del usuario. La segunda área está definida por la **tienda**, la cual estará activa en ciertos periodos del juego y el usuario podrá interactuar con ella a través de *drag and drop*. Por último, la tercera área representa las **estadísticas del juego**, las cuales deberán actualizarse en todo momento ya que están sujetas al reloj interno del programa, adicionalmente en esta área deben estar las opciones que permiten comenzar, pausar o salir del juego.

- **Ventana de pre-ronda:** Es el primer escenario del programa y el único momento donde la tienda estará activa y se podrán comprar objetos. Se puede implementar la tienda de objetos de dos maneras distintas que quedará a tu elección. La primera manera es mediante dos ventanas que interactuaran entre sí ([Figura 7](#)), por otro lado, la segunda manera es implementar la tienda en la misma ventana del juego ([Figura 8](#)).

Ambas maneras deben presentar de forma individual cada producto en venta junto a su precio y permitir al jugador comprar objetos si así lo desea (mediante *drag and drop*).

Por último, para comenzar la partida se deberá dar *click* en el botón “comenzar” lo cual bloqueará la posibilidad de comprar en la tienda y comenzará la ronda.



Figura 7: Ejemplo del *DCCafé* con dos ventanas.



Figura 8: Ejemplo del *DCCafé* con una ventana.

- **Ventana de ronda:** En este escenario se desarrolla principalmente el juego. El área que contenga el mapa del *DCCafé* junto con nuestro personaje y el resto de las entidades debe ser lo suficientemente grande, como para que el desarrollo del juego se cumpla sin problemas. El botón de “comenzar” deberá estar inhabilitado y solo se podrán usar los botones de “pausar” y “salir”. Cabe mencionar que las estadísticas de los clientes y la cantidad de dinero acumulado deben estar actualizándose a medida que avanza el juego. Al finalizar la ronda, el *DCCafé* debe quedar preparado para la siguiente ronda. Esta ventana no se debe cerrar al pasar a la **post-ronda**.



Figura 9: Ejemplo ronda del *DCCafé*.

- **Ventana de post-ronda:** Esta será una ventana que notificará al usuario con un resumen de la última ronda, por lo tanto debe contener al menos los siguientes elementos:

- Reputación actual del restaurante.
- Dinero acumulado.
- Cantidad de clientes atendidos.
- Cantidad de clientes que no lograron ser atendidos.
- Un botón para poder guardar el progreso.



Figura 10: Ejemplo ventana post-ronda.

Para pasar a la siguiente ronda quedará a tu criterio si implementar un botón que cierre la ventana o simplemente cerrar la ventana manualmente. Ambas maneras te deben situar en la **ventana de pre-ronda**. En el caso de que tu cafetería alcance el mínimo posible de reputación, se le debe informar al jugador que ha perdido y por lo tanto el *DCCafé* ha sido clausurado.

5. Interacción del usuario con *DCCafé*

5.1. *Drag and Drop*

Al comprar los distintos objetos de la tienda, deberás usar *drag and drop* para **arrastrar** los objetos de la tienda a su posición deseada en la ventana de juego. Ten en consideración que la posición en la que quedan se mide usando la esquina superior izquierda de la entidad. Al soltar dicho objeto debe quedar en una **posición válida**, es decir, no puede quedar por encima de otro ni del jugador, o fuera del mapa. En caso de que se intente lo anterior, no se descontará dinero del jugador y el objeto desaparecerá o volverá a la tienda (queda a tu criterio). Luego de esto, el objeto debe volver a estar disponible para ser arrastrado desde la tienda.

5.2. *Click*

Cada vez que quieras comenzar una nueva ronda de juego, deberás hacer *click* en un botón destinado a comenzar la siguiente ronda.

Además, las mesas y los mesones se pueden eliminar haciendo *click* sobre ellos. Esta acción no se puede deshacer, no se reembolsa dinero y **nunca puedes tener menos de una mesa y un mesón**.

5.3. Movimiento

El personaje debe poseer movimientos animados para avanzar. Para lograr, esto, debe diferenciarse su aspecto al moverse, considerando al menos tres estados de movimiento para cada dirección como se muestra en la imagen:



Figura 11: Estados de movimiento del personaje.

De esta forma, el movimiento se verá más fluido. Para esta sección, puedes usar *sprites* buscadas en internet, usar los entregados junto a la tarea o crear tus propias *sprites*. Para más información respecto a las *sprites* entregadas, puedes revisar la sección [Sprites](#).

El movimiento del personaje se da a través de las teclas WASD, y no es necesario que implementes movimiento en diagonal.

Cabe recalcar que el personaje contará con una velocidad de movimiento que indicará la cantidad de píxeles que se moverá al apretar una tecla, la cual está dada por el parámetro [VEL_MOVIMIENTO](#).

Además de los movimientos para el mesero, el chef y los clientes también deberán cambiar de *sprite* **cada vez que cambien de estado**.

Un ejemplo de los estados del chef es el siguiente:



(a) Esperando para cocinar.

(b) Cocinando.

(c) Bocadillo terminado.

Figura 12: Distintos estados del chef.

Un ejemplo de los estados de un cliente es el siguiente:



(a) Cliente alegre.

(b) Cliente enojado.

Figura 13: Distintos estados de un cliente.

5.3.1. Colisiones

Una **colisión**⁷ se produce cuando tu personaje intenta moverse hacia una **dirección inválida**. Debes tener en cuenta que una dirección inválida se da cuando el personaje intenta avanzar por encima de cualquier objeto, o intenta salirse de los límites del mapa.

Durante una colisión el personaje **no debe** detener su animación de movimiento a pesar de que no avance en el mapa.

5.3.2. Colisiones especiales

Una **colisión especial** son similares a las colisiones, con el efecto adicional de activar un evento específico en el juego. Los distintos tipos de colisiones especiales son los siguientes:

- **Clientes:** Si el personaje colisiona con uno de ellos mientras está llevando un plato, **dejará** ese plato con el cliente. El personaje queda con las manos vacías luego de esta colisión.
- **Mesón del Chef:** Si el personaje colisiona con algún mesón del chef o con algún chef, puede ocurrir lo siguiente:
 - Si el chef no está cocinando y no tiene ningún bocadillo preparado, entonces este **comenzará a preparar un bocadillo**.

⁷Una forma de comprobar colisiones es utilizando la clase `QRect` de `QtCore` y el método `intersects`. Este permite obtener la intersección de dos objetos `QRect`.

- Si el chef tiene un bocadillo preparado y el mesero no está llevando uno actualmente, entonces este último lo **recogerá**, y el chef **quedará en espera** de preparar el siguiente bocadillo.

5.4. Pausa

Debe existir un botón de pausa que al ser presionado pause o continúe el juego, el cual también debe ser activable con la tecla P. El juego debe estar visualmente pausado, sin ocultar sus elementos. Durante la pausa no se podrá mover al personaje, no avanzará la preparación de los pedidos ni el tiempo de espera de los clientes, y no llegarán nuevos clientes. Una vez que se vuelva a presionar la tecla P o se haga *click* sobre el botón de pausa, todos los procesos reanudarán lo que estaban haciendo, sin reiniciarse ni perderse.

6. Archivos

Para administrar tu *DCCafé* de un modo más profesional, se te entregarán los archivos `mapa.csv` y `datos.csv`. Estos archivos deberán **modificarse** con cada nueva ejecución del programa.

Ten en cuenta que estos archivos estarán en *encoding* UTF-8, por lo que se espera que se pueda leer y/o modificar bajo el mismo formato a lo largo de la ejecución del programa.

Además, se te entregará la carpeta `sprites`, la cual contiene imágenes que te pueden ser de utilidad para la tarea.

Deberás crear un archivo llamado `parametros.py`, el cual contendrá una serie de parámetros que serán de utilidad para la elaboración de tu tarea. **Es obligatorio importar este archivo como un módulo y utilizar los parámetros contenidos en él.**

Puedes suponer que los archivos entregados siempre se encontrarán dentro de tu carpeta T02/, por lo que, siempre habrá una partida guardada.

6.1. `mapa.csv`

Archivo que contiene las **posiciones en píxeles** de los objetos en el mapa de juego. Estas siempre deben estar dentro de los parámetros `LARGO_MAPA` y `ANCHO_MAPA`.

Las líneas de este archivo siguen el siguiente formato:

```
tipo, posicion_x, posicion_y
```

En cada línea, `tipo` puede ser `mesero`, `mesa` o `chef`, según el tipo de entidad que sea. `posicion_x` y `posicion_y` indican la coordenada en la que se ubica la esquina superior izquierda de la entidad respecto a tu punto de referencia⁸. No es necesario que estas líneas estén ordenadas por tipo.

Un ejemplo de este archivo es el siguiente:

1	<code>mesero,50,60</code>
2	<code>mesa,100,36</code>
3	<code>chef,60,55</code>
4	<code>chef,80,72</code>

El mesón y el chef cuentan como una misma entidad, por lo que la posición guardada en el archivo se refiere a la del *sprite* de ambos.

Cada vez que crees una nueva partida este archivo **debe sobrescribirse**. Para más información respecto a las posiciones iniciales de cada objeto, puedes revisar la sección [Reinicio del *DCCafé*](#).

⁸Deberás indicarlo en tu `README.md`.

6.2. `datos.csv`

Este archivo contiene los datos de la partida y consiste en dos líneas. La primera línea del archivo sigue este formato:

```
dinero,reputacion,rondas_terminadas
```

En esta línea, `dinero` es el dinero total del *DCCafé*, `reputacion` es el valor de la reputación del *DCCafé* y `rondas_terminadas` son las rondas completadas al momento de guardar la partida. La segunda línea tiene el siguiente formato:

```
platos_preparados_chef_1, platos_preparados_chef_2,...,platos_preparados_chef_n
```

En esta línea está la cantidad de platos que ha preparado cada chef. Queda a tu criterio el modo de relacionar cada entidad de chef con sus platos preparados, pero debes tener en cuenta que a cada chef del mapa le corresponde solo un valor de platos preparados y viceversa.

Un ejemplo de este archivo es el siguiente:

1	500,3,6
2	16,13

6.3. `sprites`

En esta carpeta se encuentran separados los distintos *sprites* que puedes usar para la tarea. Para mayor información respecto a las *sprites*, puedes revisar la sección [Sprites](#).

6.4. `parametros.py`

A lo largo del enunciado se han ido presentando distintos números y palabras en [ESTE FORMATO](#), estos son conocidos como **parámetros** del programa y son valores que permanecerán constantes a lo largo de toda la ejecución de tu código.

Para esta tarea deberás **crear y subir un archivo `parametros.py`** el cual contenga todos los parámetros nombrados anteriormente, además de todos los *paths* y cualquier otro valor constante que vayas a utilizar en tu código. Cada línea de este archivo representará una constante junto con su respectivo valor. En caso de que no se especifique el valor de un parámetro, deberás asignarlos a tu criterio, procurando que no dificulte la interacción con el programa.

Es posible que los ayudantes modifiquen estos parámetros durante la corrección, pero los que hagan referencia a dimensiones de mapa o de *sprites* **no serán modificados** para no complicar la visualización de estos. Sin embargo, debes agregarlos al archivo de igual modo.

7. Otras funcionalidades

Con la finalidad de ~~facilitar la corrección~~ mejorar la experiencia de juego, es posible presionar un conjunto de teclas en orden, o al mismo tiempo (queda a tu criterio)⁹, para hacer “trampa”:

- **M + O + N:** Esta combinación de teclas aumenta tu dinero en [DINERO_TRAMPA](#).
- **F + I + N:** Esta combinación de teclas termina la ronda actual, lo que hace que todos los clientes se vayan al instante sin pagar. Si se usa estando fuera de una ronda, no tendrá efecto.

⁹Debes especificarlo en tu `README.md`.

- **R + T + G:** Esta combinación de teclas aumenta al instante la reputación de tu *DCCafé* en [REPUTACION_TRAMPA](#).

8. *Sprites*

Para esta tarea, se recomienda el uso de *sprites* para representar gráficamente las distintas entidades del juego. Se entregará la carpeta **sprites**, la cual contiene más *sprites* de los que necesitarás para esta tarea¹⁰, y están separadas en:

- **mapa:** Carpeta que contiene todas las imágenes relacionadas al mapa de juego.



Figura 14: Ejemplos de *sprites* relacionadas al mapa.

- **mesero:** Carpeta que contiene los *sprites* de movimiento del personaje.



Figura 15: Ejemplos de *sprites* del mesero.

- **chef:** Carpeta que contiene las *sprites* de las distintas faces del chef en su mesón y otras que tal vez te puedan servir.



Figura 16: Ejemplos de *sprites* del chef.

- **clientes:** Carpeta que contiene los *sprites* de distintos clientes y otros que puedes usar con libertad.



Figura 17: Ejemplos de *sprites* de clientes.

- **bocadillos:** Carpeta que contiene los *sprites* de distintos tipos de bocadillos.

¹⁰Esto es con el fin de que puedas darle a tu programa la apariencia que más te guste.



Figura 18: Ejemplos de *sprites* de los bocadillos.

Adicionalmente en algunas de estas carpetas se incluirán las *spritesheets*¹¹ relacionadas, para que puedas ver con mayor facilidad cómo se relacionan las *sprites* entregadas.

9. *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**¹².
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 8 décimas. Deberás indicar en tu README si implementaste alguno de los bonus, y cuáles fueron implementados.

9.1. Presidente (3 décimas)

No es un día cualquiera si el presidente llega a *DCCafé*, sus gustos son únicos y es quién mayor influencia puede tener en tu reputación. Si eres atento con él, aumentará en **2** puntos tu reputación, pero si lo descuidas pagarás con la misma cantidad. Su tiempo de espera es un valor aleatorio entre los **15** y los **25**, el cual cambiará cada vez que el presidente aparezca en el *DCCafé*. La probabilidad de que aparezca al comenzar una nueva ronda es del **10** por ciento. Además de lo ya mencionado, deberá ser tratado como cualquier cliente.



Figura 19: Presidente junto a su bocadillo preferido.

9.2. Multijugador (5 décimas)

Para obtener este *bonus*, deberás implementar la posibilidad de que haya un segundo jugador en el juego. En la ventana de inicio, debe existir un botón con el cual activar el modo multijugador, independiente de si se juega una nueva partida o se continua una anterior.

A diferencia del jugador principal, el segundo jugador aparecerá en una posición inicial aleatoria (teniendo en cuenta que debe ser dentro del mapa y no puede aparecer encima de otras entidades). Este deberá

¹¹Una *spritesheet* es una serie de imágenes (usualmente cuadros de animación) combinadas en una imagen más grande.

¹²Esta nota es sin considerar posibles descuentos.

poder moverse usando las teclas IJKL, y debe tener el mismo comportamiento que el jugador principal. Esto incluye las colisiones comunes, especiales y se debe detener al pausar el juego.

Un jugador no puede pasar por encima de otro, y cuando un jugador colisiona con otro debe comportarse igual que al colisionar con una mesa o el borde del mapa.

Ambos jugadores deben poder diferenciarse por su aspecto. Eres libre de elegir con cuál *sprite* representar al segundo jugador. Un ejemplo de representación es el siguiente:

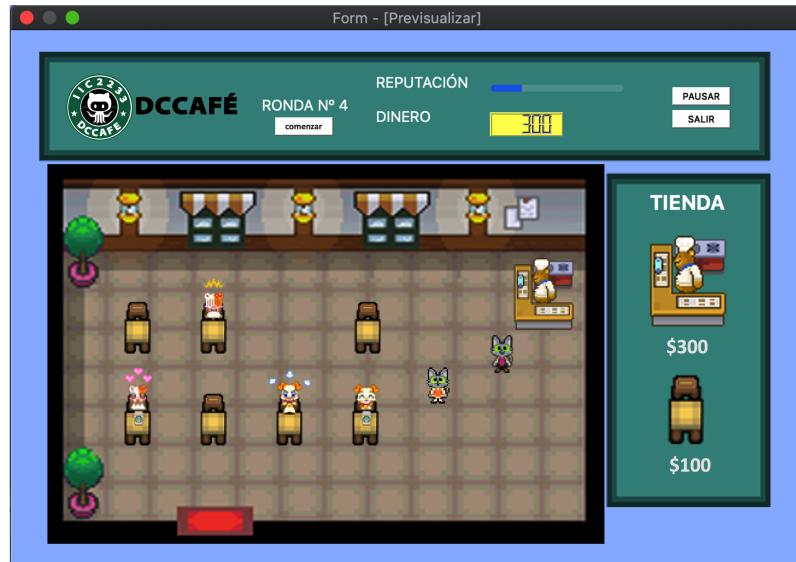


Figura 20: Ventana de juego con dos jugadores.

9.3. Configuración de parámetros (3 décimas)

Para obtener este *bonus*, debe existir un botón **Configuración** en la ventana de inicio, el cual debe mostrar un *pop-up* que permita **editar todos los parámetros** del juego. Debe haber un campo numérico o de texto por parámetro¹³.

Esta ventana de configuración debe tener un botón para **aplicar los cambios**, el cual modifica el archivo `parametros.py`, y también un botón para **cancelar**, el cual cerrará la ventana y no guardará los cambios. Los cambios deben surtir efecto al momento de presionar el botón de aplicar. Un ejemplo de ventana de configuración es el siguiente:

¹³Si se dificulta ver todos al mismo tiempo, puedes utilizar `QScrollArea` de `QtWidgets` para hacer una ventana *scrollable*, o también puedes implementar varias páginas usando botones.

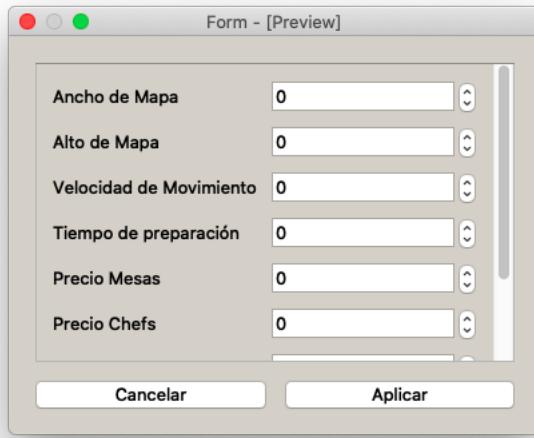


Figura 21: Ejemplo de ventana de configuración.

9.4. Ratones en el café (3 décimas)

¡Tu mala reputación ha atraído a nefastos roedores al *DCCafé!* Para obtener este *bonus*, deberás implementar la aparición de ratones en el mapa del juego.

Este evento ocurrirá cuando se alcance **REPUTACION_RATONES** o menos de reputación. Al comenzar la siguiente ronda, aparecerán entre **1** y **5** ratones en posiciones aleatorias válidas del mapa, y deberás eliminarlos colisionando con ellos. Cuando se produzca la colisión, el ratón desaparecerá y el jugador ganará **DINERO_RATONES**. No es necesario que implementes movimiento de ratones.

Un ejemplo de ventana de juego con ratones es el siguiente:



Figura 22: Ventana de juego con ratones.

10. Avance de tarea

Para esta tarea, el avance corresponderá a implementar el cargado visual del mapa a partir de las información contenida en el archivo `mapa.csv`.

Es importante recalcar que deberás entregar:

- Una interfaz gráfica diseñada con PyQt5, la cual deberá presentar un **correcto uso de señales** para realizar la comunicación entre el *back-end* y el *front-end*.
- El cargado del mapa deberá adaptarse a los cambios que pueda presentar el contenido del archivo `datos.csv`.

A partir de los avances entregados, se les brindará un *feedback* general de lo que implementaron en sus programas y además, les permitirá optar por **hasta 2 décimas** adicionales en la nota final de su tarea.

11. Actividad Formativa 5

Para obtener el puntaje asociado a la Actividad Formativa 5, se espera que realices un subconjunto de los requisitos ya pedidos en este enunciado. El conjunto de funcionalidades que te pedimos realizar a continuación, tiene el objetivo de poder interiorizarte en la elaboración de un programa mediante una ventana que contenga *widgets* en su interior, además de poder trabajar con *threading* y señales.

Los requisitos relacionados con la AF05 se refieren a la utilización de elementos gráficos, señales y *threading* para representar el **mapa de juego** dentro de una ventana en la que se pueda apreciar una representación visual de las entidades Jugador, Chef, Clientes y Mesas. Esta debe incluir la animación y cambio de estado del Cliente. Se espera que el Cliente llegue a la mesa, cambie de estado después de un tiempo, y luego se vaya, tal como debería ocurrir en el juego completo. No se esperan otras animaciones ni la implementación de interacción por medio de colisiones, que no son requisitos para obtener este puntaje. En esta sección tampoco se evaluarán el área de **tienda** ni **estadísticas de juego**, sino que solo el **mapa de juego**.

Como notarás, esto ya es parte de los requisitos de la tarea, por lo que puedes implementar tu tarea normalmente y obtener este puntaje adicional (que irá a tu puntaje acumulado de actividades formativas), ya que la implementación de los requisitos de la AF05 no interfiere con la implementación de la T02. Si no deseas realizar la T02, pero sí obtener el puntaje de la AF05, puedes implementar los requisitos ya descritos, indicando en tu `README` que solo estás entregando la AF05 y no la T02.

12. `.gitignore`

Para esta tarea **deberás utilizar un `.gitignore`** para ignorar los archivos indicados, este deberá estar dentro de tu carpeta `T02/`. Puedes encontrar un ejemplo de `.gitignore` en el siguiente [link](#).

Deberás ignorar el enunciado, los archivo entregados `mapa.csv`, `datos.csv` y la carpeta llamada **sprites**. El archivo `parametros.py` **no debe ser ignorado. Debe ser entregado**.

En caso de que hagas uso de tus propias *sprites*, deberás guardarlas en otra carpeta y asegurarte que no sean ignoradas por el `.gitignore`.

Se espera que no se suban archivos autogenerados por programas como PyCharm, o los generados por 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.

13. Entregas atrasadas

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

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

14. 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.

Cuando se publique 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.

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, se recomienda el uso de *prints* para ver los estados del sistema (estado de los clientes, posición del personaje, colisiones, etc.) pero no se evaluará ningún ítem por consola. Esto implica que hacer *print* del resultado una función, método o atributo no valida, en la corrección, que dicha función esté correctamente implementada. **Todo debe verse reflejado en la interfaz.** Por ejemplo, si se utiliza el dinero del café al comprar un ítem, que hagan *print* del dinero disminuyendo no será válido, para este ítem se evaluará que el contador de dinero en la interfaz disminuya.

15. 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 3.7.
- 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 48 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 cumplen con las restricciones del enunciado obtendrán la calificación mínima (1,0).