Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

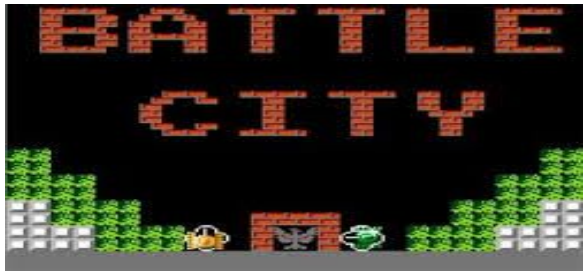
Programación Orientada a Objetos

Proyecto Programado 2

Prof. Mauricio Avilés Cisneros

Andrei Leon Salas(201015265)
Darío Monestel Corella (2014073400)
José Pablo Murillo Vargas(2014055704)

I periodo
2016



Battle City

1. Introducción

1.1 ¿Por qué se hace este proyecto y qué se incluye?

Battle City es un videojuego de tanques producido y publicado por Namco como una adaptación del clásico arcade Tank Battalion.

El juego consiste en controlar un tanque sobre un escenario plagado de tanques enemigos. Su misión consiste en evitar que destruyan su base militar(logo águila). Se completa el nivel cuando haya destruido todos los tanques enemigos. El número de niveles a superar varía dependiendo de la revisión del juego que se esté jugando. El juego termina si el enemigo destruye la base o el jugador pierde todas sus vidas al ser atacado.

Battle City fue uno de los primeros juegos en permitir el juego simultáneo de dos jugadores, ambos jugadores defendiendo conjuntamente la base. Si uno de ellos disparaba al otro, éste queda congelado unos instantes (aunque puede seguir disparando). Asimismo, si por error se disparaba contra la base una vez quedara desprotegida, se destruía y acababa el juego. También fue uno de los primeros juegos en permitir la creación de niveles personalizados, si bien no podía ser guardados en el cartucho.

Este proyecto fue realizado con el propósito de practicar el uso de herencia, que facilita mantener una buena jerarquización en el código, la extensibilidad y favorece la reutilización de código en el programa.
Además se usará Polimorfismo para ser aplicado en los diferentes comportamientos que tomarán los tanques durante la ejecución del juego, además se implementará el diseño de diagramas UML para ayudar a los desarrolladores del programa a llevar una elaboración del software más adecuada.

Así se reforzará la habilidades de modelado de aplicaciones de software y los conceptos previos vistos en el proyecto anterior, para elaborar de manera óptima este producto de software planteado como propuesta propia.

Así mismo, se usará el patrón de diseño Factory Method es definir una interfaz o clase abstracta para crear objetos, sin especificar la clase exacta de los objetos que serán creados. Esto permite crear un objeto que a veces requiere procesos complejos que no deberían incluirse en el código de la clase que lo necesita. Permite evitar la duplicación de código y maneja la información que no debe ser accesible para la clase cliente. Su papel en la programa será que el controlador va a llamar al Factory Method donde va a recibir posición “x” , “y” y un int que indica el poder a retornar.

Flujo de juego:

Se empezará el juego controlando un tanque el cual estará localizado en un escenario determinado con ríos, bosques, muros de roca y metálicos los cuales son indestructibles, además se estará en presencia de varios tanques enemigos los cuales tendremos que matar, sin embargo el juego cuenta con una opción para agregar un segundo jugador el cual juega un rol de aliado que nos ayudará a cumplir el objetivo de matar los tanques enemigos y proteger la base de los invasores, no obstante si el usuario ataca el mismo la base o nuestro aliado se perderá de manera automática el juego, cabe mencionar que antes de empezar a disparar con el tanque el jugador debe primero movilizar la tanque hacia una dirección.
Luego al matar todos los tanques en el campo de batalla saldrá un mensaje de victoria, se contará con un total de **4** niveles, que se seleccionan de forma manual por parte del usuario. Cabe agregar que a lo largo del juego aparecerán poderes en el campo de batalla los cuales podremos tomar para facilitar nuestro objetivo durante la ejecución del juego.
Entre los cuales podemos mencionar :

Casco: se aumenta en 100 unidades la vida del jugador, este es acumulativo durante el nivel.

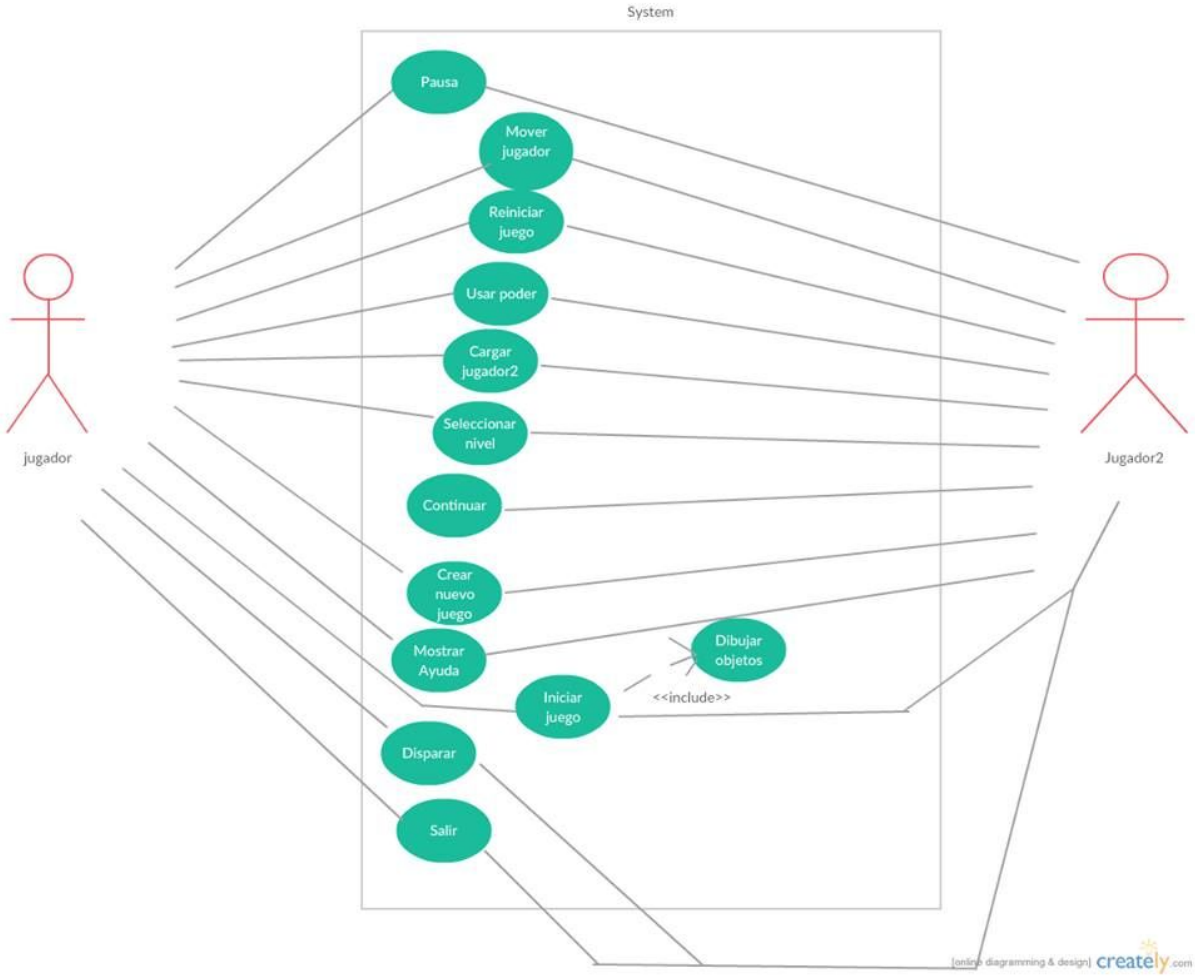
Estrella: Incrementa la velocidad de disparo de del jugador al doble

Tanque: .Incrementa la velocidad de movimiento del tanque del jugador en un 25% inicialmente, este poder es acumulativo durante el nivel

El juego finalizará cuando se mate todos los tanques del último nivel, el cual nos mostrará un mensaje de victoria indicando que fuimos victoriosos en el campo de batalla.

2.Casos de uso

2.1 Diagrama de casos de uso



2.2 Descripción de los casos de uso

Nombre	Mover jugador
Objetivo	Guardar la última Tecla que presiono el jugador para mover el tanque
Precondiciones	Haber creado la partida Que el tanque esté vivo
Condición terminación exitosa	El tanque se mueve en la dirección especificada
Condición de terminación fallida	
Desencadenador	El usuario presiona una tecla
Flujo Principal	El usuario presiona una tecla Si la tecla es permitida, la tecla se guarda a su equivalente de dirección. El tanque jugador cambia su posición x, y dependiendo de la dirección El sistema llama a la función framPaint() para colocar el objeto en su nueva posición
Extensiones	2.1La tecla inválida es ignorada y el jugador no se mueve a una nueva posición

Nombre	Disparar
Objetivo	Crear una instancia de bala, para que luego colisione con un objeto en el juego
Precondiciones	Haber creado la partida Que un tanque esté vivo Que el tanque halla llamado la función disparar()
Condición terminación exitosa	La bala es creada y agregada a la lista de balas del sistema
Condición de terminación fallida	
Desencadenador	El usuario presiona una tecla
Flujo Principal	El usuario presiona una tecla Si la tecla es permitida, la tecla llama la función disparar() La bala es creada y se muestra en pantalla El sistema guarda la bala en la lista de balas El sistema valida que la bala choque con los otros objetos La bala se va moviendo en la pantalla La bala desaparece una vez que choque con un objeto o los bordes de la ventana
Extensiones	2.1La tecla inválida es ignorada y la bala no es creada

Nombre	Usar Poder
Objetivo	Usar el poder para modificar los objetos en el juego
Precondiciones	Haber creado la partida Que el jugador esté vivo Que el jugador halla pasado encima del poder Que el poder esté en pantalla
Condición terminación exitosa	El jugador usa el poder.
Condición de terminación fallida	
Desencadenador	El jugador pasa por un poder
Flujo Principal	El jugador se moviliza dentro de la ventana El jugador pasa por un poder El poder, dependiendo de su tipo, cambia los elementos de la ventana.
Extensiones	

Nombre	Dibujar Objetos
Objetivo	Dibujar los objetos en la ventana
Precondiciones	Haber creado la partida Haber creado los objetos
Condición terminación exitosa	Todos los objetos se muestran en pantalla
Condición de terminación fallida	
Desencadenador	El jugador pasa cerca de un poder
Flujo Principal	Se crea la partida Se crean los objetos El sistema recorre las listas de objetos y válida que aún estén vivos Los objetos vivos se dibujan en pantalla
Extensiones	

Nombre	Reiniciar juego
Objetivo	El juego vuelve a su estado inicial
Precondiciones	La partida debe estar creada
Condición terminación exitosa	Se crea una nueva partida
Condición de terminación fallida	
Desencadenador	El jugador presiona la tecla “R” para reiniciar la partida
Flujo Principal	El jugador selecciona la opción de “Reiniciar” El sistema recibe el comando y regresa a su estado original de inicio.
Extensiones	

Nombre	Seleccionar nivel
Objetivo	El juego carga una ventana con un diseño de tablero dependiendo del nivel
Precondiciones	La partida debe estar creada
Condición terminación exitosa	Se carga el nuevo nivel
Condición de terminación fallida	
Desencadenador	El usuario selecciona un Nivel del menú
Flujo Principal	El jugador selecciona un nivel desde el menú El sistema recibe el comando El sistema borra la partida actual y carga un mapa nuevo
Extensiones	

Nombre	Cargar Jugador2
Objetivo	El jugador2 aparece en la pantalla
Precondiciones	La partida debe estar creada
Condición terminación exitosa	El jugador2 se coloca en la pantalla y puede moverse y disparar como el jugador1
Condición de terminación fallida	
Desencadenador	El jugador selecciona Addition->Agregar jugador2
Flujo Principal	El jugador selecciona agregar jugador2 del menú Addition El sistema recibe el comando y hace que el jugador2 aparezca en pantalla
Extensiones	

Nombre	Crear nuevo juego
Objetivo	El juego crea una nueva partida
Precondiciones	La partida debe estar creada
Condición terminación exitosa	Se crea una nueva partida
Condición de terminación fallida	
Desencadenador	El jugador selecciona “Crear nueva partida”
Flujo Principal	El jugador selecciona la opción de “Crear Partida” del menú “Juego” El sistema recibe el comando El sistema borra la ventana y crea una nueva El sistema crea nuevos objetos Los objetos son colocados en la pantalla
Extensiones	

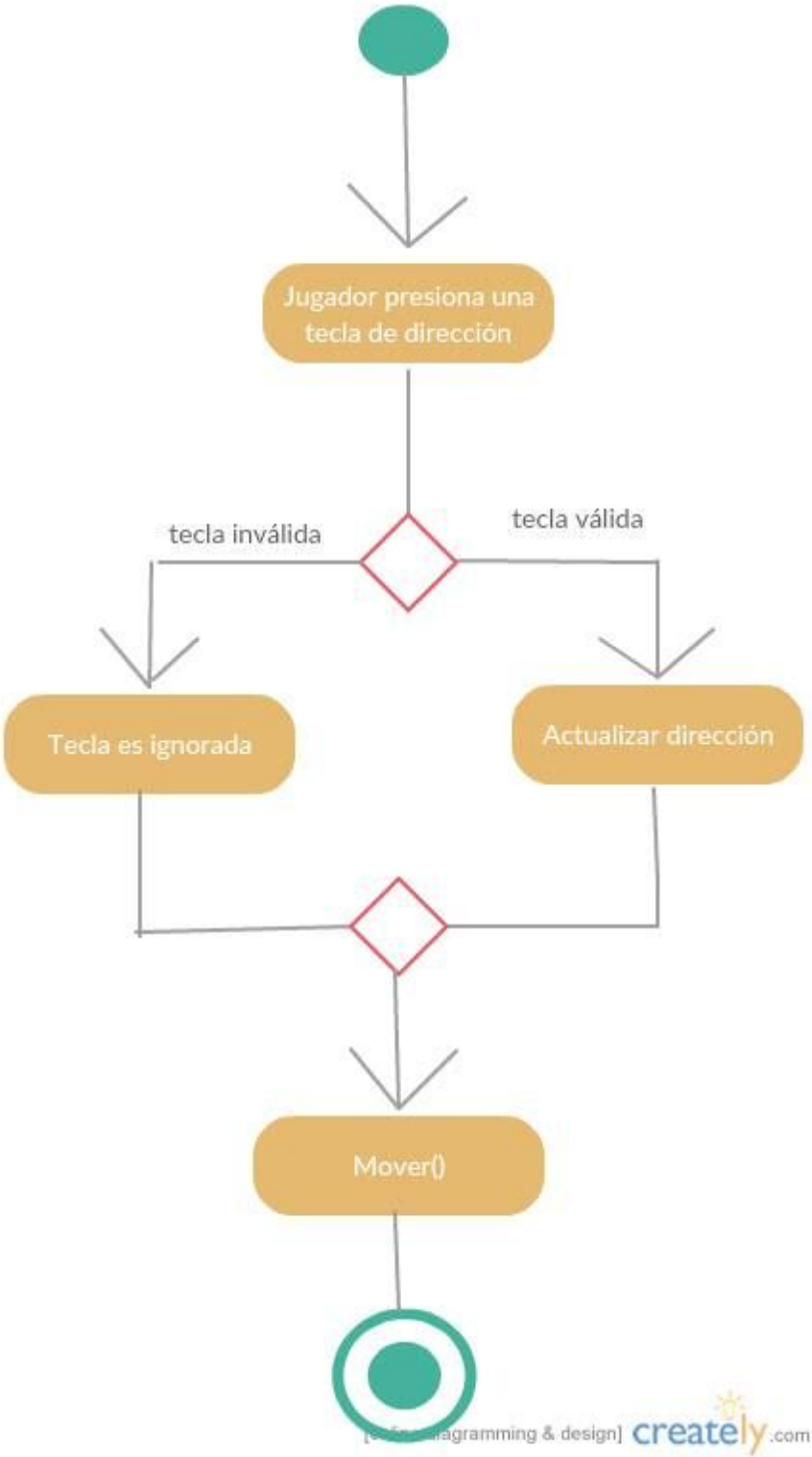
Nombre	Pausa
Objetivo	La ventana se congela y los tanques no se mueven
Precondiciones	La partida debe estar creada
Condición terminación exitosa	La ventana se congela y ningún tanque se mueve
Condición de terminación fallida	
Desencadenador	El jugador selecciona la opción Pausa del menú Pause/Continue
Flujo Principal	El jugador selecciona la opción de “Pausa” El sistema recibe el comando y detiene todos los objetos No se mueve ningún objeto hasta que el usuario le de “Continuar”
Extensiones	

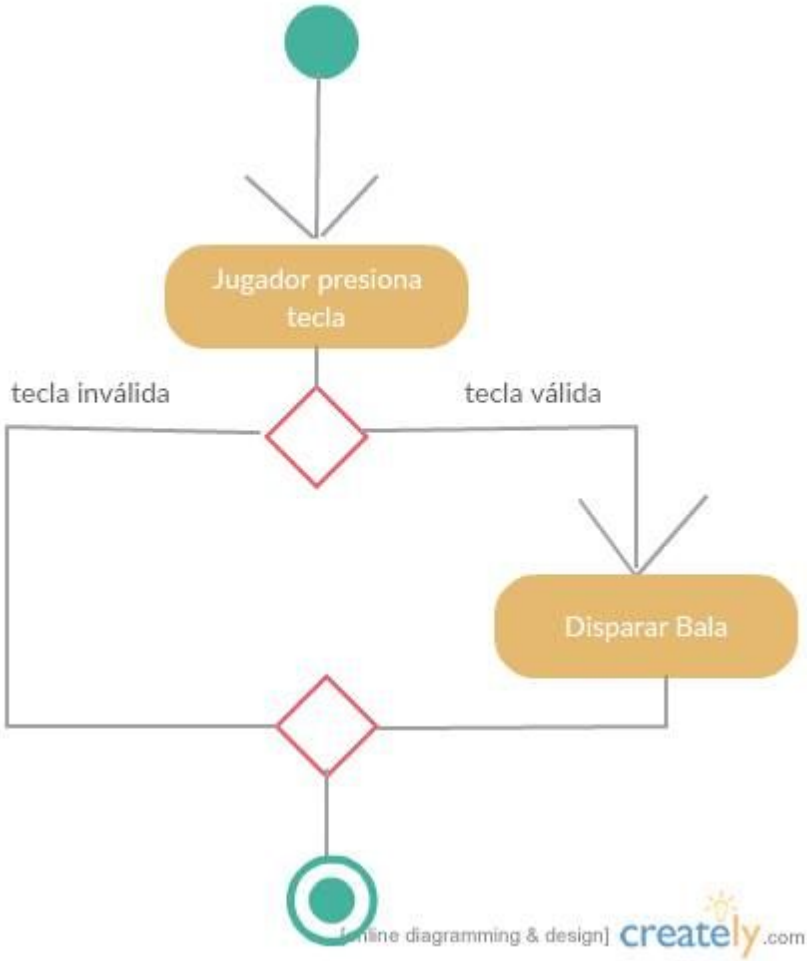
Nombre	Continuar
Objetivo	Los tanques se pueden mover en pantalla
Precondiciones	La partida debe estar creada
Condición terminación exitosa	Se “libera” el movimiento de los tanques en la pantalla
Condición de terminación fallida	
Desencadenador	El jugador presiona la opción “Continuar” del menú “Pausa/Continuar”
Flujo Principal	El jugador selecciona “Continuar” El sistema recibe el comando El sistema libera el movimiento de los tanques
Extensiones	

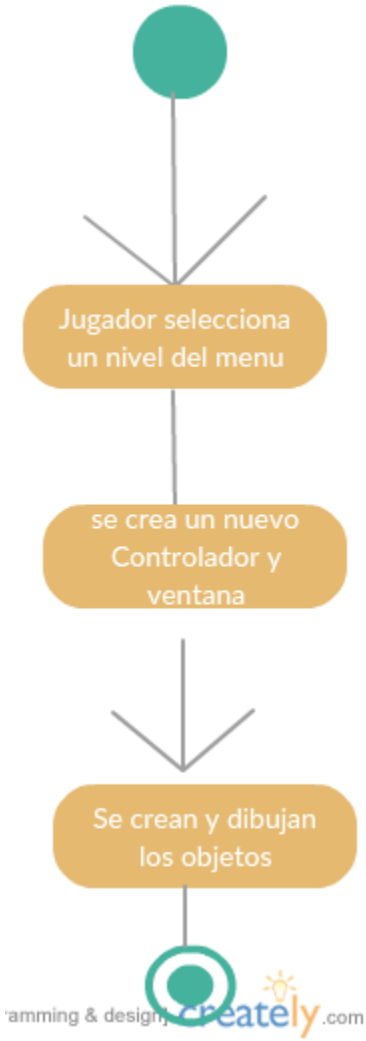
Nombre	Mostrar ayuda
Objetivo	El juego muestra una ventana con la información de cómo mover los tanques
Precondiciones	La partida debe ser creada
Condición terminación exitosa	Se muestra una ventana con la información sobre las teclas
Condición de terminación fallida	
Desencadenador	El jugador
Flujo Principal	El jugador selecciona la opción de “Reiniciar” El sistema recibe el comando y regresa a su estado original de inicio.
Extensiones	

Nombre	Salir
Objetivo	La ventana se cierra
Precondiciones	La partida debe ser creada
Condición terminación exitosa	La ventana se cierra
Condición de terminación fallida	
Desencadenador	El jugador presiona “x” en la ventana o seleccione “Salir” del menú de “juego”
Flujo Principal	El jugador selecciona la opción de “Salir” El sistema recibe el comando y la ventana se cierra
Extensiones	

3. Diagrama de actividad

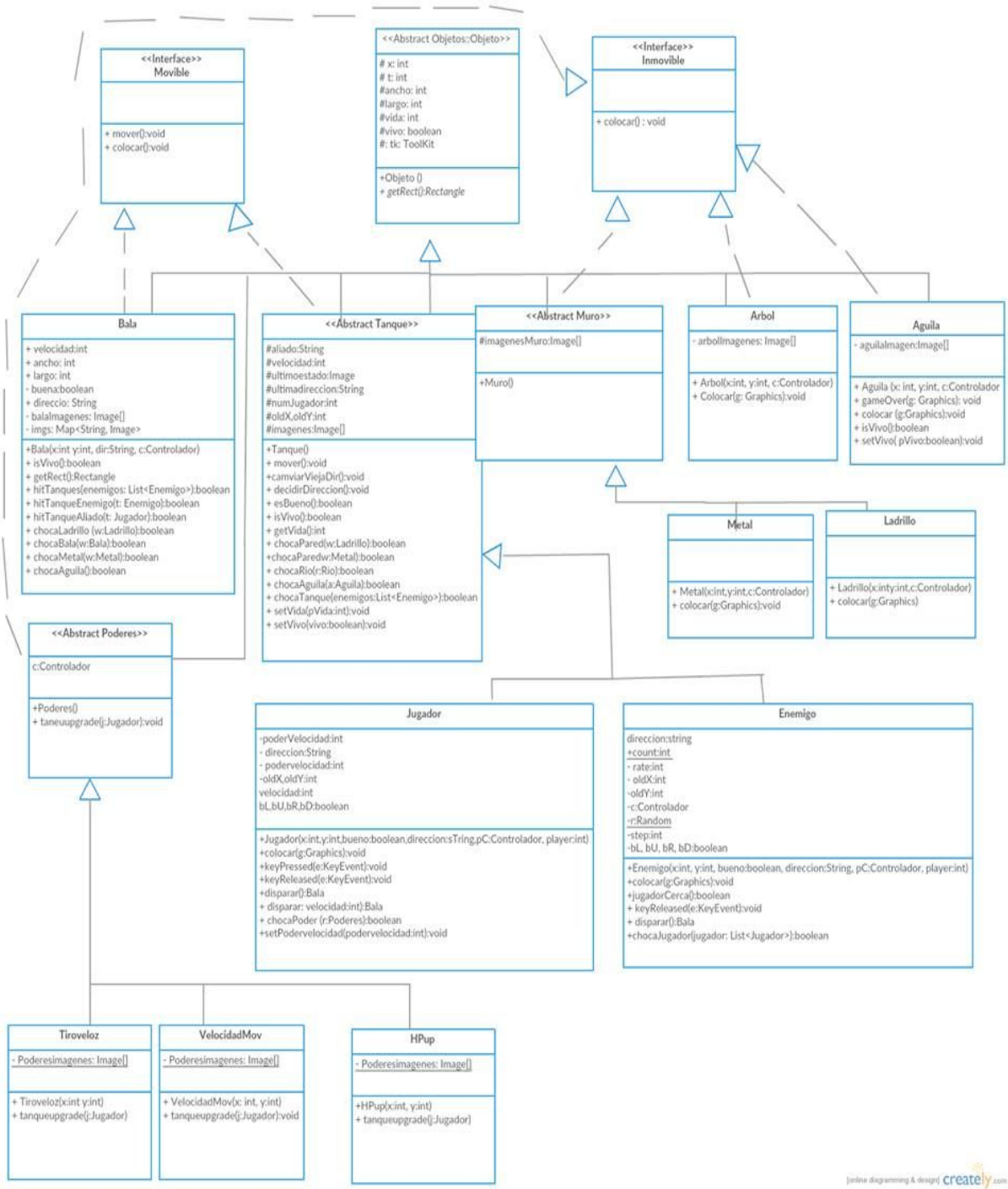


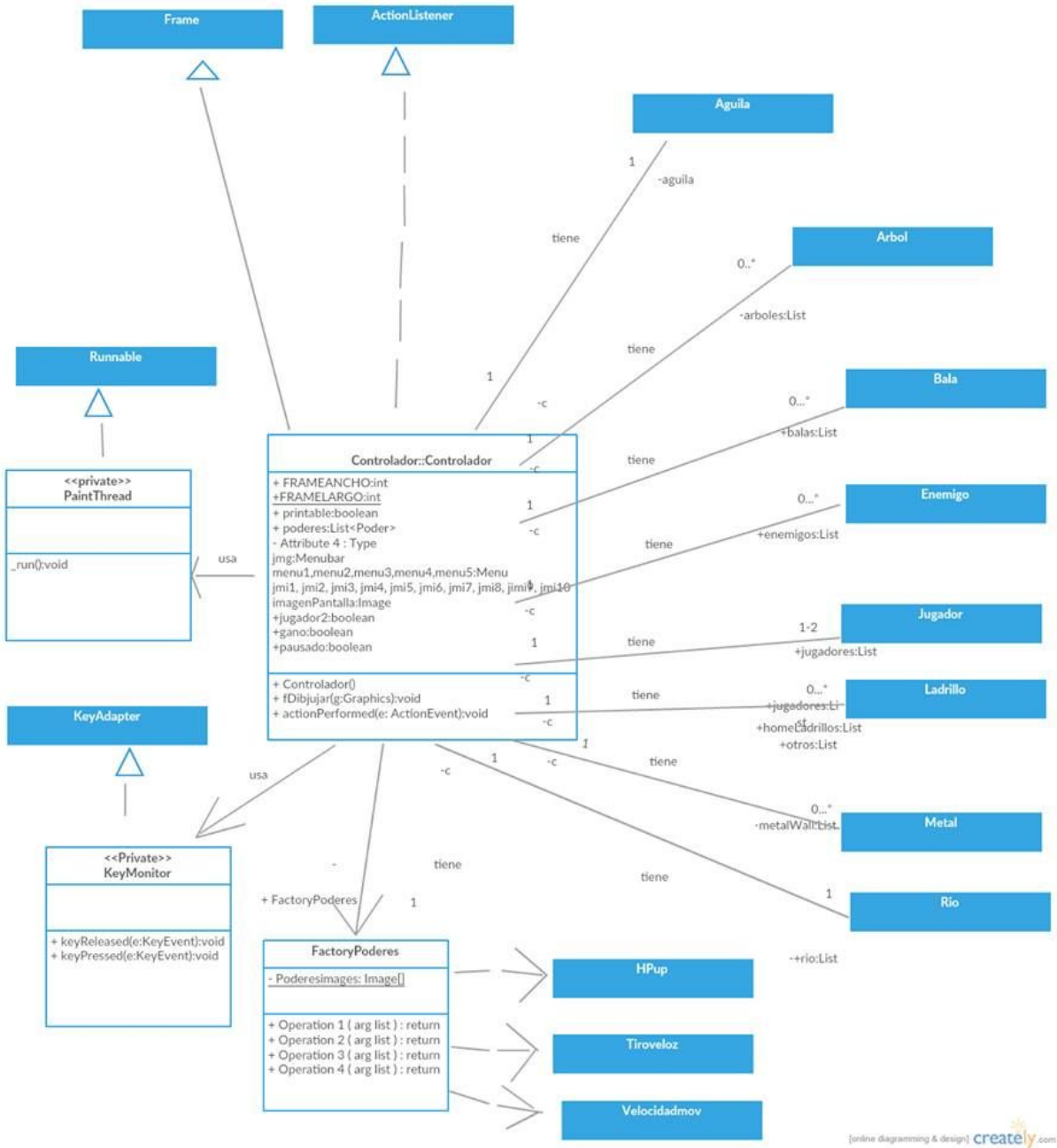

Jugador presiona tecla
tecla inválida
tecla válida
Disparar Bala
[online diagramming & design] creately.com


Jugador selecciona un nivel del menu
se crea un nuevo Controlador y ventana
Se crean y dibujan los objetos

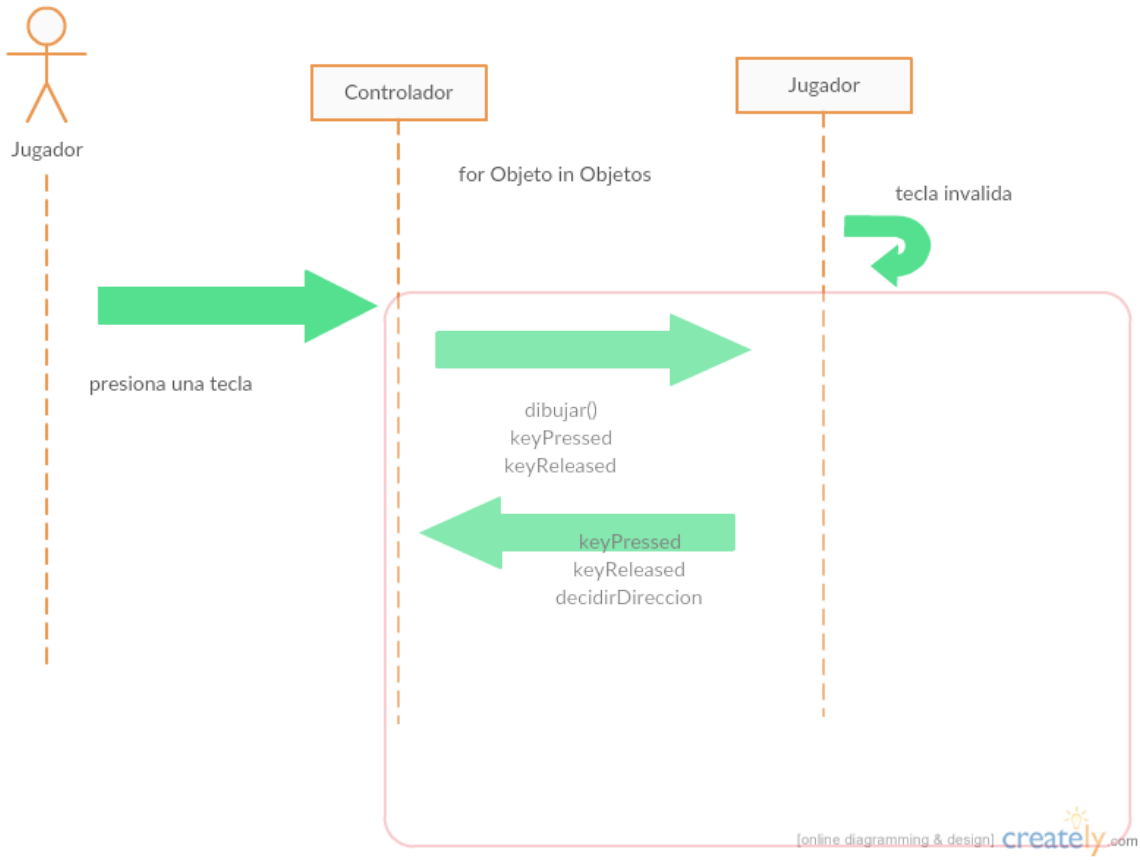
4. Modelo conceptual

4.1 Diagrama de clases



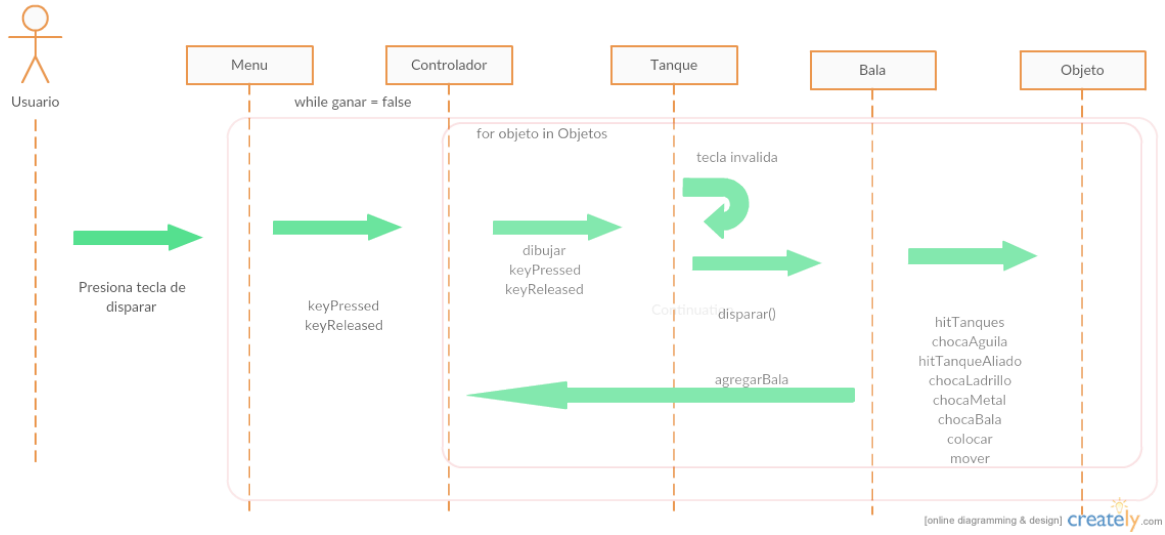

Frame
ActionListener
Aguila
1
-aguila
tiene
Arbol
0..*
-arboles:List
tiene
Runnable
1
-c
Bala
0...*
+balas:List
Controlador::Controlador
+ FRAMEANCHO:int
+FRAMELARGO:int
+ printable:boolean
+ poderes:List<Poder>
- Attribute 4 : Type
jmg:Menubar
menu1,menu2,menu3,menu4,menu5:Menu
jmi1, jmi2, jmi3, jmi4, jmi5, jmi6, jmi7, jmi8, jmi9, jmi10
imagenPantalla:Image
+jugador2:boolean
+gano:boolean
+pausado:boolean
tiene
<<private>>
PaintThread
_run():void
usa
Enemigo
0...*
+enemigos:List
tiene
Jugador
1-2
+jugadores:List
tiene
Ladrillo
0...*
+homeLadrillos:List
+otros:List
KeyAdapter
+ Controlador()
+ fDibujar(g:Graphics):void
+ actionPerformed(e: ActionEvent):void
usa
tiene
Metal
0...*
-metalWall:List
<<Private>>
KeyMonitor
+ keyReleased(e:KeyEvent):void
+ keyPressed(e:KeyEvent):void
+ FactoryPoderes
tiene
Rio
-+rio:List
FactoryPoderes
- Poderesimages: Image[]
+ Operation 1 (arg list) : return
+ Operation 2 (arg list) : return
+ Operation 3 (arg list) : return
+ Operation 4 (arg list) : return
HPup
Tiroveloz
Velocidadmov
[online diagramming & design] creately.com

5. Diagrama de secuencia

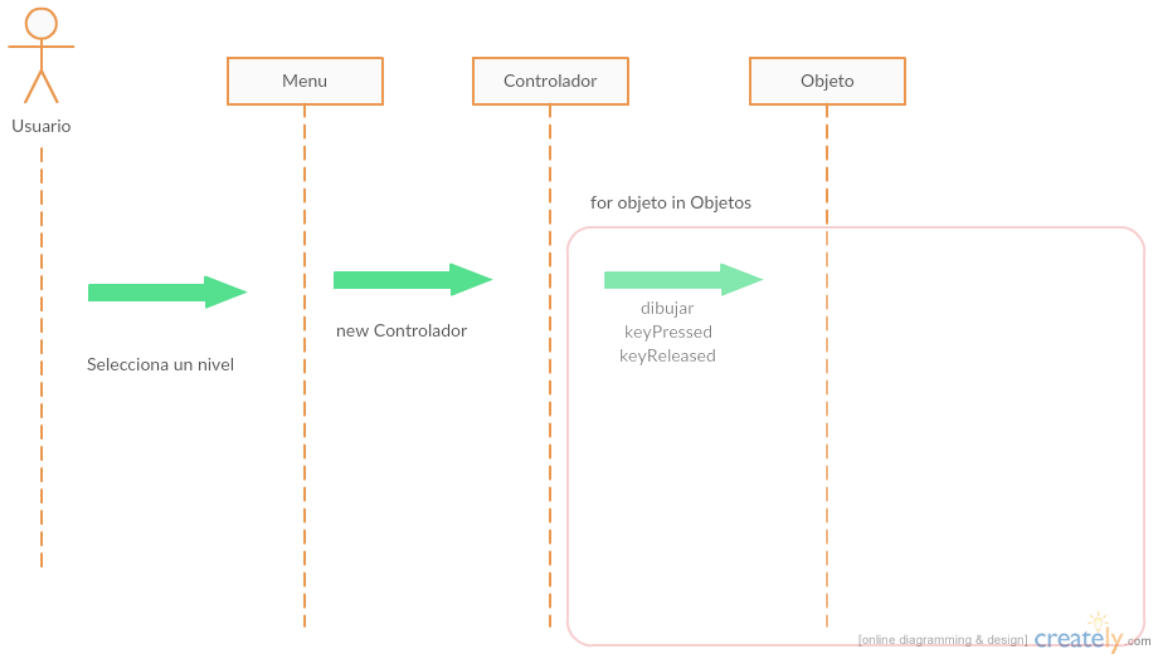


El usuario presiona una tecla. El controlador llama a las funciones: dibujar, keyReleased y keyPressed para dibujar objetos y detectar una tecla

Cuando la tecla es presionada, llama al keyPressed y keyReleased del tanque, el tanque luego decide su dirección y cambia de estado para que el controlador pueda dibujarlo dependiendo de su nuevo estado.



El usuario presiona la tecla de disparar. Mientras no se ha ganado, el menu va a llamar al KeyPressed y keyReleased del controlador que llama al keyPressed y keyReleased del Jugador. El tanque crea una Bala y la Bala recorre las instancias de Clase Objeto haciendo validaciones de choques. La Bala es agregada a la lista de balas del controlador.
El controlador dibuja los objetos.



El usuario selecciona un nivel del menu. El menu notifica al Controlador que hay que desechar el jugo actual y crear una nueva instancia de Controlador. El nuevo controlador dibuja los objetos vivos en la nueva ventana.

6. Conclusiones

- El sistema en línea usado para la generación del diagrama detallado de clases, casos de uso, actividad y secuencia, Creately, es un buen sistema para hacer diagramas UML básicos; sin embargo, debido a la magnitud del proyecto y el hecho de que este servicio es de paga, además de utilizar un sistema de menús complejos y poco intuitivos dificulta la creación de estos diagramas.

- Durante la realización de este proyecto, la mayor fuente de problemas fue hacer uso de la inteligencia Artificial para hacer que los tanques enemigos se movieran de manera inteligente en la pantalla, ya que esto no fue visto en clase y era necesario implementarlo en el proyecto, el método usado fue el algoritmo de A* para la búsqueda de caminos, el proceso de trazar un camino transitable de manera eficiente entre varios.

- El enfoque de separar la interfaz de usuario y lógica de la implementación confeccionada permitió un mayor control sobre el código fuente, como la detección de errores y mejoramiento de algunas funcionalidades. Además indujo a una mejor repartición de los componentes que integran el presente proyecto.
- El lenguaje de programación Java posee una amplia cantidad de componentes pensados para diferentes problemas y necesidades, ejemplos de ello son los contenedores (ArrayList, Map, Stack) que fueron pilares en las abstracciones representadas en el código fuente que se aporta.

- La programación orientada a objetos permite la optimización del código generado gracias a las técnicas del paradigma como la herencia

(serialización de objetos) y los atributos estáticos, estos permiten un trato genérico en el código de manera que sea reutilizable y fácil de adaptar a las condiciones que surjan en el proceso de desarrollo.

- La principal ventaja de la herencia es la capacidad para definir atributos y métodos nuevos para la subclase, que luego se aplican a los atributos y métodos heredados. Esta particularidad permite crear una estructura jerárquica de clases cada vez más especializada. La gran ventaja es que uno ya no debe comenzar desde cero cuando desea especializar una clase existente. Como resultado, se pueden adquirir bibliotecas de clases que ofrecen una base que puede especializarse a voluntad

- El patrón Factory Method permite escribir aplicaciones que son más flexibles respecto de los tipos a utilizar difiriendo la creación de las instancias en el sistema a subclases que pueden ser extendidas a medida que evoluciona el sistema. Permite también encapsular el conocimiento referente a la creación de objetos. Factory Method hace también que el diseño sea más adaptable a cambio de sólo un poco más de complejidad.

7. Recomendaciones

- Definir claramente el problema o necesidad y los objetivos del proyecto. No existe un mayor logro en el área de proyectos que ver que la misma marcha por buen camino, cumpliendo los objetivo que se tenían propuestos.

- Estar motivado (as). Tener presente que el estar motivado es clave para el desarrollo de un proyecto y constituye uno de los factores principales de todo programador. No debe olvidarse que este punto lleva al éxito o al fracaso de un proyecto.

- Llevar a cabo un buen levantamiento de información especialmente a lo que respecta a la situación en la que se encuentre el equipo del proyecto; ya que existen casos en donde se desarrollan proyectos de software donde los conocimientos del individuo son inferiores a los requeridos, este problema se debe a la ausencia de alguna investigación previa.

- Identificar a tiempo posibles fuentes de problemas, este punto va a permitir reaccionar ante cualquier eventualidad y no perder el ritmo de la programación.

- Plasmar el proyecto de la forma más entendible posible de tal manera que quien lea información capte de forma correcta lo que realmente se quiere hacer.

- Crear un código fuente organizado, ya que si se requiere cambiar alguna sección del código esto se pueda realizar cómodamente.

- Evitar agregar un componente a otro antes de crear una instancia de este.

- Hacer uso de algún software para el control de versiones, como Git o Mercurial para el fácil acceso del código actualizado a todos los miembros del trabajo.

- Aplicar el Concepto de Modularidad en la clase controladora para mejorar la legibilidad del código.

- Realizar más variedad de tanques para mejorar el flujo de juego y hacerlo más dinámico.

- Implementar sockets para habilitar el multijugador para crear partidas en línea .

8. Referencias

Moved. *Stack (Java Platform SE 7).* Recuperado el 08 de noviembre de 2015, desde http://docs.oracle.com/javase/7/docs/api/java/util/Stack.html

Herencia (informática), (s. f). En Wikipedia, la enciclopedia libre. Recuperado el 12 de noviembre de 2015.
https://es.wikipedia.org/wiki/Herencia_%28inform%C3%A1tica%29

Algoritmo A* de búsqueda En Wikipedia, la enciclopedia libre. Recuperado el 12 de diciembre de 2007. https://en.wikipedia.org/wiki/A*_search_algorithm

Threads (informática). En Wikipedia, la enciclopedia libre. Recuperado el 4 de noviembre de 2009. https://es.wikipedia.org/wiki/Hilo_de_ejecuci%C3%B3n

Java NetBeans, creating a Maven Swing Application
https://netbeans.org/kb/docs/java/maven-hib-java-se.html