**GIT**

**Laboratorio 1**

Sebastian Villalobos

Profesor:

Roberto Gonzalez

Ayudante:

Nicolas Alarcón

Santiago – Chile

1er semestre – 2020

**Índice**

Portada 01

Índice 02

Introducción 03

Contexto inicial del problema 04

Propuesta técnica 05

Lógicas de juego 06

Marco técnico 07

Ejemplo de funcionamiento 13

Conclusiones 18

**Introducción**

El presente informe tiene por objeto registrar el trabajo realizado para el taller de la clase Paradigmas de programación.

El desarrollo de este proyecto ha sido usando el lenguaje de programación Scheme como respuesta al requerimiento inicial del profesor, además de considerar el paradigma funcional, revisado amplia y detalladamente durante las cátedras. Asimismo, la abstracción de la problemática fue formada con lo que denominamos TDAs, que son representaciones que nos sirven para establecer “cualquier cosa” como una unidad manejable por el lenguaje de programación en el marco del paradigma funcional.

El fin de todo esto, es lograr un programa que sirva, en términos generales, es representar una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

**Contexto inicial del problema**

La problemática es la implementación correcta usando el paradigma de programación “funcional”, y su lenguaje de programación “scheme”, con el que se debe representar de la mejor manera la app “GIT” , utilizando solo las funciones estándar de este, tampoco se podrán utilizar herramientas tales como el set! O funciones similares que emulen el trabajo con variables, adicionalmente cada función deberá estar comentada indicando la descripción de la función, sus argumentos y retorno.

**Propuesta técnica**

Se realiza una solución con drRacket que representa el funcionamiento de la app, dando control a versiones de archivos, para esto se realizaron dos TDAs, que permitieron el desarrollo completo del laboratorio, estos fueron TDA Commit y TDA Zonas.

**Lógicas de la app**

La APP “GIT Scheme”, permite al o los usuarios trabajar colaborativamente en el desarrollo de archivos, manteniendo un total control de los cambios realizados y acceso a estos, la unidad básica que tiene toda la información necesaria para el trabajo son los “commits”, estos almacenan, el nombre del autor de los cambios realizados, el nombre del archivo en el que se realizaron cambios, la fecha en que estos fueron hechos, una descripción general o detallada de los cambios y los cambios realizados.

La aplicación cuenta con 4 zonas de trabajo, que son conformadas por la información de los commits, y estas zonas son manejadas por el usuario para obtener y compartir cambios en los archivos, la primera de ellas es el “Workspace” un entorno de trabajo local donde se trabaja en el o los archivos almacenados en el equipo o descargando las versiones actualizadas desde la nube, la siguiente es el “Index” donde se almacenan los cambios en los archivos realizados en el workspace, siguiendo a este espacio está el “Local Repository” donde los commits son guardados con un mensaje descriptivo, luego está el “Remote repository”, en esta zona se encuentran los commits en con la información compartida para todos los usuarios de estas zonas de trabajo, en forma paralela dentro de las zonas de trabajo existe un espacio que almacena los comandos aplicados por la APP (Logs).

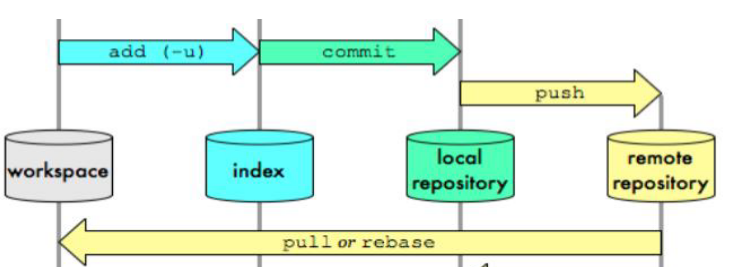
Para trabajar con la app se deben usar los siguientes comandos.

**Pull:** Trae todos los cambios (commits) desde el Remote Repository al Local Repository.

**Add:** Añade los cambios locales del Workspace al Index.

**Commit:** Genera un commit con los cambios almacenados en el Index.

**Push:** Envía los commits del Local Repository al Remote Repository.



**Marco Tecnico**

El programa considera una serie de abstracciones para su ejecución basado en la terminología conocida como TDA, a fin de encapsular las partes esenciales del juego y poder ser codificadas mediante la siguiente estructura técnica definida:

* Capa 0: Representación
* Capa 1: Constructor
* Capa 2: Pertenencia
* Capa 3: Selectores
* Capa 4: Modificadores
* Capa 5: Funciones

**TDA: Space**

Abstracción que contempla el manejo completo del tablero de juego y los elementos que lo componen.

**Alcance y funciones**

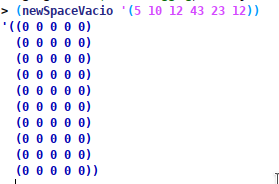
Representación: Lista de 6 elementos → N M A L P Seed

Constructor: (newSpace N M A P L seed)

**Funciones disponibles**

* **s**p*ace? Verifica la integridad del TDA space*  
  Respuesta: #T, #F
* **getN, getM, getA, getP, getL, getSeed** *Obtener el elemento que compone la lista space*  
  Respuesta: el valor del elemento
* **setN, setM, setA, setP, setL, setSeed** *Añadir un elemento nuevo al space*  
  Respuesta: nuevo listado space
* **newSpaceVacio** crea una matriz de elementos 0 con los parametros N y M obtenidos de space, que representa un tablero vacio.

Respuesta: Matriz de 0

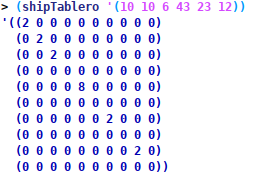
****

* **asteroidesLista** crea una matriz con asteroides ubicados aleatoreamente en el espacio representados por el numero 2, usando la semilla seed

Respuesta matriz

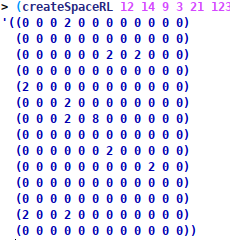
* **shipTableroSpace** inserta la nave en el centro de la matriz.

Respuesta una matriz con los asteroides creados, y en su centro la nave, representada por el numero 8



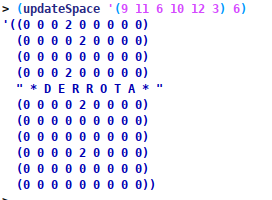
* **createSpaceRL o createSpaceRC** estas funciones utilizan distintos tipos de recursividad en su desarrollo, como entrada reciben cada parametro de space, y generan el tablero con todos los asteroides ubicados de manera random y su nave en el centro del tablero, si se vuelve a crear con el mismo seed, retornará el mismo tablero.

Respuesta La matriz base, lista para iniciar el juego

****

* **updateSpace** funcion que actualiza la posición de los asteroides en el tablero, en mi representación estos se mueven un espacio por cada turno, a la misma velocidad y dirección, esta es una funcion que se utiliza despues de mover la nave

Respuesta una matriz con todos sus elementos en nuevas posiciones y en funcion del turno, si un asteroide golpea a la nave el juego termina y retorna una DERROTA



**TDA: Tablero**

Abstracción que fué necesaria para el trabajo completo del espacio y matriz del juego, este se utilizó en todas las funciones obligatorias, en el se trabajó las listas y elementos individualmente.

**Alcance y funciones**

Representación: Matriz

**Funciones disponibles**

* **getFila** funcion que obtiene la fila n en la matriz

Respuesta lista

* **contadorMatriz** funcion que cuenta el numero de filaz de una matriz

Respuesta numero

* **setLista** una funcion que inserta una nueva lista en el lugar n de la matriz ya creada y reemplaza la que se ubicaba en ese lugar

Respuesta matriz

****

* **getElemento** funcion que obtiene un elemento n de una fila

Respuesta numero

* **contadorFila** funcion que cuenta el numero de elementos de una lista

Respuesta numero

* **setElemento** funcion que inserta un nuevo elemento en una lista ya creada en el lugar n

Resúesta lista

****

* **recorreFilas** funcion que recorre los elementos de la matriz y los compara con algun elemento

Respuesta retorna la cantidad de veces que se repite ese elemento en la matriz

****

* **setElementoFila** funcion que inserta un elemento en una fila de la matriz

Respuesta una nueva matriz con el nuevo elemento isertado en el lugar N,M de la matriz

**TDA: Space**

Abstracción que fué necesaria para el trabajo de representacion de la nave en el juego Asteroids

**Alcance y funciones**

Representación: lista

Constructor: (newShip direction posN posM speed)

Pertenencia: Shiop?

* Getters → direccion, posicion, velocidad
* Setters → direccion, posicion, velocidad
* **setShipTablero** posiciona la nave en un lugar X,Y dek tablero

Respuesta una matriz con la nave

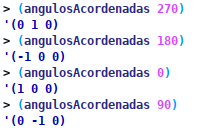
* **getShipPosicion** funcion para saber la localizacion de la nave en el tablero, en cuaquier momento del juego

Respuesta Un par con cordenadas

****

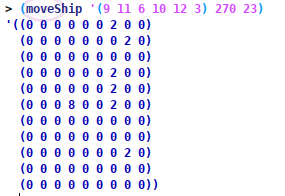
* **angulosAcordenadas** funcion que transforma el angulo en un par de cordenadas

Respusta Un par con cordenadas

****

* **moveShip** funcion principal obligatoria, esta funcion recibe el space, un angulo y un seed, en mi representación esta funcion mueve la nave en la posicion indicada por el usuario

Respuesta una matriz con los asteroides y la nave en su nueva posicion

****

**Ejemplo de funcionamiento funciones obligatorias**

3 EJEMPLOS de cada funcion **createSpaceRL** **createSpaceRC** **checkSpace** **updateSpace** **moveShip**

######################################################

1-

(**createSpaceRL** 9 11 6 10 12 3)

'((0 0 0 0 0 0 2 0 0)

(0 0 0 0 0 0 0 2 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 2 0 0)

(0 0 0 8 0 0 2 0 0)

(0 0 0 0 0 0 0 2 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 2 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0))

(**checkSpace** '(9 11 6 10 12 3))

#t

(**updateSpace** '(9 11 6 10 12 3) 5)

'((0 0 2 0 0 0 0 0 0)

(0 0 0 2 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 2 0 0 0 0 0 0)

(0 0 2 8 0 0 0 0 0)

(0 0 0 2 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 2 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0))

(**updateSpace** '(9 11 6 10 12 3) 6)

'((0 0 0 2 0 0 0 0 0)

(0 0 0 0 2 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 2 0 0 0 0 0)

**" \* D E R R O T A \* "**

(0 0 0 0 2 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 2 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0))

(**moveShip** '(9 11 6 10 12 3) 270 23)

'((0 0 0 0 0 0 2 0 0)

(0 0 0 0 0 0 0 2 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 2 0 0)

(0 0 0 0 0 0 2 0 0)

(0 0 0 8 0 0 2 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 2 0)

(0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0))

######################################################

2-

(**createSpaceRL** 10 12 10 10 12 122)

'((2 0 0 0 2 0 0 0 0 0)

(2 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0)

(2 0 0 0 2 0 0 0 0 0)

(0 0 0 0 2 0 0 0 0 0)

(0 0 0 0 8 0 0 0 0 0)

(0 0 0 0 0 0 0 0 2 0)

(0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0)

(2 2 0 0 0 0 0 0 0 0)

(0 0 2 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0))

(**checkSpace** '(10 12 10 10 12 122))

#t

(**updateSpace** '(10 12 10 10 12 122) 12)

'((0 0 2 0 0 0 2 0 0 0)

(0 0 2 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0)

(0 0 2 0 0 0 2 0 0 0)

(0 0 0 0 0 0 2 0 0 0)

(0 0 0 0 8 0 0 0 0 0)

(2 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0)

(0 2 0 2 0 0 0 0 0 0)

(0 0 0 0 2 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0))

(**moveShip** '(10 12 10 10 12 122) 180 12)

'((2 0 0 0 2 0 0 0 0 0)

(2 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0)

(2 0 0 0 2 0 0 0 0 0)

(0 0 0 0 2 0 0 0 0 0)

(2 0 0 8 2 0 0 0 0 0)

(0 0 0 0 0 0 0 0 2 0)

(0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0)

(2 2 0 0 0 0 0 0 0 0)

(0 0 2 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0))

######################################################

3-

(**createSpaceRL** 15 17 13 10 12 4234324)

'((2 0 0 0 0 0 0 0 0 0 0 2 0 0 0)

(0 0 2 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 2 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 2 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 2 2 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 8 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 2 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 2 0 0 0)

(2 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 2 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 2 0 0 0 0 0 0 0 0 0)

(0 0 0 2 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 2 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0))

(**checkSpace** '(15 17 13 10 12 4234324))

#t

(**updateSpace** '( 15 17 13 10 12 4234324) 123)

'((0 0 0 2 0 0 0 0 0 0 0 0 0 0 2)

(0 0 0 0 0 2 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 2 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 2 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 2 0 2 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 8 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 2 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 2)

(0 0 0 2 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 2 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 2 0 0 0 0 0 0)

(0 0 0 0 0 0 2 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 2 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0))

(**moveShip** '( 15 17 13 10 12 4234324) 90 12)

'((2 0 0 0 0 0 0 0 0 0 0 2 0 0 0)

(0 0 2 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 2 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 2 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 2 2 0 0 0 0 0)

(0 0 0 0 0 0 8 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 2 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 2 0 0 0)

(2 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 2 0 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 2 0 0 0 0 0 0 0 0 0)

(0 0 0 2 0 0 0 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 2 0 0 0 0 0 0 0 0)

(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0))

######################################################

**Conclusiones**

Personalmente la realización de este trabajo utilizando el paradigma funcional y el lenguaje de scheme fué un verdadero desafio, la representacion a travez de funciones me ayudó al desarrollo de tareas cotidianas en mi trabajo, ya sea creando scripts de automatización como en la configuración de algunos servicios de monitroreo en REDES, el unico inconveniente que tuve fué el tiempo, con solo las noches, y ayudantias de los sabados me costó mucho llegar al desarrollo completo.