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

IIC2233 - Programación Avanzada (I/2017)
Tarea 5

1. Objetivos

- Aplicar conocimientos de creación de interfaces.
- Aplicar conocimientos de threading en interfaces.

2. Introducción

Después de haber estado simulando la participación de los alumnos en el ramo de Avanzación Programada, la moral de todos los participantes del curso bajó un poco. Como tu eres un buen compañero, deseas subir el ánimo de tus amigos y por eso se te ocurrió una gran idea: crear un juego para distraerte de las maldades del Dr. Mavrakis.

3. League of Progra

El juego consiste en una batalla entre un jugador y la computadora. Cada uno de ellos tiene un equipo compuesto por un grupo de *súbditos*, un *campeón*, una serie de estructuras de defensa y una estructura central llamada *Nexo*. El objetivo del juego es que un equipo destruya el *Nexo* del equipo enemigo. A medida que progresa una partida, el jugador también podrá comprar en la *tienda* algunos objetos temporales que le ayudan a derrotar al enemigo. Por último el juego posee un sistema de historial donde guarda la información de las ultimas 5 partidas jugadas por el jugador. En la figura 1 se muestra un ejemplo de una partida en progreso.

Al inicio, el programa muestra una pantalla con botones para poder realizar las siguientes acciones:

- Iniciar una partida.
- Borrar el historial de partidas.

Cada vez que el jugador inicia una partida nueva, el juego permite elegir un *campeón* para el equipo, que se mantiene durante toda la partida. Dentro de la *tienda*, el juego da las opciones correspondientes de qué comprar, su valor y cuántos puntos tiene el jugador para gastar en la tienda.

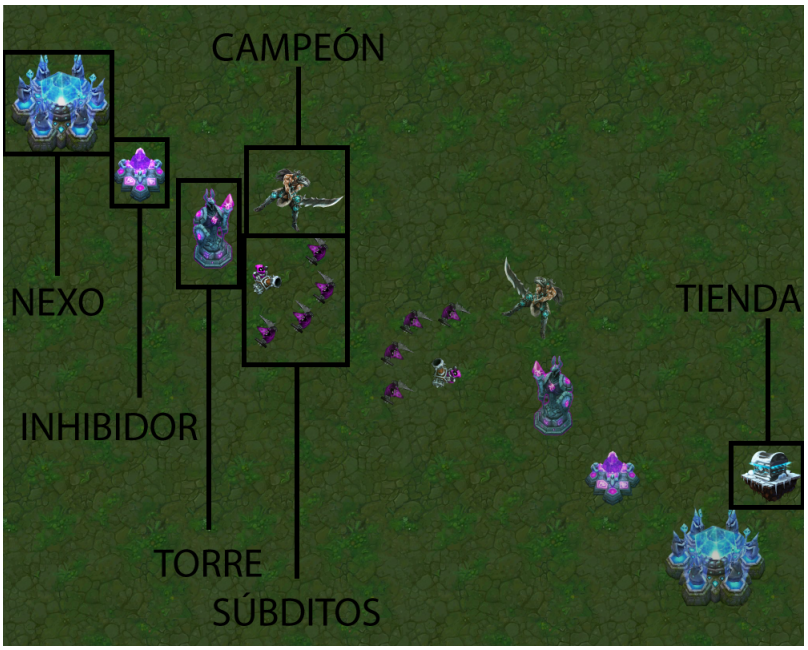

Figura 1: Ejemplo de un juego en progreso.

4. Edificios y Unidades [46.8 %]

En este juego es posible diferenciar a los elementos del juego en dos categorías: los edificios y las unidades. Exceptuando la tienda, todos los elementos del juego debe tener los siguientes atributos:

- Vida: valor numérico mayor a 0 que controla el estado de salud de la entidad. Cuando se recibe un ataque este valor se reduce y cuando llega a 0 la entidad se considera muerta. Este atributo debe poder visualizarse con una barra de vida que indique su valor actual.
- Ubicación: cada edificio y unidad tiene ubicación de la forma (x, y) que representa la posición actual en el mapa (usen la convención que `(0,0)` corresponde a la esquina superior izquierda del mapa y `(x,y)` corresponde a la esquina superior izquierda de la figura). Las posiciones iniciales están dadas por el mapa de la Figura 1. Estas ubicaciones son únicas y ningún edificio o unidad puede estar en el mismo lugar que otra. Tampoco pueden tocarse entre ellas, considerando su tamaño en pixeles. Esto significa que las unidades no son un punto, sino que hay que tener en cuenta un área al momento de evitar una colisión.

4.1. Unidades

Las unidades corresponden a los *campeones* y *súbditos*. Ambas pueden pelear, morir en combate y desplazarse en el mapa. Toda unidad tiene:

- Velocidad de movimiento: parámetro indicado en pixeles por segundo. Mientras más grande es este parámetro, más rápido se mueve la unidad.

- Daño: valor numérico que indica cuánta vida se le puede reducir a otra unidad o edificio mediante un ataque.
- Velocidad de ataque: parámetro que indica la cantidad de ataques por segundo que puede efectuar la unidad
- Rango de ataque: parámetro que indica la distancia máxima en pixeles a la que puede encontrarse una unidad enemiga para poder ser atacada.

Cada unidad conoce la ubicación de todas las entidades (aliadas y enemigas) en el mapa.

4.1.1. Campeones

Los *campeones* son controlados por el jugador o la computadora y son los personajes más poderosos del equipo. Con ellos se puede matar a los *súbditos*, *campeones* y edificios incluido el *Nexo* enemigo. Su posición inicial es al lado de la *tienda* del equipo. El jugador escoge su *campeón* antes de iniciar una partida.

Si el *campeón* del equipo muere, revive automáticamente después de un tiempo t_r. Su tiempo de resurrección (t_r) aumenta exponencialmente en función de la cantidad de muertes (c_m), según la siguiente fórmula:

$$t_r = 1{,}1^{c_m} \times t_{rb}$$

donde t_{rb} es el tiempo de resurrección inicial. Un valor adecuado para t_{rb} es 10 segundos. Mientras el *campeón* de un equipo está muerto solo puede utilizar la tienda. El *campeón* revive siempre en su posición inicial.

Cada *campeón* posee:

- Habilidad única: Ataque único de cada *campeón*, dependiendo de la elección tienen distintos efectos.
- Enfriamiento de habilidad: La habilidad única solo puede usarse cada cierta cantidad de tiempo t_e. Esta cantidad depende del *campeón* elegido.

La habilidad única se activa con el *click* derecho del mouse, y para volver a ser utilizada debe transcurrir t_e. Los *campeones* que están soportados en el juego son:

- **Chau la hechicera**. Esta valiente hechicera controla el tiempo, por lo cual puede crear un hechizo que congela a todos sus enemigos por 10 segundos mientras ella y sus *súbditos* atacan libremente. Ataca desde una gran distancia y es muy rápida pero su sagrado hechizo no hace daño a sus enemigos. Su rango es de 40 pixeles, su velocidad de ataque es de 10, su velocidad de movimiento es de 30 y su daño es de 5. Su habilidad puede ser ocupada cada 30 segundos, y tiene 500 unidades de vida.
- **Hernán el destructor**. Es un gran bárbaro barbudo cuya habilidad es producir sismos, los cuales alejan a los súbditos enemigos que estén alrededor de él a un rango de 30 pixeles y dañan en 100 a las estructuras enemigas que se encuentran a 70 o menos pixeles de Hernán. Su velocidad de movimiento es de 10 pixeles por segundo, la velocidad de ataque es de 10, tiene un daño de 20 y su rango de ataque es de 5 es decir, ataca cuerpo a cuerpo. Su habilidad puede ser ocupada cada 40 segundos y tiene 666 unidades de vida.

- **Libre elección**. Este campeón quedó sin ningún detalle esperando que algún futuro desarrollador, como ustedes, dejen volar su imaginación e inventen uno nuevo con una nueva habilidad. Debe haber un campeón creado por usted con una habilidad nueva que ustedes quieran mientras sea razonable en comparación a los otros campeones.

Personalidad enemigo En cada partida te enfrentas contra un *campeón* elegido aleatoriamente por la computadora, incluso puede ser el mismo que has escogido para la partida actual. La forma de controlar al *campeón* dependerá de la personalidad que tenga el enemigo, que puede ser:

- *Noob*: Esta es la personalidad de un novato. Solo intentará destruir lo que esté más cercano a él, sin importar si esto implica su muerte.
- *Normal*: Esta personalidad corresponde a la de un jugador normal. Tiene una noción de estrategia mejor a la del *Noob*. Tiene el siguiente comportamiento:
  - Ataca a la torre solo si hay *súbditos* atacando la torre también.
  - Si no hay ningún *súbdito* aliado en el rango de ataque del *campeón*, éste ataca si:
    - su vida actual es mayor o igual al 20 % de su vida inicial
    - o si su daño es mayor al daño enemigo y tiene más vida que él.
  - Si su vida es mayor al 50 % de su vida inicial, usará la habilidad especial cada vez que pueda. En caso contrario solo usará la habilidad si la vida del campeón enemigo es menor al 50 % de la inicial
- *Ragequitter*: Este es un personaje llorón que juega como uno de personalidad normal hasta que la diferencia entre sus muertes y asesinatos (muertes - asesinatos) es mayor a 3. En ese caso se rendirá y se dejará de mover hasta que termine el juego.

Al momento de empezar la partida, se selecciona de forma aleatoria la personalidad del *campeón* enemigo.

4.1.2. Súbditos

Los *súbditos*, a diferencia de los *campeones*, no pueden ser controlados por el usuario, es decir, se mueven y atacan automáticamente (según como están programados). Estos reaparecen en grupos de a cinco unidades cada diez segundos. La ubicación de su aparición es al frente del *Nexo* correspondiente a su equipo. Si no están peleando, se moverán en dirección al *Nexo* enemigo.

Los *súbditos* tienen objetivos fijos, con el orden de prioridad que se muestra a continuación:

1. Defender a su *campeón* en caso de que lo estén atacando, siempre y cuando esté dentro del rango de ataque del *súbdito*.
2. Atacar a la estructura más cercana.

Independiente de las prioridades anteriores, si algún *súbdito* o *campeón* enemigo se interpone en su camino, el *súbdito* lo atacará. Con interponerse significa que esté por donde va a pasar.

Existen dos tipos de *súbditos*:

- Normal: Este *súbdito* tiene una vida de 45, se mueve a 8 pixeles por segundo, su daño es 2, solo puede atacar una vez por segundo y puede hacerlo desde una distancia de 5 pixeles.
- Grande: Este *súbdito* tiene una vida de 60, se mueve a 8 pixeles por segundo, su daño es 4, solo puede atacar una vez por segundo y puede hacerlo desde una distancia de 20 pixeles. Además, cuenta con una habilidad especial. Cuando el *inhibidor* (ver sección 4.2.2) enemigo está destruido, aumenta su tamaño y poder. Esto significa que su vida aumenta a 120 y el daño que causa a 10. Al reaparecer el inhibidor, los *súbditos* vuelven a la normalidad.

Cada 10 segundos se agregarán 5 *súbditos*: cuatro normales y uno grande.

4.2. Edificios

Los edificios se caracterizan por permanecer en la misma posición durante todo el juego hasta que sean destruidas.

4.2.1. Torre

La *torre* es la única entidad fija que puede atacar. Es la primera estructura con la que se encontrará el enemigo al intentar invadir una base. Tiene los siguientes tres atributos adicionales (además de vida y ubicación):

- Daño: valor numérico que indica cuánta vida se le puede reducir a otra unidad o edificio mediante un ataque.
- Velocidad de ataque: parámetro que indica la cantidad de ataques por segundo que puede efectuar la entidad
- Rango de ataque: parámetro que indica la distancia máxima a la que puede encontrarse una unidad enemiga para poder ser atacada.

La *torre* tiene una vida de 250, su daño es 30, solo puede atacar una vez por segundo y puede atacar a enemigos que se encuentren hasta 40 pixeles de distancia.

El comportamiento de ataque de una *torre* sigue estas prioridades, en la medida que las unidades enemigas se encuentren en su rango de ataque:

1. Defender a su *campeón*: Si el *campeón* está siendo atacado por otro *campeón* o *súbdito* enemigo, la prioridad de la *torre* será atacar a esas unidades.
2. Atacar a los *súbditos grandes*.
3. Atacar a los *súbditos* con menos vida.

Cuando la vida de la *torre* llega a 0, ésta es destruida y no revive.

4.2.2. Inhibidor

El *inhibidor* es una estructura que no permite que al *Nexo* reciba daño. Al ser destruido, éste volverá a aparecer después de 30 segundos; por lo tanto, si durante el ataque al *Nexo* re-aparece el *inhibidor*, no podrás seguir haciendo daño al *Nexo* y deberás volver a destruir el *inhibidor*. Tiene 600 de vida.

4.2.3. Nexo

El *Nexo* es la estructura objetivo del juego. El equipo que destruye el *Nexo* enemigo gana la partida. El *Nexo* solamente puede ser atacado mientras el *inhibidor* está destruido. Tiene 1200 de vida.

4.2.4. Tienda

La *Tienda* se encuentra en tu base, a un costado de tu *Nexo*. En la *Tienda* puedes comprar objetos que mejorarán los atributos de tu *campeón* durante la partida. Para poder comprar estos objetos necesitarás *puntos*, los cuales se obtienen matando *súbditos*, destruyendo torres o derribando al *campeón* enemigo. Esta estructura es indestructible y se accede a su menú para comprar objetos haciendo *click* sobre ella. Solo se puede acceder a la *Tienda* cuando el *campeón* está en un rango de a lo más 50 pixeles de ella o cuando el campeón está muerto. Al ingresar a la *Tienda*, el juego **no** se pausa, sino que se abre la *Tienda* en una ventana nueva. Cada vez que se cumplan las condiciones para poder usar la tienda, se debe informar al jugador que la opción de usar la tienda está disponible. Todo esto debe ser hecho sin detener el juego.

Objetos de la tienda Existen seis objetos que mejoran distintos atributos del *campeón*. Cada objeto posee un precio p y una bonificación de atributo b. Al comprar un objeto, se descuentan los *puntos* correspondientes al equipo, y se bonifica el atributo que corresponda al *campeón*.

Después de comprar un objeto, éste recibe un *upgrade* y tiene un nuevo precio p' y una bonificación de atributo b'. El precio nuevo es $\lceil p' = p + \frac{p}{2} \rceil$, y la bonificación nueva es $b' = 1{,}1b$. En un mismo acceso a la *Tienda*, se puede comprar múltiples veces un objeto, siempre y cuando cada vez que se compre, éste reciba el *upgrade* correspondiente y se cuente con los puntos necesarios para adquirirlo nuevamente.

Los objetos de la tienda son:

- Arma de Mano: Aumenta el daño que infringe tu *campeón* a todos los enemigos. El precio base es de 5 puntos y aumenta en 2 unidades el daño del *campeón*.
- Arma de distancia: Aumenta la distancia desde la cual puede atacar tu *campeón*. El precio base es de 5 puntos y aumenta en 2 unidades la distancia desde la cual puede atacar el *campeón*.
- Botas: Aumenta la rapidez a la cual se desplaza tu *campeón*. El precio base es de 2 puntos y aumenta en 3 unidades la rapidez con la que se mueve el *campeón*.
- Báculo u Objeto mágico: Potencia la habilidad de tu *campeón*. El precio base es de 7 puntos y potencia en 2 unidades la habilidad del *campeón*.
- Armadura: incrementa la resistencia a los golpes enemigos de tu *campeón*. El precio base es de 5 puntos y disminuye en 2 unidades el daño que recibe el *campeón*.

- Carta Earthstone: Mejora tu mente, por lo que puedes mejorar aleatoriamente cualquier atributo de tu *campeón* al azar. Esto es, al momento de comprar este objeto se indica al jugador que atributo aleatoriamente fue potenciado. El precio base es de 10 puntos y aumenta en 6 unidades la categoría que salga seleccionada al azar.

A continuación puedes ver el cambio que debes mostrar en la tienda cuando se compre un objeto:

Figura 2: Ejemplo de la compra de un Arma de Mano.

5. Puntos [1,7 %]

Durante el transcurso de una partida, un jugador aniquilará *súbditos* del otro equipo y al otro *campeón*. Tu *campeón* recibe 1 punto en caso de eliminar a un *súbdito* y 5 en caso de matar un *campeón*. Al destruir la *torre* o el *inhibidor* se ganan 15 puntos. Con estos puntos podrás comprar mejoras que durarán por el resto de la partida en la que fueron compradas. La cantidad de puntos poseídos debe ser mostrada en todo momento.

6. Datos almacenados [6,6 %]

Una de las nuevas funcionalidades que tiene *League of Progra* es su capacidad de tener todos los datos del juego y usuario en archivos fuera del código principal con el fin de poder editarlos posteriormente. El formato de estos archivos puede ser un `csv`, `pickle`, `txt` o cualquier otro formato que no requiera el uso de una librería que no ha sido permitida. La aplicación debe generar un archivo para guardar los datos de la partida y otro para mantener las constantes del juego.

6.1. Partida Guardada

Este archivo se encarga de registrar todo el avance del jugador. Esto incluye la cantidad de puntos acumulados y un historial de las últimas 5 partidas, indicando por equipo:

1. Asesinatos realizados por el *campeón*
2. Número de muertes del *campeón*
3. Número de veces que el *campeón* activó su habilidad.
4. Cantidad de puntos obtenidos durante la partida.

5. Cantidad de puntos gastados durante la partida.

6.2. Constantes

Este archivo contiene las constantes utilizada en el juego, por ejemplo: vida máxima de los *campeones*, vida de los *súbditos*, cantidad de daño por habilidad, velocidades de ataque. **Incluya todas las constantes que defina para su juego**. Puede existir más de un archivo con constantes, si eso simplifica la organización. Considere que este archivo debe ser leído al iniciar el juego y a partir de él obtener todas las constantes que serán utilizadas.

7. Controles [20 %]

El *campeón* debe moverse y atacar utilizando el teclado y el *mouse*, como se indica a continuación.

7.1. Movimiento

Los botones del teclado que le permiten moverse son:

- A: Moverse hacia la izquierda.
- S: Moverse hacia atrás.
- D: Moverse hacia la derecha.
- W: Moverse hacia adelante.

El *campeón* tendrá como eje al puntero del *mouse*, por lo que siempre estará mirando en la dirección donde se encuentre éste. Las teclas le permitirán moverse hacia adelante, atrás, a la derecha o izquierda respecto de la dirección donde esté mirando el *campeón*; es decir, si oprimes la tecla `W` el *campeón* debe acercarse hacia donde esté el puntero del *mouse*. La figura 3 muestra en qué dirección se movería el *campeón* si el puntero del *mouse* está en esa posición.

7.2. Ataque

Los botones que le permiten al *campeón* atacar son los siguientes:

- Click Izquierdo: El *campeón* ataca al enemigo que se encuentre bajo el puntero del *mouse* cuando oprimas este botón. En caso de que el enemigo se encuentre fuera de rango, el *campeón* debe moverse de forma automática hasta que se encuentre dentro del rango de ataque. Mientras el *campeón* se encuentra caminando hacia su objetivo no debe seguir con la vista al puntero del mouse. Si el *campeón* choca con otra entidad en el camino hacia su objetivo, queda a tu criterio si escoges que se quede chocando hasta que esa entidad no se interponga, o que rodee esa entidad para continuar su camino.

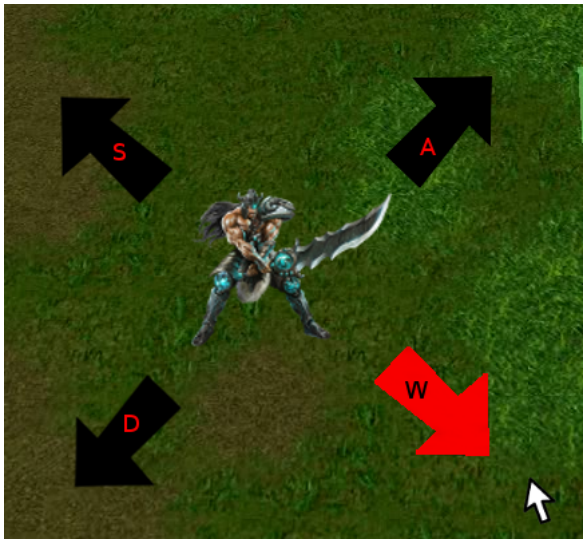

Figura 3: Ejemplo de movimiento *mouse* + teclas

- Click Derecho: El *campeón* utiliza su habilidad, si su habilidad es lanzada en alguna dirección, esta debe ser en la dirección en la que esté mirando al momento de apretar este botón.

Si oprimes cualquier tecla de movimiento mientras el *campeón* se encuentre atacando, dejará inmediatamente de hacerlo y se moverá según la tecla presionada.

Además cuando pases el puntero del *mouse* sobre alguna entidad *atacable* se debe resaltar su contorno para indicar al jugador que puede presionarla para ir a atacarla como se muestra en la figura 2.

Figura 4: Puntero del *mouse* sobre *Nexo* enemigo

8. Cheat Codes [6,6 %]

A ningún juego le pueden faltar las claves para tener beneficios adicionales y éste no será la excepción. *League of Progra* debe ser capaz de activar algunos beneficios si es que el usuario **presiona de forma consecutiva** un conjunto de 3 letras durante una partida y mientras un *Nexo* siga con vida. Deben poder utilizarse la cantidad de veces que uno quiera. Los beneficios que deben estar implementados son:

- *life*: Recupera toda la vida del *campeón*, *súbditos* y edificios que aun siguen de pie de tu equipo. Por lo tanto, si tu *torre* fue destruida, esta clave no tendrá efecto en ella. Letras a presionar: L, I, F
- *ret*: En caso de estar muerto, renaces en ese mismo instante. En caso de estar vivo, esta clave no tiene efecto. Letras a presionar: R, E, T
- *rec*: Reconstruye todos los edificios destruidos y las deja con la mitad de su vida. Solo afecta las edificaciones que ya fueron destruidas. Letras a presionar: R, E, C

9. Interfaz [13,3 %]

La interfaz debe tener ciertos requisitos mínimos. Previo a empezar el juego, debe desplegarse un menú inicial que deje empezar como un nuevo jugador o continuar con los datos almacenados (estadísticas de partidas guardadas). Luego, debe dar la opción de comenzar una partida y antes de esta, elegir el campeón a usar.

Luego de iniciar una partida, la interfaz debe presentar botones para salir al menú principal, ir a la tienda y otro para poner pausa. Estas mismas acciones se deben poder realizar presionando la tecla `I` para salir al menú principal, `O` para ir a la tienda y `P` para poner pausa. La vida de todas las entidades debe ser representada por una barra que muestra su vida actual en comparación a la máxima. También deben estar visibles los valores de los atributos del *campeón* que se está usando junto a los objetos que han sido comprados.

Al abrir la *Tienda*, la interfaz debe desplegar una nueva ventana que muestra los objetos disponibles con sus respectivos valores y permitir realizar las compras.

10. Representación gráfica [5 %]

El *campeón* debe tener movimientos animados. Para lograr esto, se debe diferenciar su aspecto al moverse. Además, se deben observar al menos 3 estados de movimiento.

Figura 5: Diferentes estados de movimiento en una dirección

De esta forma, el movimiento se verá aún más real. Para esta sección, se le recomienda buscar en internet sprites (**pinchar aquí** para ver que son). A continuación dejamos un ejemplo de un sprite sheet para un *campeón*:

Figura 6: Sprite Sheet para un *campeón*

11. Diagrama de Secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos de una aplicación a través del tiempo, en el cual se indicarán los módulos o clases que formarán parte del programa y las llamadas que cada uno de ellos harán para realizar una tarea determinada.

Para esta tarea deberán realizar tres diagramas de secuencia. Uno para cuando el *campeón* se mueve, otro cuando el *campeón* ataca y el último para cuando se muere el *inhibidor*.

A continuación dejamos un ejemplo de un diagrama de secuencia:

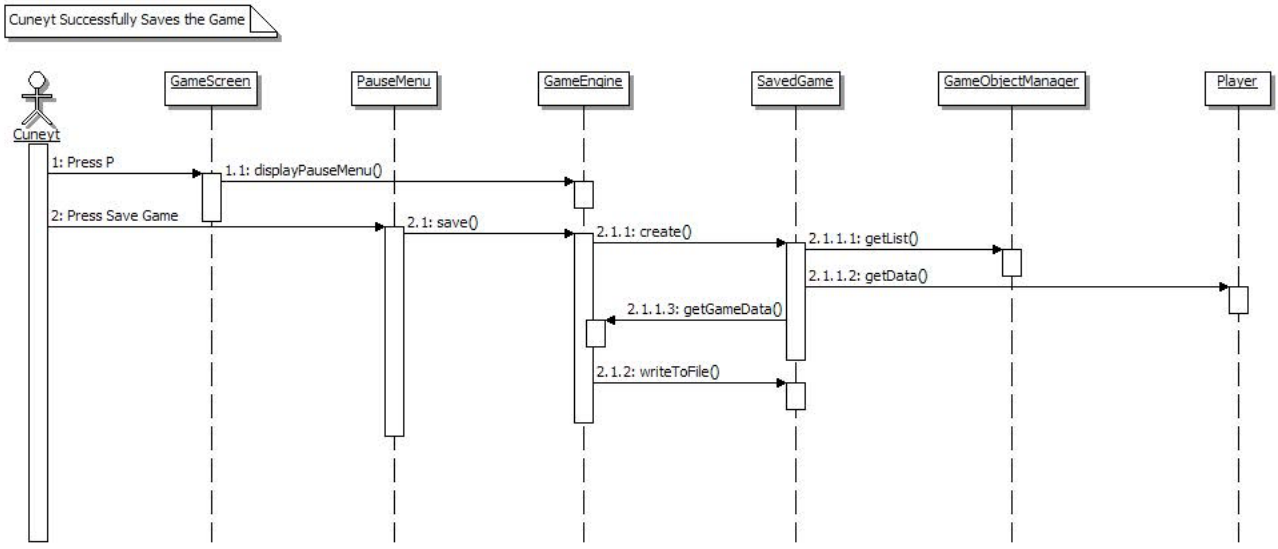

Figura 7: Diagrama de secuencia de guardar una partida de un juego

Pueden encontrar más información de este tipo de diagrama aquí. No es necesario un 100 % de exactitud

en las formalidades del diagrama, lo importante es que revisen bien todo lo que debe interactuar en cada diagrama.

12. Notas

- Cualquier detalle que no esté mencionado en el enunciado realice supuestos razonables.
- Tienen toda la libertad de cambiar cualquier constante si lo estima adecuado para mejorar la experiencia del juego.

13. Restricciones y alcances

- Desarrolla el programa en Python 3.5 y PyQt5.
- La tarea es individual, y está regida por el Código de Honor de la Escuela: Click para Leer.
- Su código debe seguir la guía de estilos `PEP8`.
- Si no se encuentra especificado en el enunciado, asuma que el uso de cualquier librería Python está prohibido. Pregunte en el foro si se pueden usar librerías específicas.
- El ayudante puede descontar el puntaje de hasta 5 décimas de tu tarea, si le parece adecuado. Recomendamos ordenar el código y ser lo más claro y eficiente posible en la creación de los algoritmos.
- Adjunte un archivo `README.md` donde comente sus alcances y el funcionamiento de su sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*. Tiene hasta 24 horas después de la fecha de entrega de la tarea para subir el `README.md` a su repositorio.
- Separe su código de forma inteligente, estructurándolo en distintos módulos. Divídalo por las relaciones y los tipos que poseen en común. **Se descontará hasta un punto si se entrega la tarea en un solo módulo.**
- Cualquier detalle no especificado queda a su criterio, siempre que no pase por sobre los requerimientos definidos en el enunciado.

14. Entrega

14.1. Diagrama de secuencia

- **Fecha/hora:** sábado 27 de mayo del 2017, 23:59 horas.
- **Lugar:** Se abrirá un cuestionario en SIDING.

14.2. Código

- **Fecha/hora:** lunes 5 de junio del 2017, 23:59 horas.
- **Lugar:** GIT - Carpeta: Tareas/T05
- Se deben subir todas las imágenes utilizadas en su programa, en una carpeta llamada `IMGS`, que se tiene que encontrar dentro de su carpeta `T05`. Su programa debe leerlas desde ahí.

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).