TP INTEGRADOR PDP

**NOSOYIMPOSTOR**

**Polimorfismo**: ¿Cuál es el mensaje polimórfico? ¿Qué objetos lo implementan? ¿Qué objeto se aprovecha de ello?

* objetos izquierda y derecha

El mensaje polimórfico mover(unObjeto), que mueve a unObjeto en la dirección izquierda o derecha, es implementado por los objetos izquierda y derecha. Los objetos que se aprovechan de ella son el personaje (player), el tiro (tirito) y el enemigo (feind).

* Estado del personaje

El mensaje polimórfico es encuentroEnemigo(cant) y recibeDisparo(disparo) en player. Al comienzo del juego, el personaje comienza con un estado simple mortal, donde pierde vida cuando se choca con el enemigo o este le dispara. Cuando el personaje encuentra a su mascota, este cambia a un estado semidios donde no le afectan los enemigos, y luego de un tiempo la mascota se va y vuelve a ser un simple mortal.

* Movimiento de la mascota

El mensaje polimórfico es position(), en Mascota. La mascota se encuentra en e stado esperando (quieto) hasta que su duenio lo encuentra y ahí pasa a seguirDuenio, donde sigue a su duenio un bloque a la derecha.

**Colecciones**: ¿Qué operaciones de colecciones se utilizan? ¿Usaron mensajes con y sin efecto? ¿Para qué?

* En disparos, en tirosPlayer y tirosEnemigo usamos una Lista de tiros para cada uno. Los mensajes con efecto utilizados fueron:

Tiros.add(tirito): para agregar el tiro a la lista de tiros

Tiros.remove(disparo): para sacar el tiro de la lista de tiros

* En enemys, en enemigos usamos una Lista de enemigos y una lista de enemigos que disparan. Los mensajes sin efecto utilizados fueron:

Lista.size(): para saber cuántos enemigos hay en la lista

Los mensajes con efecto utilizados fueron:

Lista.add(enemigo): para agregar enemigos a la lista

Lista.remove(enemigo): para sacar enemigos de la lista

enemigos.listaEnemigosDisparo().forEach({enemy => enemy.disparar()}): para activar el disparo de cada enemigo en la lista

enemigos.listaEnemigos().forEach({enemy => game.onCollideDo(enemy, {tiro => tiro.encuentra(enemy,tiro)})}): para cuando un disparo del player choca contra un enemigo, entonces lo elimine o le baje vida.

**Clases**: ¿Usan clases? ¿Por qué? ¿Dónde o cuándo se instancian los objetos?

Usamos las siguientes clases:

* Player: usamos esta clase ya que tenemos 3 modelos de jugadores distintos, que deben heredar la clase Player: los objetos caro, fran y cami.
* Feind: usamos esta clase para modelar los distintos enemigos que tendrán distintas características particulares, pero su base es la misma. A su vez esta se divide en las clases FeindSimple, FeindDispara y FeindPapota. Estos a su vez son clases ya que necesitamos conocer varios de cada uno, iguales.
* Tirito: usamos esta clase por una razón similar a la de Feind, ya que tenemos dos tipos de tiros: el tiro del player y el tiro del feind. Estos también son clases ya que necesitamos conocer varios de cada uno, iguales.
* Mascota: usamos esta clase ya que tenemos 3 modelos de mascotas que hacen lo mismo, pero la única diferencia es el method image() de cada uno. Como hacen todos lo mismo, para evitar repetir código usamos la clase mascota.

**Herencia**: ¿Entre quiénes y por qué? ¿Qué comportamiento es común y cuál distinto?

* Entre Player y los objetos caro, fran y cami, ya que estos utilizan todos los métodos declarados en su clase padre Player. Todos los comportamientos son comunes (como moveIzq, moveDer, perderVida(cantidad), encuentra(enemigo,tiro), entre otros) excepto por que cada uno tiene su propia imagen particular y una imagen para el tiro, por lo que cambia el method image().
* Entre Feind y las clases FeindDispara, FeindSimple y FeindPapota, ya que estos son subtipos del Feind, comparten la misma base con algunos cambios. Por ejemplo, algunos de los comportamientos que se mantienen son encuentra(enemigo,tiro), encuentro(jugador), enElElevator() y move(nuevaPosicion). Los comportamientos distintos son el de la imagen, ya que cada uno tiene una distinta, el method ataque, ya que cada uno devuelve un valor distinto, y disparar ya que el FeindSimple no dispara.
* Entre Tirito y las clases TiroEnemigo y TiroPlayer, ya que estos son subtipos del Tirito y deben heredar ciertos aspectos para evitar que se repita código. Los métodos que se repiten son move(nuevaPosiicon), encuentro(player), recibeDisparo(param1) y enPantalla(). Los métodos que son distintos son: en TiroPlayer, encuentra(enemigo,tiro) tiene un efecto mientras que en TiroEnemigo no; el método image() ya que tienen imágenes distintas; y el method remove() ya que en TiroPlayer se remueve a si mismo de la lista de tirosPlayer y en TiroEnemigo se remueve a si mismo de la lista de tirosEnemigo.
* Entre Mascota y los objetos mora, junior y unicornio, ya que estos usan todo lo de la clase padre (menos image()).

**Composición**: ¿Qué objetos interactúan? ¿Dónde se delega? ¿Por qué no herencia?

* Cremos que no :3

